Solving Quantum-Inspired Perfect Matching Problems via Tutte-Theorem-Based Hybrid Boolean Constraints

Moshe Y. Vardi , **Zhiwei Zhang**

Rice University, Houston, TX, USA

{vardi, zhiwei}@rice.edu

Abstract

Determining the satisfiability of Boolean constraint-satisfaction problems with different types of constraints, that is *hybrid* constraints, is a well-studied problem with important applications. We study a new application of hybrid Boolean constraints, which arises in quantum computing. The problem relates to constrained perfect matching in edge-colored graphs. While general-purpose hybrid constraint solvers can be powerful, we show that direct encodings of the constrained-matching problem as hybrid constraints scale poorly and special techniques are still needed. We propose a novel encoding based on Tutte's Theorem in graph theory as well as optimization techniques. Empirical results demonstrate that our encoding, in suitable languages with advanced SAT solvers, scales significantly better than a number of competing approaches on constrained-matching benchmarks. Our study identifies the necessity of designing problem-specific encodings when applying powerful general-purpose constraint solvers.

1 Introduction

Constraint-satisfaction problems (CSPs) [Brailsford *et al.*, 1999] play a fundamental role in mathematics, physics, and computer science. The Boolean SATisfiability problem (SAT) [Biere *et al.*, 2009] is a special class of CSPs, where each variable takes value from {True, False}. Solving SAT efficiently is of utmost significance in computer science, both theoretically and practically [Vardi, 2014]. Though most effort of the SAT community has been put into solving conjunctive-normal-form (CNF) constraints, Boolean satisfiability that also allows hybrid (non-CNF) constraints is used in numerous applications in various areas, e.g., XOR constraints in cryptography [Bogdanov *et al.*, 2011], pseudo-Boolean constraints in discrete optimization [Costa *et al.*, 2009; Dinur *et al.*, 2005; Kyrillidis *et al.*, 2020], and more. The combination of different types of constraints enhances the expressive power of Boolean formulas [Feldman *et al.*, 2005].

Extensive research has been done for general-purpose hybrid Boolean constraints solving by a variety of techniques. Given the remarkable success of modern CNF-SAT solvers [Gong and Zhou, 2017; Eén and Sörensson, 2003; Liang, 2018; Froleyks *et al.*, 2021], a natural approach is to encode hybrid constraints into CNF. Numerous types of encodings with different properties have been developed [Prestwich, 2009]. Some high-level logic-modeling languages such as Answer Set Programming (ASP) [Lifschitz, 2019; Gebser *et al.*, 2019] are also based on CNF encoding. A different technique for hybrid constraints is to handle certain types of constraints natively by devising specialized solvers, such as CNF + XOR [Soos *et al.*, 2009] and pseudo-Boolean solvers [Devriendt *et al.*, 2021; Martins *et al.*, 2014]. In addition, there also exist attempts for handling different types of hybrid constraints uniformly [Kyrillidis *et al.*, 2020; Kyrillidis *et al.*, 2021a; Kyrillidis *et al.*, 2021b; Kyrillidis *et al.*, 2022].

In this paper, we study a new application of hybrid constraints that arises in quantum computing, through quantum-inspired graph theoretical problems related to perfect matching with vertex color constraints. Solving this problem efficiently provides essential insights into how to design quantum experiments [Jooya *et al.*, 2016; Duncan *et al.*, 2020; Krenn *et al.*, 2022] and advance the realization of large-scale quantum computers [Pan *et al.*, 2012] (see Section 4). Extant algorithmic work on this problem is far from developed, providing no scalable tool to the quantum-computing community. Tackling this problem by solving hybrid constraints is promising for several reasons. First, modern constraint solvers have been well-developed to solve large-scale instances. Second, additional properties of graph instances such as symmetry can be easily integrated as extra constraints, which can be difficult for pure graph theoretical algorithms.

Applying general-purpose hybrid-constraint techniques to the problem above is, however, not trivial. A straightforward encoding of our constrained-matching problem leads to a blow-up of the length of the formula, as also observed in [Cervera-Lierta *et al.*, 2021]. A more advanced encoding based on *exact-one* constraints yields a considerably long quantified Boolean formula (QBF) with poor scalability. To address those issues, as the main technical contribution of this work, we propose an encoding for the constrained-matching problem based on Tutte's Theorem in graph theory [Ander-

son, 1971]. Tutte-Berge formula, as a generalized result of Tutte's Theorem, as well as related decompositions, are widely used in computer science and graph theory [Schrijver and others, 2003]. The efficiency of our encoding originates from the short proof of the non-existence of perfect matchings that this encoding offers.

In the experiment section, we evaluated our approach by solving novel benchmark problems that represent important tasks in quantum computing. For a comprehensive comparison, we also implemented a number of competing methods. We demonstrate that our Tutte encoding does not scale well without further optimizations and suitable languages. Specifically, expressing our encodings in high-level languages, such as Answer Set Programming (ASP) or pseudo-Boolean programming (PB), is more convenient. The resulting performance is, however, less than satisfactory. We discover that in conjunctive normal form (CNF) with additional optimizations and advanced SAT solvers, our encodings exponentially outperform other constraint-based and graph theoretical methods, significantly increasing the scalability of the state-of-the-art. Our implementations provide helpful tools for testing quantum properties efficiently. We conclude that in spite of the progress in general-purpose hybrid constraint solving, specialized encoding techniques are still required to obtain a scalable approach for specific problems.

2 Preliminaries

This section includes a background in hybrid Boolean SAT and definitions in graph theory which are needed for defining the quantum-inspired problem related to perfect matching.

Definition 1. (Boolean satisfiability and hybrid constraints) *Let $x = (x_1, ..., x_n)$ be a sequence of n Boolean variables. A Boolean constraint $f(x)$ is a mapping from a Boolean vector $\{\texttt{True}, \texttt{False}\}^n$ to $\{\texttt{True}, \texttt{False}\}$. A formula $f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is the conjunction of Boolean constraints $\{c_i\}_{i=1,\cdots,n}$. There can be multiple types of constraints, e.g.:*

- *Disjunctive Constraints, e.g., $(x_1 \vee x_2 \vee x_3)$.*

- *Cardinality constraint, e.g., $x_1 + x_2 + x_3 \geq 2$, where `False` and `True` are interpreted as 0 and 1 respectively. If the constraint is in the form of $\sum_{x \in X} x = k$, then it can be written as $\texttt{Exact-}k(X)$.*

- *Pseudo-Boolean (linear) constraints, e.g., $3x_1 - 4x_2 + 5x_3 \geq 0$.*

- *XOR constraints, e.g., $x_1 \oplus x_2 \oplus x_3$.*

A set of constraints is satisfiable *if there exists a Boolean assignment (solution) that evaluates all constraints to `True`.*

Next, we introduce concepts that relate to the quantum-inspired graph theoretical problem.

Definition 2. (Bicolored Graphs [Gu *et al.*, 2019a; Vardi and Zhang, 2022]) *In this paper, a "graph" refers to an undirected, non-simple [1] graphs. A bicolored graph G is a tuple (V, E, d), where V is the set of vertices with $|V|$, E is the set of bicolored edges, and d is the number of colors. For*

---

[1]Self-loops are prohibited but multiple edges between the same pair of vertices are allowed.
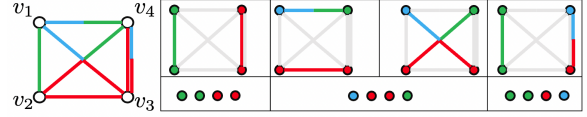

Figure 1: Left: a graph G with bicolored edges. v_3 and v_4 are connected by two edges with different colors. Right: all perfect matchings of G and their inherited vertex colorings [Vardi and Zhang, 2022].

each edge $e \in E$ that connects u and v, there are two colors $c_u^e, c_v^e \in \{1, \cdots, d\}$ that are associated with u and v, respectively, i.e., $e = \{(u, c_u^e), (v, c_v^e)\}$. If $c_u^e = c_v^e$, we call the edge e monochromatic, otherwise e is bicolored. See Figure 1 (left) as an example.

Definition 3. (Vertex coloring) *A vertex coloring of graph $G = (V, E, d)$ is a mapping c from vertices to colors, i.e., $c : V \rightarrow \{1, \cdots, d\}$. Alternatively one can write $c \in \{1, \cdots, d\}^V$. For a coloring c and a color $i \in \{1, \cdots, d\}$, we use $\texttt{count}(c, i)$ to denote the number of appearances of color i in c.*

Definition 4. (Perfect matchings and their inherited vertex colorings) [Krenn *et al.*, 2019] *A perfect matching P of a graph G is a subset of edges, i.e., $P \subseteq E$, such that for each vertex $v \in V$, exactly one edge in P is adjacent to v. For a perfect matching P of a bicolored graph G, its inherited vertex coloring, denoted by $c^P : V \rightarrow \{1, \cdots, d\}$ is defined as follows: for each vertex v, let e be the unique edge that is adjacent to v, then $c^P(v) = c_v^e$. See Figure 1 for examples.*

3 Quantum-Inspired Perfect-Matching-Related Problems

In this section, we introduce the graph-theoretical problem that relates to vertex-color-constrained perfect matching, inspired by quantum computing. Roughly speaking, the problem asks whether for *each* vertex coloring in a set of "legal" vertex colorings, a bicolored graph has at least one perfect matching with this inherited vertex coloring.

Problem 1. (Perfect matchings exist for all legal vertex colorings, abbrv. FORALL-PMVC) *Consider a bicolored graph G and a set of $\mathcal{C} \subseteq \{1, \cdots, d\}^V$ "legal" vertex colorings. Does G satisfy that, for each vertex coloring $c \in \mathcal{C}$, G has at least one perfect matching with inherited vertex coloring c?*

The hardness of FORALL-PMVC depends heavily on the set of legal vertex colorings $\mathcal{C}$. $\mathcal{C}$ can be defined by enumerating all members of the set or by constraints. Below, we list several legal sets of vertex colorings with interest in quantum computing [Gu *et al.*, 2019a; Vardi and Zhang, 2022].

- GHZ State: the legal colorings are monochromatic, i.e., $\mathcal{C} = \{(1, 1, \cdots, 1), (2, 2 \cdots, 2), \cdots, (d, d, \cdots, d)\}$, denoted as $\texttt{GHZ}(|V|, d) = 1/\sqrt{d} \cdot \sum_{i=1}^{d} |i\rangle^{\oplus |V|}$.

- W State: the graph has two colors ($d = 2$). $\mathcal{C} = \{c \in \{1, \cdots, d\}^V | \texttt{count}(c, 1) = 1\}$, i.e, there must be exactly one vertex with color 1 and $|V| - 1$ vertices with color 2, denoted as $\texttt{W}(|V|) = 1/\sqrt{|V|} \cdot \hat{S}(|2\rangle^{\oplus(|V|-1)} |1\rangle^{\oplus 1})$
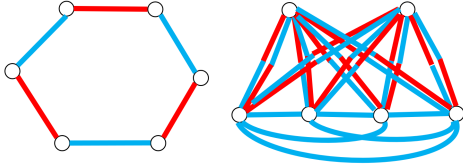
Figure 2: Left: A cycle satisfies the FORALL-PMVC condition of `GHZ(6,2)`. The graph contains perfect matchings for both of two monochromatic vertex colorings. Right: A graph that satisfies the FORALL-PMVC condition for `Dicke(6,2)`. The graph contains perfect matchings for all vertex colorings that have two red vertices.

- Dicke State: a generalization of W State. $\mathcal{C} = \{c \in \{1, \cdots, d\}^V | \text{count}(c, 1) = k\}$ $(d = 2)$, denoted as $\text{Dicke}(|V|, k) = 1/\sqrt{\binom{|V|}{k}} \cdot \hat{S}(|2\rangle^{\oplus(|V|-k)} |1\rangle^{\oplus k})$.

Complete bicolored graphs trivially satisfy the FORALL-PMVC condition for an arbitrary legal coloring set. Nevertheless, graphs with interest in quantum computing are those satisfying the FORALL-PMVC condition with as few edges as possible. In Figure 2, we show examples of such graphs.

The problem of checking the existence of perfect matching is well-known to be polynomial, via the Blossom algorithm [Edmonds, 1965]. We now show that if the size of legal colorings, i.e., $|\mathcal{C}|$, is polynomially bounded, then FORALL-PMVC is tractable. The idea is that one can enumerate all legal colorings and verify the existence of perfect matching individually, as formalized in the following proposition.

Definition 5. (Induced graph w.r.t. a vertex coloring) *Given a bicolored graph $G = (V, E, d)$ and a vertex coloring c, let $G_c = (V, E_c, d)$ be the subgraph where E_c is a subset of E, defined as follows:*

$E_c = \{e | e = \{(u, c_u^e), (v, c_v^e)\} \in E, c_u^e = c(u), c_v^e = c(v)\}.$

In other words., in G_c, we keep only the edges whose colors agree with the vertex coloring c.

Proposition 1. *A bicolored graph G has a perfect matching with inherited vertex coloring c iff G_c has a perfect matching.*

With Proposition 1, we can design a polynomial algorithm for FORALL-PMVC with polynomially many legal colorings, as shown in Algorithm 1. We assume $\text{Blossom}(G)$ is an algorithm that returns True iff. G has a perfect matching.

Algorithm 1: `Enum-Blossom`: An Enumeration-Based Algorithm for FORALL-PMVC

Input : Bicolored graph G, set of legal colorings $\mathcal{C}$.
Output: Whether G satisfies the FORALL-PMVC condition w.r.t. $\mathcal{C}$.

---

1 **for** $c \in \mathcal{C}$ **do**
2   Let G_c be the induced graph of G w.r.t. c.
3   **if** $Blossom(G_C) == False$ **then return** False;
4 **return** True

Algorithm 1 runs in $O(|\mathcal{C}| \cdot |E| \cdot |V|^2)$. Therefore it provides an efficient algorithm for the FORALL-PMVC problem for GHZ State ($|\mathcal{C}| = d$) and W State ($|\mathcal{C}| = |V|$). The

challenging part of FORALL-PMVC is when $|\mathcal{C}|$ is exponential, such that it is infeasible to enumerate $\mathcal{C}$. As an example, `Dicke(n, n/2)` has $\binom{n}{n/2} = \Theta(2^n/\sqrt{n})$ legal colorings.

The lower bound of the computational complexity of FORALL-PMVC for exponentially many legal colorings remains open, though in [Vardi and Zhang, 2022] the complexity of a related problem is addressed for some special cases. It is not obvious how to reduce a well-known NP-hard or W[1]-hard problem to FORALL-PMVC.

Open Question 1. *Is FORALL-PMVC of Dicke State NP-hard?*

Open Question 2. *Is FORALL-PMVC of* `Dicke(n, k)` *fixed-parameter tractable (FPT) or W[1]-hard w.r.t. k?* [2]

As for the upper bound, the problem is in co-NP, provided that checking if a coloring c is in $\mathcal{C}$ is in PTIME.

Proposition 2. *FORALL-PMVC is in co-NP.*

Proof. *If a graph does not satisfy the condition of FORALL-PMVC, then a witness can be obtained as a legal coloring $c \in \mathcal{C}$ such that G_c has no perfect matching. Such a witness c can be verified in polynomial time by the Blossom Algorithm.*

If one views a coloring c as a witness, then the algorithm for checking the witness can be relatively sophisticated, such as the Blossom Algorithm. There exists, however, a more powerful form of witness that allows a simpler checking algorithm, which leads to our approach for solving FORALL-PMVC, as shown in Section 5.

4 Quantum Motivation and Impact of FORALL-PMVC

4.1 Perfect Matchings and Quantum Circuits

The motivation of PMVC from the perspective of quantum-experiment design was originally introduced in [Vardi and Zhang, 2022], which we recall here to make this paper self-contained. An example of a (simplified) *quantum optical circuit* is shown in Figure 3 (left). A non-linear *crystal* (shaded box in Figure 3) can emit a pair of entangled *photons* simultaneously if activated by a laser-pump power. Each photon has a *mode*, encoded by an integer index. Photons emitted by crystals travel on *optical paths*. A *receiver*, which detects the arrival of a photon and identifies photon mode, is placed at the end of each optical path. Each crystal is associated with two optical paths [Gu *et al.*, 2019b].

A *coincidence* of a quantum optical circuit happens when each receiver detects exactly one photon, due to the activation of some crystals. The *quantum state* of a coincidence is the mode of photons caught by all receivers in the coincidence. The quantum state of an optical circuit is the *superposition* [Ballentine, 2014] of the quantum states of all coincidences.

The optical circuit in Figure 3 (left) has 7 nonlinear crystals. A coincidence happens only when crystals in {I,II}, or {I,III}, or {IV,VI} or {V,VII} are activated simultaneously.

---

[2] In [Gaspers *et al.*, 2012], it is proven that deciding whether an uncolored graph has a Tutte set of exactly k vertices is W[1]-hard.
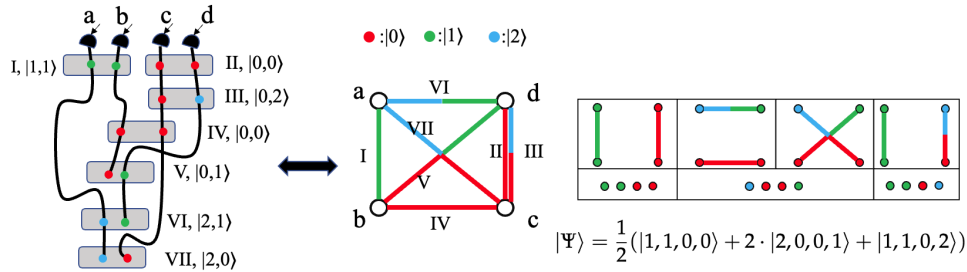
Figure 3: Left: a quantum optical circuit corresponding to the graph in Figure 1. Each crystal (shaded box) corresponds to an edge in the graph. Each optical path is converted into a vertex in the graph. The modes of photons translate to edge colors. Right: all PMs in the graph reflect the quantum state of the optical circuit as a superposition of coincidences. [Vardi and Zhang, 2022]

In [Gu *et al.*, 2019b], a coincidence of an optical circuit is shown to be equivalent to a perfect matching under vertex-color constraints in an undirected graph G with bi-colored edges, as described below.

1. Each optical path corresponds to a vertex in G.

2. Each crystal corresponds to an edge in G. This edge connects two vertices corresponding to two optical paths to which the crystal can emit photons.

3. The modes of photons emitted by crystals correspond to edge colors. Since a crystal can emit two photons with different modes, edges in G are bi-colored.

4. A coincidence of an optical circuit corresponds to a perfect matching P in G. The set of edges in P indicates activated crystals. The quantum state of the coincidence corresponds to the inherited vertex coloring of P.

5. Each crystal has an amplitude as a complex number, corresponding to the edge weight in the graph. The amplitude of a coincidence is the product of the amplitudes of all activated crystals. The weight of a perfect matching is the product of the weights of all its edges.

4.2 Impact of FORALL-PMVC

It is shown in a series of work [Gu *et al.*, 2019a; Gu *et al.*, 2019b] that designing quantum circuits can be done by constructing a graph with bi-colored edges. The ultimate goal of this line of research is to automatically construct bi-colored graphs by combinatorial approaches for arbitrary quantum states, aiming to contribute to the development of large-scale and reliable quantum computers.

The graphs proposed originally in [Gu *et al.*, 2019b] allow complex edge weights, representing the amplitude and phase of the optical components. A quantum state of an optical circuit (resp., graph) can be viewed as a superposition of coincidences (resp., PMs) with predefined "legal" quantum states. An optical circuit (resp., a graph) exhibits a quantum state if 1) the total amplitude (resp., weight) of all coincidences (resp., PMs) of each legal state is 1, and 2) all coincidences (resp., PMs) with an "illegal" state have total amplitude (resp., weight) 0 [Krenn *et al.*, 2017].

The original graph construction problem displays a continuous nature, while in this paper (FORALL-PMVC) as well as [Vardi and Zhang, 2022] (EXISTS-PMVC) we study two

simplified, discrete problems, which can be viewed as focusing on the case where edge weights are real and positive. The simplified, discrete setting is not only important in graph theory but also of interest in quantum experiments. Graphs with real, positive edge weights correspond to circuits with optical components with zero-phase. Such a circuit is efficient to set up. Although the existence of graphs with real, positive weights of GHZ states has been studied [Krenn *et al.*, 2019], the simplified setting still remains open and challenging for many other quantum states, e.g., Dicke State. A key feature of our approach is its generality in terms of algorithmically accepting arbitrary user-defined quantum states. In the future, we aim to extend our approach to allow complex weights.

FORALL-PMVC and EXISTS-PMVC test two conditions of whether a graph can be a candidate for displaying a quantum state, under the real-positive-weight-assumption. The algorithms for solving those two decision problems can be applied as building blocks for further developing the actual graph constructors. For instance, a search-based graph constructor can use those algorithms as oracles for quickly pruning unwanted structures in the space of all bi-colored graphs.

5 Boolean Encodings for FORALL-PMVC

In this section we introduce our hybrid-constraint-based approach for FORALL-PMVC. We show that a natural and straightforward `Exact-One` encoding yields a non-scalable QBF approach. Rather, we exploit the powerful witness of the non-existence of perfect matching provided by Tutte's Theorem to design a novel encoding. We further propose optimization techniques for our Tutte's Theorem-based encoding to enhance the performance.

5.1 The Exact-One Encoding for PM and QBF Encoding for FORALL-PMVC

From the definition of perfect matching, it is natural to use `ExactOne` constraints to encode perfect-matching-related problems. For each $e \in E$, we introduce a variable `edge`$_e$, which is `True` iff e is in the perfect matching:

$$\text{PM} = \{\text{ExactOne}(\{\text{edge}_e\}_{e \in \text{adj}(v)})\}_{v \in V}.$$

PM is satisfiable iff the graph has a perfect matching. The variables $\{\text{edge}_e\}_{e \in E}$ in PM are existentially quantified. As FORALL-PMVC requires that for each legal coloring the

graph has a perfect matching, one can add a universally quantified layer corresponding to vertex colorings. For each vertex v and color i, we define a variable $\mathtt{vc}_v^i$, which is $\mathtt{True}$ iff. vertex v has color i. Then FORALL-PMVC can be expressed in the following $\forall$-$\exists$ quantified Boolean formula (QBF):

$$\forall\{\mathtt{vc}_v^i\}_{v\in V, 1\le i\le d}\exists\{\mathtt{edge}_e\}_{e\in E},$$
$$(\mathtt{ValidColoring} \wedge \mathtt{LegalColoring}) \rightarrow \mathtt{PM}. \quad (1)$$

$\mathtt{ValidColoring}$ is satisfied if the assignment to the vertex-color variables $\{\mathtt{vc}\}$ represents a valid vertex coloring, i.e. each vertex has exactly one color:

$$\mathtt{ValidColoring} = \{\mathtt{ExactOne}(\{\mathtt{vc}_v^i\}_{1\le i\le d})\}_{v\in V}.$$

$\mathtt{LegalColoring}$ includes the constraints that define the set $\mathcal{C}$ of legal coloring. For example, for $\mathtt{Dicke}(|V|,k)$, we have

$$\mathtt{LegalColoring} = \{\sum_{v\in V}\mathtt{vc}_v^1 = k\}.$$

Therefore the QBF formula (1) is satisfiable iff. the graph satisfies the FORALL-PMVC condition. Extant QBF solvers, however, accept only problems in CNF format, which requires encoding the implication and cardinality constraints in (1) to CNF. Even using scalable encoding techniques such as Tseytin encoding [Tseitin, 1983], the length of the CNF-QBF formula increases rapidly as the graph scales, which makes QBF solvers perform poorly, as demonstrated in Section 6.

5.2 A Tutte's Theorem-Based Encoding

We show that it is possible to encode FORALL-PMVC with only existential quantifiers, such that a broader set of solvers, e.g., SAT solvers and hybrid Boolean solvers, can be applied. It is enabled by the witness of the non-existence of perfect matching given by Tutte's Theorem.

Theorem 1. (Tutte's Theorem) *Let the number of odd connected components (connected components having an odd number of vertices) of a graph G be $\#odd(G)$. For a subset $V' \subseteq V$ of vertices, let $G[V']$ be the induced subgraph of G on V'. A graph G has no perfect matching if and only if there exists a set $S \subseteq V$ of vertices such that*

$$\#odd(G[V \setminus S]) > |S|.$$

We call such a set S a Tutte set.

For example, consider the complete bipartite graph $K_{n-2,n}$, where every vertex of the first set containing $(n-2)$ vertices is connected to every vertex of the second set including n vertices. It is easy to see that $K_{n-2,n}$ has no perfect matching. All $(n-2)$ vertices in the first set form a unique Tutte set, since removing those $n-2$ vertices yields n isolated vertices, i.e., n odd connected components.

The idea is, with Tutte's Theorem, we can convert the problem of "finding a legal coloring whose induced graph has no perfect matching" to "finding a legal coloring that allows a Tutte set". If for all legal colorings, there exists no Tutte set, then the graph satisfies the FORALL-PMVC condition. In the following, we introduce the details of our encoding for FORALL-PMVC based on Tutte's Theorem.

For each vertex v, we introduce a variable T_v, which is $\mathtt{True}$ iff v is in the Tutte set. For each edge $\{(u,a),(v,b)\} \in E$, we use a variable $\mathtt{e}_{uv}^{ab}$ to represent whether this edge remains in the induced subgraph $G[V \setminus S]_c$ w.r.t. a vertex coloring c. Then we have:

$$\mathtt{RemainingEdges}$$
$$=\{\mathtt{e}_{uv}^{ab} \leftrightarrow (\neg T_v \wedge \neg T_u \wedge \mathtt{vc}_u^a \wedge \mathtt{vc}_v^b)\}_{\{(u,a),(v,b)\}\in E}.$$

One of the technical challenges of encoding Tutte's Theorem is how to count odd connected components of $G[V \setminus S]_c$. For each vertex v and an index i of a connected component, we introduce a variable $\mathtt{cc}_v^i$, which is $\mathtt{True}$ iff vertex v is in the i-th connected component. Although there can be up to $|V|$ non-empty connected components in $G[V \setminus S]_c$ (consider $|V|$ isolated vertices), multiple connected components are often empty in practice. For two vertices u and v, if they are connected by an edge that remains in $G[V \setminus S]_c$, then they must be in the same connected component, represented by the following constraints:

$$\mathtt{ConnectedComponent}$$
$$=\{\mathtt{e}_{uv}^{ab} \rightarrow \bigwedge_{1\le i\le |V|} (\mathtt{cc}_v^i \leftrightarrow \mathtt{cc}_u^i)\}_{\{(u,a),(v,b)\}\in E}.$$

It is worth pointing out that $\mathtt{ConnectedComponent}$ does not enforce different components to be indexed differently. Nevertheless, $\mathtt{TutteCondition}$ below is in favor of using as many indices of connected components as possible.

For each vertex v that is not in the Tutte set, we require v to be in exactly one connected component. If v is in the Tutte set, then it does not belong to any of the connected components, represented by:

$$\mathtt{ValidComponent}$$
$$=\{\mathtt{ExactOne}(\{\mathtt{cc}_v^i\}_{1\le i\le |V|} \cup \{T_v\})\}_{v\in V}$$

For each index i ($1 \le i \le |V|$) of connected components, we use a variable $\mathtt{Odd}_i$ to indicate whether the connected component with index i has an odd number of vertices. The set of variables $\{\mathtt{Odd}\}_i$ complies with the following set of XOR constraints:

$$\mathtt{Odd} = \{\mathtt{Odd}_i \leftrightarrow \bigoplus_{v\in V} \mathtt{cc}_v^i\}_{1\le i\le |V|}.$$

Finally, we enforce that the number of odd connected components in $G[V \setminus S]_c$ is greater than the size of the Tutte set by a cardinality constraint:

$$\mathtt{TutteCondition} = \{\sum_{i=1}^{|V|} \mathtt{Odd}_i > \sum_{v\in V} T_v\}.$$

Let $\mathtt{TutteEncoding}$ be the union of constraints in $\mathtt{RemainingEdges}$, $\mathtt{ConnectedComponent}$, $\mathtt{ValidComponent}$, $\mathtt{Odd}$ and $\mathtt{TutteCondition}$, as well as $\mathtt{ValidColoring}$ and $\mathtt{QuantumState}$ defined previously. Then we have the following proposition, whose proof is delayed to the appendix.

Proposition 3. (Correctness of Tutte encoding) *$\mathtt{TutteEncoding}$ is satisfiable iff the bicolored graph does not satisfy the FORALL-PMVC condition.*

Our Tutte encoding uses $|V|^2 + |E| + (d+2) \cdot |V|$ variables.

Remark 1. *The Tutte encoding can be alternatively written in ILP, which allows the Boolean vectors for expressing colorings and indices of connected components to be replaced by a single integer variable. Meanwhile, the constraints for* `ValidComponent` *and* `ValidColoring` *are no longer necessary in the ILP encoding. The evaluation of Tutte encoding on ILP solvers is left as future work.*

5.3 Optimizations of the Tutte Encoding

The Tutte encoding we proposed above can be improved, by exploiting the symmetric nature of Tutte set and the input graph. In the following, we describe optimization techniques that reduce the search space by 1) using fewer variables and 2) eliminating redundant solutions. As will be seen in the experiment section, the optimized Tutte encoding scales exponentially better than the vanilla one proposed above.

Exploiting the Symmetry in Tutte's Theorem

Each vertex in $V \setminus S$ belongs to a connected component. Given a solution of the Tutte encoding, permuting the indices of connected components also generates a solution. To reduce the solution space, we can leave each vertex with fewer options regarding the index of the connected component. Specifically, for a vertex v with $1 \leq v \leq |V|$, we only keep variables $cc_v^1, \cdots, cc_v^v$. That is, a vertex v must belong to a connected component whose index is smaller or equal to v. $\frac{|V|(|V|-1)}{2}$ variables are removed from the vanilla Tutte encoding.

Additionally, we add constraints to reduce the number of solutions. In spite of that $G[V \setminus S]_c$ can have up to $|V|$ connected components, the number of non-empty connected components is often much smaller than $|V|$. Therefore, for the same Tutte set, there is still considerable flexibility in the assignment of connected component indices, which we aim to further reduce. If a vertex v is in the Tutte set or belongs to a connected component with an index smaller than v, we can rule out all solutions that contain non-empty connected components indexed by v, expressed by the following constraints:

$$\begin{aligned} \mathtt{OPT} = & \{T_v \rightarrow \big(\bigwedge_{v \leq u \leq |V|} \neg cc_u^v \big)\}_{v \in V} \\ & \cup \{\big(\bigvee_{1 \leq i < v} cc_v^i \big) \rightarrow \big(\bigwedge_{v \leq u \leq |V|} \neg cc_u^v \big)\}_{v \in V}. \end{aligned}$$

Exploiting the Symmetry of the Input Graph

One of the advantages of constraint-based approaches is that the properties of the input can be easily taken into account as extra constraints. Previous work has been done in symmetry-breaking by leveraging graph automorphism when solving combinatorial problems by propositional logic [Aloul *et al.*, 2006]. Since adding all potential symmetry-breaking constraints often leads to an exponential overhead, in this work we apply a lightweight symmetry-breaking strategy. If there exists a set $U \subseteq V$, such that all vertices are equivalent in graph G, i.e., permuting vertices in U leads to a graph isomorphism to G. Then we want the solution of the Tutte encoding to have the smallest lexicographically representative in U.

For such a set U and an order of vertices in U: $(u_1, \cdots, u_{|U|})$, we add the following constraints:

`GraphSymmetryBreaking` $= \{T_{u_i} \rightarrow T_{u_{i-1}}\}_{2 \leq i \leq |U|}.$

Note that our approach is not the only way of breaking symmetry. There exists a trade-off between the size of constraints and solution-space reduction [Walsh, 2006]. We leave investigating the interaction of different symmetry-breaking techniques on FORALL-PMVC as a future work.

6 Experiment Results

This section evaluates our Tutte encoding on solving FORALL-PMVC problems.

6.1 Implementations and Methods in Comparison

RQ1. How do the optimization techniques in Section 5.3 improve the performance of the Tutte encoding?

RQ2. Among conjunctive normal form (CNF), pseudo-Boolean+XOR (PBXOR), and Answer Set Programming (ASP), which language fits best for the Tutte encoding?

RQ3. How do methods based on Tutte encoding perform on FORALL-PMVC benchmarks?

6.2 Implementations and Competing Methods

We implemented the Tutte encoding in CNF formulas (TutteCNF), pseudo-Boolean+XOR constraints (TuttePBXOR), and Answer Set Programming (TutteASP). Additionally, we implemented the QBF-based approach in Section 5.1 and Algorithm 1 based on the Blossom Algorithm. Our implementations are based on the following software.

- For solving ASP programs, we use Clingo [Gebser *et al.*, 2019]. Due to the powerful expressiveness of ASP, the implementation of Tutte encoding is less than 30 lines of code, which is considerably more compact than CNF and pseudo-Boolean encoding.

- For solving CNF Tutte encoding, we use two SAT solvers: Kissat [Fleury and Heisinger, 2020], a winner of recent SAT Competitions, and Clasp [Gebser *et al.*, 2007], the underneath SAT solver in Clingo in order for fairly comparing CNF with ASP.

- For solving pseudo-Boolean+XOR encoding, we use LinPB [Yang and Meel, 2021], a recent advanced hybrid Boolean constraint solver with native XOR support.

- We implemented Alg. 1 (Enum-Blossom) with Python package `networkx` [Hagberg and Conway, 2020].

- For solving QBF formulas, we use DepQBF, a medal winner of QBF evaluations [Lonsing and Biere, 2010].

Besides the vanilla Tutte encoding in Section 5.2, we use Tutte[Language]+Opt to denote the Tutte encoding including constraints in `OPT` defined in Section 5.3 and with unnecessary variables removed. We use Tutte[Language]+Opt+GS to denote the Tutte encoding including constraints in both `OPT` and `GraphSymmetryBreaking`.

All experiments were run on single CPU cores of a Linux cluster at 2.60-GHz and with 16 GB of RAM. We set the time

Methods	Running Time/s
Blossom	0.02
ExactOne-PB	0.11
TutteCNF+Opt+GS	10.61
TutteCNF+Opt	12.85
TutteASP+Opt+GP	14.89
TutteASP+Opt	66.40
TutteCNF	133.73
ExactOne-CNF	Timeout (> 1000s)

Table 1: Running time for proving the uncolored complete bipartite graph $K_{48,50}$ has no perfect matching.

limit for each problem instance to 1000 seconds. The implementations and benchmarks can be found in the GitHub repository: https://github.com/zzwonder/PMVC.

Experiment 1: Proving the Non-existence of PM of Uncolored Graphs. We first evaluate the efficiency of Tutte encoding in identifying the non-existence of PM of uncolored graphs, before solving FORALL-PMVC problems. The benchmarks include complete bipartite graphs $K_{n-2,n}$ for $n \in \{10, 12, 14, \cdots, 70\}$. All graphs have no perfect matching. The first partition with $n - 2$ vertices forms the unique Tutte set. We compare the Tutte encoding with the `Exact-One` encoding in Section 5.1, whose unsatisfiability indicates the non-existence of perfect matching. We also include the Blossom Algorithm in comparison. The running time for each algorithm to prove $K_{48,50}$ has no perfect matching is shown in Table 1 and the comprehensive results are delayed to the appendix. Each running time is the median of 100 independent executions.

Experiment 2: Verifying graphs that satisfy the Dicke FORALL-PMVC conditions. In this experiment, we use different techniques to verify graphs that satisfy the Dicke state. We generate two types of graphs. First, for $n \in \{6, 8, \cdots, 40\}$ we generate graphs that satisfy $\text{Dicke}(n, n/2)$ (see Section 3 and Figure 2), where the size of legal coloring grows exponentially as n increases. Second, we fix $n = 36$ and generate graphs that satisfy $\text{Dicke}(36, k)$ with $k \in \{1, 2, \cdots, 18\}$. The Tutte-encoding-based approaches are supposed to output "unsatisfiable", as the formula translates to finding a legal coloring with a Tutte set. The results are shown in Figure 4 and 5. Each running time is the median of 100 independent executions.

Experiment 3: Refuting graphs that violate the Dicke FORALL-PMVC conditions. We evaluate the searching power of different approaches via refuting graphs that violates the Dicke state. Tutte-encoding-based approaches are supposed to output "satisfiable". Graphs in this benchmark are obtained by removing edges from graphs that satisfy the Dicke state. For each (n, k) pair where $n \in \{10, 15, 20, \cdots, 80\}$, and $k \in \{0.1 \cdot n, 0.2 \cdot n, 0.3 \cdot n, 0.4 \cdot n\}$, we first generate 20 graphs that satisfy $\text{Dicke}(n, k)$. Then for each graph, we randomly remove either 40% of blue edges or 2 bicolored edges such that the resulting graph violates the Dicke state. In Enum-Blossom, we shuffled the list of legal colorings for more stable performance. The number of
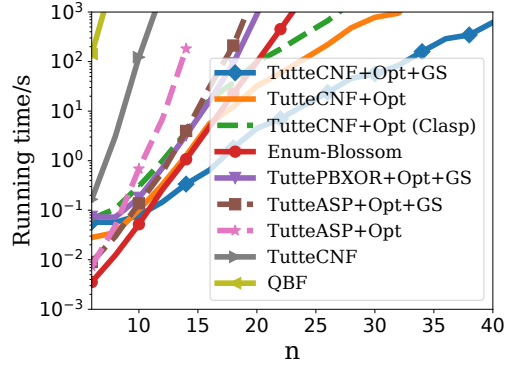

Figure 4: Running time for verifying the FORALL-PMVC condition for graphs that satisfy $\text{Dicke}(n, n/2)$. The unsatisfactory performance of TutteASP and TuttePBXOR (without +Opt, +GS) is omitted for readability.
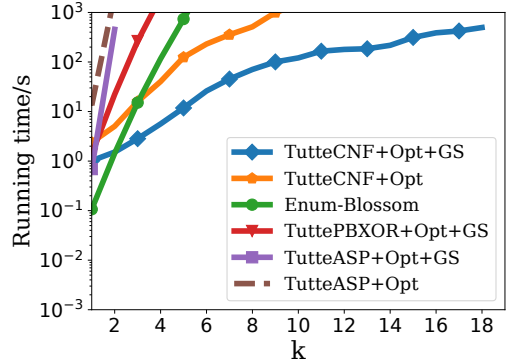

Figure 5: Running time for verifying FORALL-PMVC condition for graphs that satisfy $\text{Dicke}(36, k)$. QBF failed to solve a single instance even for $k = 1$ within the time limit.

instances that each method solved v.s. the running time is plotted in Figure 6.

6.3 Analysis of Results

Answer to RQ1
A significant performance enhancement by optimization techniques in Section 5.3 can be observed in Table 1 and Figure 4-6. For example, verifying $\text{Dicke}(10, 5)$ was accelerated by 1200 times from TutteCNF (122.7s) to TutteCNF+Opt (0.1s). On searching tasks (Figure 6), TutteCNF+Opt (741) solved 466 more instances than TutteCNF (275). Adding constraints for graph symmetry exploitation also yields great improvement for the verification of relatively large symmetric graphs. TutteCNF+Opt+GS successfully verifies all Dicke graphs in Experiment 2 within 1000s, while TutteCNF+Opt timed out for verifying $\text{Dicke}(36, 10)$ and $\text{Dicke}(34, 17)$. We conclude that our optimization techniques are critical for the performance of Tutte encoding.

Answer to RQ2
All experiments agree that CNF is the language leading to the best performance for Tutte encoding, with a signif-
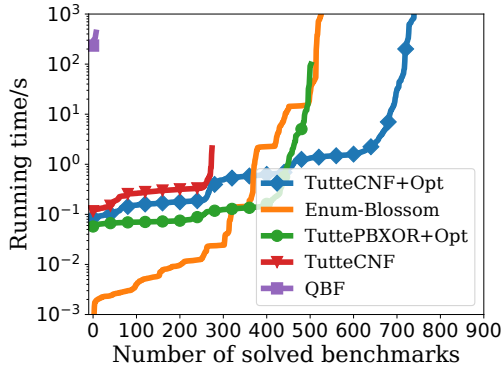
Figure 6: The number of benchmarks solved by each method v.s. running time (1280 instances in total).

icant advantage against PBXOR and ASP. The ASP solver Clingo is deployed with a relatively slow SAT solver Clasp with specialized adaptions, which makes it difficult to replace Clasp with a faster SAT solver, e.g., Kissat. To compare CNF encoding with ASP fairly, in Experiment 2 we also solved the Tutte encoding by Clasp. In Figure 4, TutteCNF+Opt (Clasp) still outperforms TutteASP+Opt+GS and TuttePBXOR+Opt+GS by far. We conclude that while high-level languages such as ASP and PBXOR can be convenient for modeling the problem, low-level languages such as CNF are necessary for better performance.

Answer to RQ3

Experiment 1 (Table 1). When proving the non-existence of perfect matching in uncolored graphs $K_{n-2,n}$, the polynomial Blossom Algorithm is the fastest approach as expected. Among all encoding-based approaches, `Exact-One` encoding with pseudo-Boolean solver (ExactOne-PB) scales best while ExactOne-CNF is the slowest. This observation aligns with the theoretical results about 1) the polynomial behavior of cutting-plane methods [Chandrasekaran *et al.*, 2016] and 2) the exponential lower bound of resolution-based approaches on perfect matching [Razborov, 2004]. Compared with the Blossom Algorithm and ExactOnePB, approaches based on Tutte encoding take a longer time to solve the problem by searching for the unique Tutte set.

Experiment 2 (Figure 4, 5). For the verification of Dicke FORALL-PMVC conditions, all approaches tend to scale exponentially. The QBF encoding presents the worst scalability, which failed to verify `Dicke(10, 5)` and `Dicke(36, 1)` within the time limit. Methods based on Vanilla Tutte encoding (TutteCNF) also do not scale well, which only outperform QBF. With optimization techniques, Tutte encodings in ASP and PBXOR languages (TuttePBXOR+Opt+GS, TutteASP+Opt+GS) were able to achieve better scalability, although the Enum-Blossom algorithm still performs better. Only when equipped with both optimization techniques and state-of-the-art SAT solvers (TutteCNF+Opt, TutteCNF+Opt+GS), does the Tutte encoding display promising potential. For verifying graphs that satisfy `Dicke(n, n/2)`, TutteCNF+Opt+GS can successfully verify graphs with $n = 40$ and TutteCNF+Opt is able to solve for $n = 34$ within

1000s, both of which scale exponentially better than Enum-Blossom ($n = 26$) and QBF ($n = 8$) according to Figure 4. As for verifying `Dicke(36, k)` (Figure 5), when k is small ($k \leq 2$), the set of legal coloring is polynomially bounded and Enum-Blossom is the fastest algorithm. As k increases, optimized CNF Tutte encoding with SAT solvers dominates other approaches by far.

Experiment 3 (Figure 6). On identifying graphs that violate FORALL-PMVC conditions, QBF only solved 8 instances (1280 in total). TuttePBXOR+Opt and Enum-Blossom solved 503 and 529 instances respectively. Enum-Blossom is able to solve relatively small-size instances quickly (within 0.1s). The optimized Tutte encoding in CNF (TutteCNF+Opt) solved 741 instances, which outperforms Enum-Blossom mainly on large-size problems.

Summary. Not surprisingly, the Blossom Algorithm dominates constraint-based approaches in proving the non-existence of PM for uncolored graphs. However, when solving FORALL-PMVC tasks with vertex coloring involved, our method displays promising performance, especially when the legal coloring set grows exponentially. The vanilla Tutte encoding does not scale well without optimization techniques and suitable language. With modern CNF-SAT solvers, optimized Tutte encoding scales exponentially better than QBF and a graph theoretical algorithm (Enum-Blossom) on FORALL-PMVC problems. We conclude that while hybrid-constraint-based techniques are powerful and expressive, extra effort is necessary for solving FORALL-PMVC efficiently. Our approaches also scale significantly better than previous implementations for similar tasks from the quantum-computing community. The graph instances used in previous work have less than 15 vertices [Cervera-Lierta *et al.*, 2021].

7 Conclusions and Future Directions

In this paper, we study a new application of hybrid Boolean constraints, named FORALL-PMVC, which arises in quantum computing. We propose a novel encoding based on Tutte's Theorem in graph theory as well as optimization techniques for this encoding. Empirical results demonstrate that in the most suitable encoding language (CNF), our approach based on optimized Tutte encoding scales significantly better than competing approaches on tasks with interest in quantum computing. Our study identifies the necessity of designing problem-specific encodings when applying powerful general-purpose constraint solvers. Our implementations also provide useful tools for the quantum computing community.

For future directions, we plan to apply constraint-based methods to solve problems beyond graph property checking. For example, the ultimate goal of this line of research [Cervera-Lierta *et al.*, 2021] is to *construct* a graph w.r.t. a quantum state. Considering an alternative constraint-programming approach for our problem based on filtering algorithms and `All-Different` constraints [Bessiere *et al.*, 2010] is another interesting direction. It is also promising to apply our Tutte encoding for solving graph theoretical problems related to Tutte's Theorem algorithmically [Bauer *et al.*, 2007].

Acknowledgements

Contribution Statement

The author list has been sorted alphabetically by last name. Zhiwei Zhang is the corresponding author.

References

[Aloul *et al.*, 2006] Fadi A Aloul, Karem A Sakallah, and Igor L Markov. Efficient symmetry breaking for boolean satisfiability. *IEEE Transactions on Computers*, 55(5):549–558, 2006.

[Anderson, 1971] Ian Anderson. Perfect matchings of a graph. *Journal of Combinatorial Theory, Series B*, 10(3):183–186, 1971.

[Ballentine, 2014] Leslie E Ballentine. *Quantum mechanics: a modern development*. World Scientific Publishing Company, 2014.

[Bauer *et al.*, 2007] Douglas Bauer, Haitze J Broersma, Aurora Morgana, and E Schmeichel. Tutte sets in graphs i: Maximal tutte sets and d-graphs. *Journal of graph theory*, 55(4):343–358, 2007.

[Bessiere *et al.*, 2010] Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Propagating conjunctions of alldifferent constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 27–32, 2010.

[Biere *et al.*, 2009] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[Bogdanov *et al.*, 2011] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. In *ASIACRYPT 2011*, 2011.

[Brailsford *et al.*, 1999] Sally C Brailsford, Chris N Potts, and Barbara M Smith. Constraint satisfaction problems: Algorithms and applications. *European journal of operational research*, 119(3):557–581, 1999.

[Cervera-Lierta *et al.*, 2021] Alba Cervera-Lierta, Mario Krenn, and Alán Aspuru-Guzik. Design of quantum optical experiments with logic artificial intelligence. *arXiv preprint arXiv:2109.13273*, 2021.

[Chandrasekaran *et al.*, 2016] Karthekeyan Chandrasekaran, László A Végh, and Santosh S Vempala. The cutting plane method is polynomial for perfect matchings. *Mathematics of Operations Research*, 41(1):23–48, 2016.

[Costa *et al.*, 2009] M.-C. Costa, D. de Werra, C. Picouleau, and B. Ries. Graph Coloring with Cardinality Constraints on the Neighborhoods. *Discrete Optimization*, 6(4):362 – 369, 2009.

[Devriendt *et al.*, 2021] Jo Devriendt, Stephan Gocht, Emir Demirovic, Jakob Nordström, and Peter J Stuckey. Cutting to the core of pseudo-boolean optimization: Combining core-guided search with cutting planes reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3750–3758, 2021.

[Dinur *et al.*, 2005] Irit Dinur, Oded Regev, and Clifford Smyth. The Hardness of 3-Uniform Hypergraph Coloring. *Combinatorica*, 25(5):519–535, Sep 2005.

[Duncan *et al.*, 2020] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John Van De Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus. *Quantum*, 4:279, 2020.

[Edmonds, 1965] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

[Eén and Sörensson, 2003] Niklas Eén and Niklas Sörensson. An Extensible SAT-Solver. In *SAT*, pages 502–518, 2003.

[Feldman *et al.*, 2005] J. Feldman, M. J. Wainwright, and D. R. Karger. Using Linear Programming to Decode Binary Linear Codes. *IEEE Transactions on Information Theory*, 51(3):954–972, March 2005.

[Fleury and Heisinger, 2020] Armin Biere Katalin Fazekas Mathias Fleury and Maximilian Heisinger. Cadical, kissat, paracooba, plingeling and treengeling entering the sat competition 2020. *SAT COMPETITION*, 50:2020, 2020.

[Froleyks *et al.*, 2021] Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda. Sat competition 2020. *Artificial Intelligence*, 301:103572, 2021.

[Gaspers *et al.*, 2012] Serge Gaspers, Eun Jung Kim, Sebastian Ordyniak, Saket Saurabh, and Stefan Szeider. Don't be strict in local search! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 486–492, 2012.

[Gebser *et al.*, 2007] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 260–265. Springer, 2007.

[Gebser *et al.*, 2019] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot asp solving with clingo. *Theory and Practice of Logic Programming*, 19(1):27–82, 2019.

[Gong and Zhou, 2017] Weiwei Gong and Xu Zhou. A Survey of SAT Solver. *AIP Conference Proceedings*, 1836(1):020059, 2017.

[Gu *et al.*, 2019a] Xuemei Gu, Lijun Chen, Anton Zeilinger, and Mario Krenn. Quantum experiments and graphs. iii. high-dimensional and multiparticle entanglement. *Physical Review A*, 99(3):032338, 2019.

[Gu *et al.*, 2019b] Xuemei Gu, Manuel Erhard, Anton Zeilinger, and Mario Krenn. Quantum experiments and graphs ii: Quantum interference, computation, and state

generation. *Proceedings of the National Academy of Sciences*, 116(10):4147–4155, 2019.

[Hagberg and Conway, 2020] Aric Hagberg and Drew Conway. Networkx: Network analysis with python. *URL: https://networkx. github. io*, 2020.

[Jooya *et al.*, 2016] Hossein Z Jooya, Kamran Reihani, and Shih-I Chu. A graph-theoretical representation of multiphoton resonance processes in superconducting quantum circuits. *Scientific Reports*, 6(1):1–10, 2016.

[Krenn *et al.*, 2017] Mario Krenn, Xuemei Gu, and Anton Zeilinger. Quantum experiments and graphs: Multiparty states as coherent superpositions of perfect matchings. *Physical review letters*, 119(24):240403, 2017.

[Krenn *et al.*, 2019] Mario Krenn, Xuemei Gu, and Daniel Soltész. Questions on the structure of perfect matchings inspired by quantum physics. *arXiv preprint arXiv:1902.06023*, 2019.

[Krenn *et al.*, 2022] Mario Krenn, Jonas Landgraf, Thomas Foesel, and Florian Marquardt. Artificial intelligence and machine learning for quantum technologies. *arXiv preprint arXiv:2208.03836*, 2022.

[Kyrillidis *et al.*, 2020] Anastasios Kyrillidis, Anshumali Shrivastava, Moshe Vardi, and Zhiwei Zhang. Fouriersat: A fourier expansion-based algebraic framework for solving hybrid boolean constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1552–1560, 2020.

[Kyrillidis *et al.*, 2021a] Anastasios Kyrillidis, Anshumali Shrivastava, Moshe Y Vardi, and Zhiwei Zhang. Solving hybrid boolean constraints in continuous space via multilinear fourier expansions. *Artificial Intelligence*, 299:103559, 2021.

[Kyrillidis *et al.*, 2021b] Anastasios Kyrillidis, Moshe Vardi, and Zhiwei Zhang. On continuous local bdd-based search for hybrid sat solving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3841–3850, 2021.

[Kyrillidis *et al.*, 2022] Anastasios Kyrillidis, Moshe Y Vardi, and Zhiwei Zhang. Dpms: an add-based symbolic approach for generalized maxsat solving. *arXiv preprint arXiv:2205.03747*, 2022.

[Liang, 2018] Jia Hui Liang. *Machine Learning for SAT Solvers*. PhD thesis, University of Waterloo, December 2018.

[Lifschitz, 2019] Vladimir Lifschitz. *Answer set programming*. Springer Heidelberg, 2019.

[Lonsing and Biere, 2010] Florian Lonsing and Armin Biere. Depqbf: A dependency-aware qbf solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(2-3):71–76, 2010.

[Martins *et al.*, 2014] Ruben Martins, Vasco Manquinho, and Inês Lynce. Open-wbo: A modular maxsat solver,. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing – SAT 2014*, 2014.

[Pan *et al.*, 2012] Jian-Wei Pan, Zeng-Bing Chen, Chao-Yang Lu, Harald Weinfurter, Anton Zeilinger, and Marek Żukowski. Multiphoton entanglement and interferometry. *Reviews of Modern Physics*, 84(2):777, 2012.

[Prestwich, 2009] Steven D Prestwich. Cnf encodings. *Handbook of satisfiability*, 185:75–97, 2009.

[Razborov, 2004] Alexander A Razborov. Resolution lower bounds for perfect matching principles. *Journal of Computer and System Sciences*, 69(1):3–27, 2004.

[Schrijver and others, 2003] Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

[Soos *et al.*, 2009] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT Solvers to Cryptographic Problems. In *SAT*, pages 244–257, 2009.

[Tseitin, 1983] Grigori S Tseitin. On the complexity of derivation in propositional calculus. In *Automation of reasoning*, pages 466–483. Springer, 1983.

[Vardi and Zhang, 2022] Moshe Y Vardi and Zhiwei Zhang. Quantum-inspired perfect matching under vertex-color constraints. *arXiv preprint arXiv:2209.13063*, 2022.

[Vardi, 2014] Moshe Y. Vardi. Boolean satisfiability: theory and engineering. *Commun. ACM*, 57(3):5, 2014.

[Walsh, 2006] Toby Walsh. General symmetry breaking constraints. In *Principles and Practice of Constraint Programming-CP 2006: 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006. Proceedings 12*, pages 650–664. Springer, 2006.

[Yang and Meel, 2021] Jiong Yang and Kuldeep S Meel. Engineering an efficient pb-xor solver. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.