Encoding Constraints as Binary Constraint Networks Satisfying BTP

Ruiwei Wang
School of Computing, National University of Singapore
wangruwei@u.nus.edu

Abstract

Recently, the Binary Constraint Tree (BCT), a tree structured Binary Constraint Network (BCN), has been shown to be more succinct than various ad-hoc constraints. In this paper, we investigate the modelling power of a well-known tractable hybrid class generalizing BCT, i.e., the class of BCNs satisfying Broken Triangle Property (BTP) called BTP Networks (BTPNs). We show that the consistency checker of BTPN can be computed by polysize monotone circuit, thus, some global constraints cannot be encoded as polysize BTPN, such as the AllDifferent and Linear constraints. Then our study reveals that BTPN is strictly more succinct than the DNNF constraint and all 14 ad-hoc constraints discussed in (Wang and Yap 2023), such as the context-free grammar, BCT and smart table constraints. Furthermore, we also show that BTPN is as powerful as DNNF in terms of computing various operations and queries. In addition, we prove that it is NP-hard to determine the minimum sized BTPN encoding a constraint.

1 Introduction

Many tractable classes (Carbonnel and Cooper 2016) of Constraint Networks (CNs) have been proposed to study the tractability of Constraint Satisfaction Problems (CSPs). However, there is less work on encoding constraints with the tractable classes. Recently, (Wang and Yap 2022b, 2023) showed that Binary Constraint Tree (BCT) can be used to model various ad-hoc constraints in polysize, where BCT is the class of tree structured Binary CNs (BCNs) (Freuder 1982). In this paper, we investigate the class of BCNs satisfying Broken Triangle Property (BTP) called BTP Networks (BTPNs), which is a well-known tractable hybrid class generalizing BCT (Cooper, Jeavons, and Salamon 2010).

(Dechter 1990) showed that the expressiveness of BCNs can be significantly improved with hidden variables. In addition, various binary encodings with hidden variables have been proposed to encode table constraints (Yap, Xia, and Wang 2020) as BCNs, such as the dual encoding (Dechter and Pearl 1989), hidden variable encoding (Rossi, Petrie, and Dhar 1990), double encoding (Stergiou and Walsh 1999) and bipartite encoding (Wang and Yap 2020). Although all constraint relations can be represented as table constraints, the tabulation of constraints may cause exponential blowup. For example, the Multi-valued Decision Diagram (MDD) can be exponentially smaller than its table representation. As such (Wang and Yap 2022b) proposed binary encodings to encode MDD constraints as BCTs. They showed that BCT is strictly more succinct than many ad-hoc constraints (Wang and Yap 2022a, 2023). Then a question that arises is can we define more succinct constraint representations with other tractable classes?

Usually, the tractable classes of CNs can be classified as language classes, structural classes and hybrid classes (Carbonnel and Cooper 2016). Language classes restrict the language of constraint relations. For example, the largest tractable language class restricts that all constraint relations are preserved by a weak near-unanimity (WNU) operation (Bulatov 2017; Zhuk 2017). Then structural classes restrict the structure of constraint graphs, such as bounded fractional hypertree width (Grohe and Marx 2014). Hybrid classes simultaneously consider both constraint relations and graphs, such as the classes defined by forbidden patterns (Cohen et al. 2012; Cooper and Živný 2016). Different from most previous works, which are mainly about the time complexity of CSPs and focus on the CSP dichotomy conjecture (Feder and Vardi 1998), we will investigate the tractable classes from a constraint modelling perspective.

In the paper, we first show that it is unlikely to improve the succinctness of BCT by directly using known tractable structural and language classes. Then we investigate the modelling power of the class of BTPNs. To be specific, we prove that the consistency checker (Bessiere et al. 2009) of the BTPN constraint can be computed by polysize monotone circuit, thus, some global constraints cannot be encoded as polysize BTPN, such as Circuit (Beldiceanu and Conette 1994), Channelling (Cheng et al. 1999) and Linear (Yuanlin and Yap 2000) constraints. Meanwhile, we introduce a binary encoding to transform the DNNF constraint (Gange and Stuckey 2012) into polysize BTPN, and show that BTPN is strictly more succinct than the DNNF constraint and all 14 ad-hoc constraints discussed in (Wang and Yap 2023), such as the context-free grammar (Quimper and Walsh 2006), BCT and smart table (Mairy, Deville, and Lecoutre 2015) constraints. Moreover, we prove that it is NP-hard to minimizing the size of the BTPN encoding a constraint. In addition, we also show that BTPN is as powerful as DNNF in terms of computing operations and queries.
2 Preliminaries

A Constraint Network (CN) is a pair \((X, C)\) where \(X\) is a set of variables, \(D(x)\) is the domain of a variable \(x \in X\), and \(C\) is a set of constraints. A literal of \(x\) is a pair \((x, a)\). A tuple over variables \(\{x_1, \ldots, x_n\}\) is a set of literals \(\{(x_1, a_1), \ldots, (x_n, a_n)\}\). Each constraint \(c\) has a scope \(scp(c) \subseteq X\) and a relation \(rel(c)\), where \(rel(c)\) is a set of tuples over \(scp(c)\). Then \(|scp(c)|\) is the arity of \(c\), and \(c\) is a binary constraint if \(|scp(c)| = 2\). A CN \(P\) is a Binary CN (BCN) if the arity of all constraints in \(P\) is at most 2. A BCN is normalized if all binary constraints have different scopes.

In this paper, we only consider normalized BCNs.

Given any set \(V\) of variables and a set \(\tau\) of literals, we use \(\tau[V]\) to denote a subset of \(\tau\), and \(T[\tau]\) is the projection of tuples \(T\) in \(\tau\). Then for any CN \(P = (X, C)\) and literals \(L, P[L]\) is a sub-problem \((X, C \cup \{x\} | x \in X\)) of \(P\), where \(\{x\}\) is a constraint over \(\{x\}\) with \(rel(c) = \{(x, a)\} | (x, a) \in L\).

A tuple \(\tau\) over variables \(V\) is consistent on a CN \(P = (X, C)\) if for all \(c \in C\) such that \(scp(c) \subseteq V\), \(\tau[scp(c)] \subseteq rel(c)\). An assignment \(\tau\) is an assignment if for all \((x, a) \in \tau\), \(a \in D(x)\). A literal \((x, a)\) is valid on \(P\) if \((x, a) \in \tau\) and \((x, a) \in D(x)\). A tuple \(\tau\) is consistent on \(P\) if all literals in \(\tau\) are valid on \(P\). In addition, an assignment \(\tau\) over \(X\) is a solution of \(P\) if \(\tau\) is consistent on \(P\) and \(sol(X, C)\) or \(sol(P)\) denotes the solutions of \(P\). A CN \(P\) is satisfiable if \(sol(P) \neq \emptyset\). A Constraint Satisfaction Problem (CSP) is to check whether a CN is satisfiable.

A literal \((x, a)\) is a Generalized Arc Consistency (GAC) on a constraint \(c\) if \(rel(c)\) has a valid tuple \(\tau\) including \((x, a)\). A variable \(x\) is GAC on \(c\) if all valid literals of \(x\) are GAC on \(c\). Then \(c\) is GAC if all variables in \(scp(c)\) are GAC on \(c\). A CN is GAC if all constraints are GAC. Enforcing GAC on a CN \(P\) on \(P\) is to find the maximum set \(L\) of valid literals such that \(P[L]\) is GAC, and replacing \(P\) with \(P[L]\). Additionally, GAC on BCNs is also called Arc Consistency (AC). A BCN \(P\) is solvable by AC means \(P\) is satisfiable if and only if all variables have valid literals after enforcing AC on \(P\).

A CN \(P = (X, C)\) encodes a constraint \(c\) if \(scp(c) \subseteq X\) and \(sol(P)[scp(c)] = rel(c)\), where the variables in \(scp(c)\) and \(X \setminus scp(c)\) are called original variables and hidden variables. The size of a BCN is the sum of its variable domain sizes and constraint sizes, where the size of a binary (unary) constraint \(c\) is defined as \(|rel(c)|\). A class \(S\) of BCNs can encode a constraint \(c\) in size if \(S\) has a BCN encoding \(c\) whose size is polynomial in the size of \(c\). Then \(S\) is conservative if for any \(P \in S\) and literals \(L, P[L]\) is in \(S\).

3 Language and Structural Classes

In this section, we discuss why it is unlikely to improve the modelling power of the BCT constraint by directly using existing tractable language and structural classes.

Language Classes

It has been proved that language classes are either NP-complete or \(P\), and the CNs in a language class are tractable if and only if their constraint relations are preserved by a WNU operation (Bulatov 2017; Zhuk 2017). Then as shown in (Jeavons, Cohen, and Gyssens 1997), the conjunction and projection operations on the constraint relations preserved by a WNU operation can only be used to encode the constraint relations preserved by the WNU operation. Therefore, tractable language classes cannot be directly used to encode arbitrary constraint relations.

Structural Classes

Unlike language classes, there are structural classes between NP-complete and \(P\) (Bodirsky and Grohe 2008). To the best of our knowledge, bounded treewidth is the largest tractable structural class of BCNs (Grohe 2007), and bounded fractional hypertree width (FHW) is the most general known tractable structural class of CNs (Grohe and Marx 2014).

For any tree decomposition \(T\) of a CN \(P = (X, C)\) and a bag \(B \subseteq X\) and the size of \(sol(P)[B]\) is bounded by a polynomial whose degree is the FHW of \(T\) (Grohe and Marx 2014). Then we can construct a CN \(P_T = (X, C_T)\) such that \(sol(P) = sol(P_T)\) and \(C_T = \{c_B | B \in T\} \) where \(scp(c_B) = B\) and \(rel(c_B) = sol(P)[B]\). (Dechter and Pearl 1989; Wang and Yap 2023; Kucera 2023) showed that \(P_T\) can be encoded as polysize BCT, thus, we cannot directly define a strictly more succinct constraint representation with existing tractable structural classes.

4 Broken Triangle Property Networks

As discussed in Section 3, it is unlikely to directly improve the succinctness of BCNs with existing tractable language and structural classes. In this section, we introduce a well known hybrid tractable constraint generalizing BCT, i.e. the class of BCNs satisfying Broken Triangle Property (BTP).

Definition 1. A Broken Triangle (BT) on a BCN \(P\) is a set of 4 valid literals \(\{(x, a_1), (y, a_2), (z, a_3), (z, a_4)\}\) such that the 3 tuples \(\{(x, a_1), (y, a_2)\}, \{(z, a_3), (z, a_4)\}\) are consistent on \(P\) and the 2 tuples \(\{(x, a_1), (z, a_4)\}, \{(y, a_2), (z, a_3)\}\) are not consistent on \(P\).

Figure 1a gives a BCN \(P\) encoding the constraint \(x \neq y \wedge x \neq z \wedge y \neq z\). The figure has 3 BTs where each BT consists of 2 dashed lines. For example, a BT is the set of literals \(\{(x, 1), (z, 2), (y, 2), (y, 1)\}\), where the 2 tuples \(\{(x, 1),\)}
(y, 1), \{(y, 2), (y, 3)\} are not consistent on P (corresponding to 2 dashed lines), and the 3 tuples \{\{(x, 1), (z, 2)\}, \{(x, 2), (y, 1)\}\} are consistent on P.

**Definition 2.** A BCN \(P = (X, C)\) satisfies the Broken Triangle Property (BTP) w.r.t. an ordering \(O\) over \(X\) if there is not any BT \(\{(O_i, a_i), (O_j, a_j), (O_k, a_k), (O_\ell, a_\ell)\}\) on \(P\) such that \(i < j < k\), where \(O_i, O_j, O_k\) denote the \(i^{th}, j^{th}, k^{th}\) variables in \(O\). Then \(P\) satisfies BTP if it satisfies BTP w.r.t. an ordering over \(X\). A Broken Triangle Property Network (BTPN) is a BCN satisfying BTP.

The BCN \(P\) given in Figure 1a does not satisfy BTP, as each variable in \(\{x, y, z\}\) has 2 valid literals included by a BT on \(P\). Then Figure 1b is a BTPN \(P'\) encoding \(P\) with 2 hidden variables, that does satisfy BTP w.r.t. the variable ordering \(h_{\{x,y,z\}} < h_{\{x,y\}} < x < y < z\).

We remark that a BTPN may not satisfy BTP w.r.t. some other orderings, e.g. \(\{(x, 1), (y, 1), (h_{\{x,y\}}, \mu_3), (h_{\{x,y\}}, \mu_4)\}\) is a BT in Figure 1b, and the BTPN \(P'\) does not satisfy BTP w.r.t. the ordering \(x < y < z < h_{\{x,y,z\}} < h_{\{x,y\}}\).

BCT is a special case of BTPN. For any BCT, there is a variable ordering \(O\) such that every variable \(v\) is constrained by at most one variable before \(v\) in \(O\). Then for any two variables \(x, y\) before a variable \(z\) in \(O\), there is no binary constraint between \(x, z\) or \(y, z\), i.e. there is no BT on the variables \(x, y, z\). Correspondingly, the BCT satisfies BTP w.r.t. the variable ordering \(O\), see Proposition 4.5 in (Cooper, Jeavons, and Salamon 2010).

The expressiveness of BTPN can be dramatically improved with hidden variables. For example, the monotone Conjunctive Normal Form (CNF) is not a BCN, however, it can be encoded as a BTPN with hidden variables. Every clause \(x_1 \lor \cdots \lor x_r\) in a monotone CNF \(F\) can be encoded with a hidden variables \(h\) and \(r\) binary constraints \(c_1, \cdots, c_r\) where \(D(h) = \{a_1, \cdots, a_r\}\) and for all \(i \in [1, r]\), the binary constraint \(c_i\) is defined as \(h \notin a_i \lor x_i = True\). Then the monotone CNF \(F\) can be encoded as a BTPN w.r.t. a variable ordering such that the hidden variables in the BTPN are before the original variables.

**GAC on BTPN Constraints**

The class of BTPNs is conservative and can be solved by AC. Therefore, we can enforce GAC on a BTPN constraint by enforcing AC on the BTPNs generated by assigning a variable value in the constraint. Then it is tractable to enforce AC on a BCN, so there exists polytime propagators to enforce GAC on the BTPN constraint.

We remark that AC on the BTPN encoding a constraint \(c\) is weaker than GAC on \(c\). For example, a BCP \(P\) modelled with the variables \(\{x_1, x_2, x_3\}\) and constraints \(\{x_1 \geq x_2, x_2 \geq x_3, x_1 + x_3 \leq 1\}\), where variable domains are \(\{0, 1\}\), satisfies BTP w.r.t. the variable ordering \(x_1 < x_2 < x_3\), however, AC on \(P\) cannot remove the literal \((x_3, 1)\) which is not included by any solution of \(P\).

In the future, it is interesting to explore a more efficient BTPN GAC propagator. For example, we can design a GAC propagator based on Directional Arc Consistency (DAC) (Dechter and Pearl 1987), as BTPN is solvable by DAC, see Proposition 5.5 in (Cooper, Jeavons, and Salamon 2010). In addition, similar to other binary encodings (Wang and Yap 2019, 2020, 2022b), it has potential to design specialized AC and DAC propagators for BTPN.

**5 Circuit Complexity of Checking BTPN**

The consistency checker (Bessiere et al. 2009) of a constraint \(c\) is a monotone function \(f_c\) where for any subset \(L\) of the literals \(\{(x, a)|x \in scp(c), a \in D(x)\}\), if there is a tuple \(\tau \in rel(c)\) such that \(\tau \subseteq \tau\), then \(f_c(L) = 1\), otherwise \(f_c(L) = 0\). Then a monotone circuit \(G\) is a rooted direct acyclic graph (DAG) where the leaves of \(G\) are 0, 1 values and the inner nodes are \(\land\) and \(\lor\) gates of which the children are inputs of the gates and the output of \(G\) is the output of the root. We remark that we abuse the values 0, 1 to denote the values \(False\) and \(True\).

The circuit complexity of \(f_c\) is useful for comparing the succinctness of different constraint representations. We then show that the consistency checker of BTPN constraint can be computed by polytime monotone circuit, therefore, some constraints cannot be encoded as polytime BTPN, such as the AllDifferent constraint (Régim 1994).

**Lemma 1.** There is a polytime monotone circuit computing the consistency checker of the constraints encoded with a class of BCNs which is conservative and solvable by AC.

**Proof.** Let \(P = (X, C)\) be the BCN encoding a constraint \(c\). WLOG, we assume \(P\) is AC and \(|\text{rel}(c)| > 0\). Then AC propagators can be simulated by a function \(f_P = f_0 \circ f_{P-1} \circ \cdots \circ f_1\) where \(k = \sum_{x \in X} |D(x)|\). The function \(f_0\) maps each tuple \(\tau\) over the Boolean variables \(B^i = \{b^i_{x,a}|(x, a)\}\) valid on \(P\) to a tuple \(\tau_{+1}\) such that \(\tau_{+1}\) is valid on \(P\) if \(\tau_{+1}\) is valid on \(P\) such that \(b^{i+1}_{y,b} = b^{i}_{y,b} \land (\land_{a \in X\setminus\{y\}} \lor_{a \in S_{y,b,x}} b^{i}_{x,a})\) where \(S_{y,b,x} = \{a \in D(x)|\{(x,a), (y,b)\}\}\) and \(\tau_{+1}\) denotes the valid supports of \((y, b)\) on \(x\).

For any assignment \(\tau\) over \(B^i\), if a literal \((y, b) \in L_{\tau}\) is not AC on a constraint in \(P_{\{\sigma\}}\), where \(\sigma\) denotes the literals \(\{(y, b)|b^{i}_{y,b,1} \in \tau\}\), then there is \(x \in X\) such
Incomparable, Permutation, AllDifferent, GCC, NValue, Channelling constraints.

In addition, if \( L_{f_p(\tau_i)} = L_{\tau_i} \) (i.e., \( P[L_{\tau_i}] \) is AC), then \( L_{\tau_i} \) is also equal to \( L_{\tau_{i+1}} \). \( P \) has at most \( k \) valid literals, so for any tuple \( \tau_i \) over \( B^1 \), \( L_{f_p(\tau_i)} \) gives the maximum subset of \( L_{\tau_i} \) such that \( P[L_{f_p(\tau_i)}] \) is AC.

\( P \) is in a conservative class solvable by AC, thus, for any tuple \( \tau_i \) over \( B^1 \), \( P[L_{\tau_i}] \) has a solution iff every \( x \in X \) has a literal in \( L_{f_p(\tau_i)} \): Hence, the consistency checker \( f_c \) can be computed by the function \( f^o \circ f_p \), where \( f^o = (\bigwedge_{x \in X} \bigvee b_{x,a}^1 \in B^1 b_{x,a}^k) \). We can construct the input of \( f^o \circ f_p \) based on the input of \( f_c \), where for any \( x \in \sigma(p) \) and \( (x,a) \notin L \), then \( b_{x,a}^1 = 0 \), else \( b_{x,a}^k = 1 \).

The functions \( f_p \) and \( f^o \) only use the \( \vee \) and \( \wedge \) operations, thus, \( f^o \circ f_p \) can be directly encoded as a DAG by regarding the \( \vee \) and \( \wedge \) operations as nodes where the children of the nodes are inputs of the operations. In addition, the number of \( \vee, \wedge \) operations used in the functions is \( O(k^3) \). So \( f_c \) can be computed by poly-size monotone circuit.

The class of BTPNs is conservative and solvable by AC, so the consistency checker of the BTPN constraint can be computed by poly-size monotone circuit.

**Theorem 1.** The consistency checker of BTPN constraints can be computed by poly-size monotone circuit.

Based on Theorem 1, we can separate BTPN from the constraints which cannot be computed by poly-size monotone circuit. For example, the constraints encoded with a system of XOR constraints cannot be computed by poly-size monotone circuit (see Section 6.2 in (Feder and Vardi 1998)), thus, they cannot be encoded as poly-size BTPN.

## 6 Encoding Various Constraints as BTPN

We then investigate the succinctness of BTPN on encoding various constraints. A constraint representation \( A \) is more succinct than another constraint representation \( B \) if every constraint encoded as \( B \) with a size \( n \) can be encoded as \( A \) with a size that is polynomial in \( n \). A constraint representation \( A \) is strictly more succinct than \( B \) if \( A \) is more succinct than \( B \), while \( B \) is not more succinct than \( A \). Then \( A \) is incomparable to \( B \) if \( A \) is not more succinct than \( B \), and \( B \) is not more succinct than \( A \).

Table 1 shows that BTPN is strictly more succinct than the DNNF constraint and all 14 ad-hoc constraints discussed in (Wang and Yap 2023), namely the table (Wang et al. 2016; Demeulenaere et al. 2016), regular (Pesant 2004), non-deterministic finite automaton (NFA) (Quimper and Walsh 2006), context free grammar (CFG) (Quimper and Walsh 2006), c-table (Katsirelos and Walsh 2007), MDD (Cheng and Yap 2010), short table (Jefferson and Nightingale 2013), sliced table (Gharbi et al. 2014), multi-valued variable diagram (MVD) (Amilhatre et al. 2014), smart table (smaT) (Maity, Deville, and Lecoutre 2015), basic smart table (Verhaeghe et al. 2017), semi-MDD (sMDD) (Verhaeghe, Lecoutre, and Schaus 2018), segmented table (segT) (Audemard, Lecoutre, and Maamar 2020) and BCT (Wang and Yap 2022b) constraints.

In addition, some special purpose global constraints are incomparable to the BTPN constraint, such as the permutation, AllDifferent (R´egin 1994), GCC (R´egin 1996), NValue (Pachet and Roy 1999), channelling (Cheng et al. 1999), cyclic, circuit (Beldiceanu and Contejean 1994), linear (Yuanlin and Yap 2000) and knapsack (Trick 2003) constraints.

**Theorem 2.** The results in Table 1 hold.

In the rest of this section, we give the proof of Theorem 2. The context free grammar, smart table and segmented table constraints are incomparable to each other (Wang and Yap 2023), thus, we only need to prove BTPN is more succinct than them. In addition, for the special purpose global constraints, if they cannot be encoded as poly-size BTPN, then BTPN is incomparable to them, as they cannot be used to encode arbitrary constraint relations.

### DNNF Constraints

DNNF is a subset of the negation normal form (Darwiche 1999). We consider a multi-valued variant (Gange and Stuckey 2012; Kučera 2023). A negation normal form (NNF) \( F \) over a set of variables \( X \) is a rooted DAG where each inner node is an or-node labeled with \( \vee \) or an and-node labeled with \( \wedge \); and each leaf is labeled with a literal of a variable in \( X \). For each node \( \eta \) in \( F \), we use \( \vars(\eta) \) to denote the set of variables \( x \in X \) such that \( \eta \) can reach a leaf labeled with a literal of \( x \). In addition, the set of variables \( \vars(F) \) is equal to \( \vars(\rho) \) where \( \rho \) is the root of \( F \). For technical convenience, we also allow the NNF consisting of a single node labeled with \( 0 \) or \( 1 \).

A NNF is decomposable (DNNF) if \( \vars(\eta_1) \wedge \vars(\eta_2) \) is empty for any \( 2 \) children \( \eta_1, \eta_2 \) of an and-node. Then a DNNF is smooth if \( \vars(\eta_1) = \vars(\eta_2) \) for any children \( \eta_1, \eta_2 \) of an or-node. A certificate (Bova et al. 2016) of a DNNF \( F \) over a set of variables \( X \) is a subgraph \( G \) of \( F \) such that (i) \( G \) is a DNNF including the root of \( F \), (ii) each or-node in \( G \) has exactly one child (iii) and each and-node \( \eta \) in \( G \) has the same children as \( \eta \) in \( F \). An assignment \( \tau \) over \( X \) satisfies \( F \) if there is a certificate \( G \) of \( F \) such that each leaf of \( G \) is labelled with \( 1 \) or a literal from \( \tau \).

A DNNF constraint \( c \) represents its constraint relation as a DNNF \( F \) over \( \vars(c) \) where \( \rel(c) \) consists of all assignments over \( \vars(c) \) satisfying \( F \). The size of \( c \) is the sum of the number of edges in \( F \) and variable domain sizes.

Each DNNF can be encoded as a smooth DNNF in polynomial time (Darwiche and Marquis 2002). Every and-node having one child can be merged with its only child. In addition, the DNNF with a single node labelled with 0,1 can be encoded as unary constraints. So in the rest of this subsection, we only consider the smooth DNNF \( F \) without any 0,1 labels and each and-node in \( F \) has at least 2 children.
Definition 3. A BTP binary encoding $btp(F)$ of a DNNF $F$ over variables $X$ is a DNNF $(X \cup H, C)$ where

- $H$ is a set of hidden variables $\{h_{vars(\eta)} | \eta$ is an and-node in $F \}$ and if $vars(F) = \emptyset$, then $D(h_{\eta}) = \{\text{and-nodes } \eta$ in $F \}$; else $D(h_{\eta}) = \{\eta \} \cup \{\text{and-nodes } \eta$ in $F \}$.
- For all $x \in vars(F), C$ has a constraint over $\{x\}$ where $\{(x, a)\}$ is in rel($c$) iff $F$ has a leaf labelled with $(x, a)$.
- For all $h_{\eta} \in H$ and $x \in V$, there is a constraint in $C$ between $x, h_{\eta}$ such that $\{(h_{\eta}, \eta_x, (h_x, a_2))\}$ is in rel($c$) iff $\eta_x \subseteq \eta$ or $\eta_x$ can reach a leaf labelled with $(x, a_2)$.
- For all $h_{\eta}, h_{\eta'} \in H$ in $V$ with $V_1 \cap V_2 \neq \emptyset$, there is a constraint in $C$ between $h_{\eta}, h_{\eta'}$ such that a tuple $\{(h_{\eta}, \eta), (h_{\eta'}, \eta')\}$ is in rel($c$) if it satisfies one of the following conditions:
  - $\eta_1 \supseteq \eta_2 = \emptyset$;
  - $\eta_1 = \emptyset$ and for all children $a$ of $\eta_2$, $V_1 \neq \emptyset$; 
  - $\eta_2 = \emptyset$ and for all children $a$ of $\eta_2$, $V_2 \neq \emptyset$; 
  - There is a path from $\eta_1$ to $\eta_2$ or from $\eta_2$ to $\eta_1$.

Example 2. Figure 2 shows a DNNF $F$ encoding the BCN given in Figure 1a, where $vars(F) = \{x, y, z\}$. $F$ is smooth, as for each or-node $\eta$ in $F$, all children of $\eta$ can reach the leaves labelled with literals of the same variables. Figure 1b gives the BTP binary encoding $btp(F)$ of the DNNF. Each certificate of $F$ corresponds to a solution of $btp(F)$, e.g. the subgraph with red color and dashed edges is a certificate of the DNNF, which corresponds to the solution $\{(h_{\{x,y,z\}}, \eta_1), (h_{\{x,y\}}, \eta_3), (x, 0), (y, 1), (z, 3)\}$.

Lemma 2. An assignment $\tau$ over $X \cup H$ is a solution of $btp(F)$ if and only if there is a certificate $G$ of the DNNF $F$ such that the and-node values in $\tau$ are the and-nodes of $G$ and the labels of all leaves of $G$ are also included in $\tau$.

Proof. Let $V = vars(F)$ and $n$ be the number of and-nodes in $F$. If $F$ does not have any and-node, then $|V| = 1$. The unary constraint over the only variable in $V$ can ensure that $\tau$ is a solution of $btp(F)$ iff $F$ has a leaf $\eta$ labelled with a literal included in $\tau$ where the path from the root to $\eta$ is a certificate of $F$. So the lemma holds for $n = 0$.

Assume the lemma holds for $n < k$. Then we prove it holds for $n = k$. There are 2 different cases.

Case 1: $|D(h_{\eta})| > 1$. For all and-nodes $\eta$ in $D(h_{\eta})$, let $F_{\eta}$ be an induced subgraph of $F$ on the nodes that are reachable by $\eta$ or can reach $\eta$ (note that $\eta$ is reachable by itself), and $H_{\eta}$ is the variables $h_S$ in $H$ such that all nodes in $D(h_S)$ are not reachable by $\eta$. $F_{\eta}$ has less and-nodes than $F$, as the nodes in $D(h_{\eta})$ cannot reach each other. For the BCN $btp(F_{\eta})$, if $h_{\eta} = \emptyset$, then all variables in $H_{\eta}$ are assigned with $\top$ and all assigned and-nodes must be reachable by $\eta$. Then $btp(F_{\eta})$ is a sub-problem of $btp(F)$ by removing values and the variables in $H_{\eta}$. For all $a \in D(h_{\eta})$, a tuple $\tau$ having $(h, \eta)$ is a solution of $btp(F_{\eta})$ iff each $h_S \in H_{\eta}$ is assigned with $\top$ and $\tau((X \cup H) \setminus H_{\eta})$ is a solution of $btp(F_{\eta})$. In addition, a subgraph $G$ having $\eta$ is a certificate of $F$ iff $G$ is a certificate of $F_{\eta}$. So the lemma holds for case 1.

Case 2: $D(h_{\eta}) = \{\eta\}$. For all children $\eta_i$ of $\eta$, let $F_{\eta_i}$ be an induced subgraph of $F$ on all nodes reachable by $\eta_i$, and $H_{\eta_i}$ be the variables $h_S$ in $H$ such that $D(h_{\eta_i})$ has and-nodes in $F_{\eta_i}$. The constraints between $h_{vars(\eta)}$ and $h_{\eta}$ ensure that $h_{vars(\eta)} \neq \tau$. Then for any two children $\eta_i, \eta_2$ of $\eta$, $F_{\eta_1} \cap F_{\eta_2}$ do not share any nodes and $H_{\eta_1} \cap H_{\eta_2} = \emptyset$, as $\eta_1 \cap \eta_2 \neq \emptyset$. Hence, an assignment $\tau$ over $X \cup H$ is a solution of $btp(F)$ iff all children $\eta_i$ of $\eta$, $\tau(H_{\eta_i} \cup X)$ is a solution of $btp(F_{\eta_i})$. In addition, a subgraph $G$ of $F$ is a certificate of $F$ if $G$ has exactly one or-path from the root to $\eta$, and all edges from $\eta$ and for all children $\eta_i$ of $\eta$, the subgraph of $G$ including the nodes reachable by $\eta_i$ is a certificate of $F_{\eta_i}$. So the lemma holds for case 2.

So by induction on $k$, the lemma holds for any $F$.

Lemma 3. The BTP binary encoding $(X \cup H, C)$ of a DNNF $F$ is a BTPN.

Proof. Let $O$ be an ordering over $X \cup H$ where (i) for any $h_{\eta_1}, h_{\eta_2} \in H$, if $V_2 \subseteq V_1$, then $h_{\eta_1}$ is before $h_{\eta_2}$; (ii) variables in $X$ are behind the variables in $H$. There is no constraint between the variables in $X$, so if $btp(F)$ does not satisfy BTP w.r.t. $O$, it has a $B$ $\{(h_{\eta_1}, \eta_1), (h_{\eta_2}, \eta_2), (y, a_3), (y, a_4)\}$ where $y \in X$ or $y \in H$ (i.e. $y = h_{\eta_3}$).

Case 1: $y \in X$. $\{(h_{\eta_2}, \eta_1), (y, a_3)\}$ and $\{(h_{\eta_2}, \eta_2), (y, a_3)\}$ are not consistent, so $a_1, a_2 \neq \top$ and $h_{\eta_1}, h_{\eta_2}$ then $a_1$ cannot reach $a_2$. Otherwise $\{(h_{\eta_2}, \eta_1), (y, a_3)\}$ is consistent due to $\{(h_{\eta_2}, a_2), (y, a_3)\}$ is consistent. Similarly, $a_3$ cannot achieve $a_1$. However, $\{(h_{\eta_3}, \eta_1), (h_{\eta_2}, a_3)\}$ is consistent and $a_3, a_2 \neq \top$ and $V_1 \cap V_3 \neq \emptyset$, which can imply that $a_1, a_2$ are connected (a contradiction). So $y \notin X$.

Case 2: $y = h_{\eta_3}$. The ordering implies $V_1 \cup V_2 \cup V_3$. $\{(h_{\eta_1}, \eta_1), (h_{\eta_2}, \eta_2)\}$ and $\{(h_{\eta_2}, \eta_2), (h_{\eta_3}, \eta_3)\}$ are not consistent, thus, $a_1, a_2 \neq \top, V_3 \cap V_1 \neq \emptyset$ and $V_3 \cap V_2 \neq \emptyset$. Then $\{(h_{\eta_1}, a_1), (h_{\eta_2}, a_3)\}$ is consistent, so $a_3 = \top$ or $a_1$ can reach $a_3$. So $a_2$ cannot reach $a_1$, otherwise $\{(h_{\eta_2}, a_3), (h_{\eta_3}, \eta_3)\}$ is consistent. Similarly, $a_1$ cannot reach $a_2$. Next, if $V_3 \subseteq V_1$ and $V_2 \subseteq V_2$, then $a_1, a_2$ cannot reach $a_3$, thus, $a_3 = a_4 = \top$ (this is impossible). Hence, we have $V_3 \subseteq V_1$ such means $V_1 \cap V_3 \neq \emptyset$ and $a_1, a_2$ are connected (a contradiction). So $y \notin H$.

Both cases are impossible, so $btp(F)$ is a BTPN.

Lemma 4. BTPN is strictly more succinct than DNNF.

Proof. Lemma 2 and Lemma 3 show that for any DNNF $F$ over variables $X$, the BTP binary encoding $btp(F)$ is a BTP encoding $F$. The number of variables in $btp(F)$ is at most $n + |X|$, and the total size of all hidden variable...
domains is less than $2(n+1)$, where $n$ is the number of and-nodes in $F$. Therefore, the size of $btp(F)$ is polynomial in the size of $F$, and BTPN is more succinct than DNNF.

In addition, 2-SAT can be encoded as polysize BTPN by enforcing path consistency (Cooper, Jeavons, and Salamon 2010), and monotone 2-SAT cannot be encoded as polysize DNNF (Bova et al. 2014). Therefore, we can get that BTPN is strictly more succinct than DNNF.

### CFG Constraints
(Quimper and Walsh 2007) showed that the CFG constraint can be encoded as a polysize rooted DAG (and/or graph) $G$ where each and-node in $G$ has exactly 2 children that encodes tuples over 2 variable subsets partitioning a set of variables, which means that the rooted DAG $G$ is a DNNF. So BTPN is more succinct than the DNNF constraint (based on Lemma 4) and the CFG constraint.

### Smart Table Constraints
A smart tuple over variables $X$ is a set $S$ of specific unary and binary constraints such that the constraint graph of $(X, S)$ is acyclic (Maity, Deville, and Lecoutre 2015). Then $(X, S)$ satisfies BTP w.r.t. an ordering over $X$ such that every variable $x \in X$ is constrained by at most one variable before $x$. So the smart tuple $(X, S)$ itself is a BTPN.

A smart table (segmented table) $R$ is a disjunction of a set of smart tuples (segmented tuples) where the tuples encoded by $R$ is the union of the tuples encoded by the smart tuples (segmented tuples). Then the disjunction of BTPNs can be computed in polytime (see Lemma 6), thus, the smart table and segmented table can be encoded with polysize BTPN.

### Segmented Table Constraints
A segmented tuple over variables $X$ is a set $S$ of specific unary constraints and table constraints such that for any 2 constraints $c_1, c_2 \in S$, $scp(c_1) \cap scp(c_2) = \emptyset$ (Audemard, Lecoutre, and Maamar 2020). The hidden variable encoding (Rosi, Petrie, and Dhar 1990) of $(X, S)$ is acyclic, thus, it is also a BTPN encoding the segmented tuple.

A segmented table $R$ is a disjunction of a set of segmented tuples where the tuples encoded by $R$ is the union of the tuples encoded by the segmented tuples. Therefore, the segmented table can be encoded as polysize BTPN.

### Other Ad-Hoc Constraints
Except for the smart table and segmented table constraints, the CFG constraint is more succinct than the other 11 ad-hoc constraints discussed in (Wang and Yap 2023). Therefore, BTPN is strictly more succinct than all 14 ad-hoc constraints discussed in (Wang and Yap 2023).

### Permutation Constraints and Its Generalizations
A permutation constraint $c$ over $r$ variables $\{x_1, \cdots, x_r\}$ encodes that the $r$ variables take distinct values between 1 and $r$. As shown in (Regin 1994), the permutation constraint $c$ can be regarded as a bipartite graph $G$, where literals are edges and each tuple in $rel(c)$ is a perfect matching of $G$. Then (Razborov 1985) showed that there is no polysize monotone circuit determining whether $G$ has a perfect matching, thus, the permutation constraint cannot be encoded as polysize BTPN. In addition, the AllDifferent, GCC and NValue constraints generalize the permutation constraint. So they all cannot be encoded as polysize BTPN.

### Channeling Constraints
A channeling constraint encodes a connection between two sets of variables (Cheng et al. 1999; Walsh 2001). The permutation constraint can be modelled as a projection of the channeling constraint on a set of variables (Walsh 2001). Then the BTPN encoding a constraint can also encode its projections. Then there is no polysize BTPN encoding the permutation constraint, thus, there is no polysize BTPN encoding the channeling constraint.

### Circuit and Cycle Constraints
A circuit constraint encodes Hamilton circuits on a graph (Beldiceanu and Contejean 1994). Then (Alon and Boppana 1987) showed that there is no polysize monotone circuit checking whether a graph has a Hamilton circuit. In addition, the circuit constraint is a kind of cycle constraints (Beldiceanu and Contejean 1994). So the circuit and cycle constraints cannot be encoded as polysize BTPN.

### Knapsack and Linear Constraints
A knapsack constraint over $r$ variables $\{v_1, \cdots, v_r\}$ is $l \leq \sum_{i=1}^{r} w_i v_i \leq u$ (Trick 2003). We use $c_r$ to denote the constraint $\sum_{i=1}^{r} r^i \leq \sum_{i=1}^{r} i^r \leq \sum_{i=1}^{r} i^r$ where $D(v_i) = \{r^k|k \in [1, r]\}$. There must be a variable in $c_r$ taking the value $r^r$, as $r^r > \sum_{i=1}^{r-1} i^{r-1}$. Then $r^k > \sum_{i=1}^{k-1} i^{k-1}$ for all $k \in [1, r]$. So by induction on $k$, we can get that the variables in $c_r$ take different values from $\{r^k|k \in [1, r]\}$.

Hence, we can use the consistency checker $f_{c_r}$ to compute the consistency checker $f_c$ of a permutation constraint $c$ over $r$ variables $x_1, \cdots, x_r$. For any literals $L_1$ of variables in $scp(c)$, $f_c(L_1)$ is equal to $f_{c_r}(L_2)$ where $L_2 = \{(v_i, r^r)(x_i, a) \in L\}$. Therefore, the monotone circuit computing $f_{c_r}$ can also be used to compute $f_c$.

Then the permutation constraint cannot be computed by polysize monotone circuit, and $c_r$ is a knapsack and linear constraint, therefore, there is no polysize monotone circuit computing the knapsack and linear constraint.

### 7 Time Complexity of Minimizing BTPN
A constraint can be encoded with different BTPNs, thus, an interesting problem is to determine the minimum sized BTPN encoding the constraint. In this section, we show that two specific biclique cover problems ($P1$ and $P2$) on a bipartite graph $G$ are NP-hard and can be solved by minimizing the BTPN encoding a constraint, where a biclique cover of a subset $S$ of edges in $G$ is a set of complete bipartite subgraphs (CBS’s) of $G$ including all edges in $S$. Correspondingly, the BTPN minimization problem is NP-hard.

Let $G = (U, W, E)$ be a bipartite graph. Without loss of generality, we assume that all vertices in $G$ are included by
at least one edge in $E$. Note that the biclique covers of a subset of edges in $E$ are not affected by removing the vertices which are not included by any edge in $E$.

Then we construct a constraint $c_G$ between 2 variables $x_V$ and $x_W$ such that $D(x_V) = U \cup A \cup B$ and $D(x_W) = W \cup A \cup B$ and $rel(c_G) = R_G \cup R_A \cup R_B$ where $|A| = |B| = 3|E| + 10$ and $(A \cup B) \cup (U \cup W) = \emptyset$ and $A \cap B = \emptyset$ and $R_G = \{(x_U, \mu), (x_W, \omega)\} \in E$) and $R_A = \{(x_V, a_1), (x_W, a_2)\} \in A$ and $R_B = \{(x_V, b), (x_W, B)\} \in B$.

The values in $A, B$ are chosen to ensure that the minimum BCN encoding $c_G$ has exactly one hidden variable and 2 binary constraints including the hidden variable.

**Lemma 5.** The minimum BCN $P$ encoding $c_G$ has exactly one hidden variable and 2 constraints between the hidden variable and the 2 variables $x_V$, $x_W$.

**Proof.** Let $P_1 = \{(x_W, x_V, h), (c_W, c_U)\}$ be a BCN encoding $c_G$ with a hidden variable $h$ where $D(h) = E \cup B \cup \{a\}$ and $c_W$ is defined as $(h = a \land x_V \in A) \lor (h \in E \land x_W \in h) \lor (h \in B \land x_W = h)$ and $c_U$ is $(h = a \land x_U \in A) \lor (h \in E \land x_U \in h) \lor (h \in B \land x_U = h)$. By the construction, the size of $P_1$ is $4|A| + 5|B| + 3|E| + |W| + |U| + 1$.

$rel(c_G)$ has $|A| + |E| + |B|$ tuples, thus, $P$ does not have any constraint between 2 variables in $x_V$. Then $rel(c_G)$ is not a universal relation, so $x_W$, $x_U$ must be constrained by hidden variables in $P$. Let $k_W$ and $k_U$ be the number of constraints including $x_W$ and $x_U$, respectively. Then the size of $P$ is at least $(k_W + 1)(|A| + |B| + |W|) + (k_U + 1)(|A| + |B| + |U|)$, as each value of $x_W$, $x_U$ is included by at least 1 tuple on each constraint. The size of $P$ is not greater than that of $P_1$, thus, $k_W = 1$ and $k_U = 1$.

Assume $x_W$ ($x_U$) is constrained by a hidden variable $h_W$ ($h_U$). For all $b_2 \in B$, there is $b \in D(h_W)$ such that $(b, h_W)$ can be extended to a solution of $P$ having $(x_V, b_1)$ and for all other $b_2 \in B$, $(x_W, b_2, (h_W, b))$ is not consistent on $P$ due to $\{(x_W, b_2), (x_U, b_1)\} \notin rel(c_G)$. Hence, $h_W$ has at least $|B|$ values. Similarly, $h_U$ has at least $|B|$ values. So $h_W = h_U$, otherwise the size of $P$ is greater than that of $P_1$.

$x_W$ and $x_U$ are constrained by the same hidden variable, which means the other hidden variables in $P$ can be eliminated. So $P$ has exactly one hidden variable and 2 constraints between the hidden variable and $x_V$, $x_U$.

Therefore, the minimum BCN encoding $c_G$ here is also a BCT and BTPN. We then show that it is NP-hard to minimize the BCN, BCT and BTPN encoding $c_G$.

**Theorem 3.** The following 4 problems are NP-hard:

- **P0** Determine the fewest number of cliques which include all of the vertices of a graph.
- **P1** Determine the fewest number of vertices included in a biclique cover of a specified subset $S$ of edges of $G$.
- **P2** Determine the fewest number of vertices included in a biclique cover of the edges of $G$.
- **P3** Find the minimum BCN/BCT/BTPN encoding $c_G$.

**Proof.** Theorem 8.1 in (Orlin 1977) shows that $P0$ is NP-hard and can be solved by determining the fewest number of CBS’s included by a biclique cover of the subset $S = \{\mu_1, \omega_1, \ldots, \mu_n, \omega_n\}$ of edges of a bipartite graph $G$.

We then prove the NP-hardness of $P1$, $P2$, $P3$ by reducing (i) $P0$ to $P1$ ($P0 \propto P1$) and (ii) $P1$ to $P2$ ($P1 \propto P2$) and (iii) $P2$ to $P3$ ($P2 \propto P3$).

$P0 \propto P1$. Let $B_G$ be a biclique cover of $S$ having the fewest number of vertices, where $S = S_1 \cup S_2$ and $S_1 = \{\mu_1, \omega_1, \ldots, \mu_n, \omega_n\}$ and $S_2 = \{\mu_1, \omega_1, \ldots, \mu_n, \omega_n\}$. Each edge $\{\mu_i, \omega_i\}$ in $S_1$ is only included by 1 CBS in $B_G$, otherwise $\mu_i$ can be removed from some CBS’s to reduce the number of vertices. If there is a CBS $G'$ in $B_G$ which has $\{\mu_i, \omega_i\}$ but not $\{\mu_i, \omega_i\}$, then we can remove $\omega_i$ from $G'$ and add $\mu_i$ to a CBS having $\{\mu_i, \omega_i\}$. Hence, we can assume $\{\mu_1, \omega_1\}$ and $\{\mu_i, \omega_i\}$ are included by the same CBS in $B_G$. So the number of vertices in $B_G$ is equal to $|S_1| + 2n$. Note that $B_G$ can also be regarded as a biclique cover of $S$ (by removing $\mu_i$). In addition, we can construct a biclique cover of $S$ with $|B_G| + 2n$ vertices by adding $\mu_i$ to all CBS’s in any biclique cover $B_G$ of $S_1$. So $|B_G|$ is the fewest number of CBS’s covering $S_1$, i.e. $P1$ is NP-hard. $P1 \propto P2$. Let $S^* = \{x_U, x_V, h\}$ be a biclique cover of $G$ having 3 edges $e_1 = \{x_U, x_V\}$, $e_2 = \{x_U, h\}$ and $e_3 = \{x_V, h\}$ for each edge $e = \{x_U, x_V\}$ in $S^*$. A CBS having $x_U$ or $x_V$ can only have the vertices $x_U$, $x_V$, $\mu$, or $\omega$. So the CBS’s with the fewest vertices covering the 3 edges $e_1, e_2, e_3$ are the CBS with the vertices $x_U$, $x_V$, $\mu$, or $\omega$, which can also cover $e$. Correspondingly, the biclique cover of $G_1$ with the fewest number of vertices consists of a biclique cover of the edges in $S$ with the fewest number of vertices and the $|S^*|$ CBS’s covering the other edges. So $P2$ is also NP-hard. $P2 \propto P3$. Let $P = \{x_W, x_U, h\}$ be a minimum BCN/BCT/BTPN encoding $c_G$ (based on Lemma 5). For each value $a \in D(h)$, we can construct a CBS $G_a$ with the vertices $\mu \in U, \omega \in W$ such that $\{x_U, \mu, (h, a)\} \in rel(c_G)$ and $\{(x_W, h), (a)\} \in rel(c_G)$. Then $B_G = \{G_a | a \in D(h), G_a$ is not empty $\}$ is a biclique cover of $G$, where the number of tuples including values of $U \cup W$ is equal to the number of vertices in $B_G$. If $B_G$ is not a biclique cover with the fewest vertices, then we can construct a smaller BCN/BCT/BTPN encoding $c_G$ by replacing the values $\{a \in D(h) | G_a$ is not empty $\}$ with the CBS’s in a biclique cover $B_G$ having fewer vertices, where for each CBS $G'$ in $B_G$ and any vertices $\mu \in U, \omega \in W$ in $G'$, the tuples $\{(x_U, \mu), (h, G')\}$ and $\{(x_W, h), (h, G')\}$ are constructed. Therefore, $B_G$ must be a biclique cover of $G$ with the fewest vertices, which means $P3$ is NP-hard.

8 Operations and Queries on BTPN

We consider the operations and queries from (Darwiche and Marquis 2002). More details of the operations and queries on constraints can be found in (Wang and Yap 2023). Table 2 gives the complexity of computing them on BTPN.

**Theorem 4.** The results in Table 2 hold.

It is NP-hard to compute the conjunction between 2 BTPNs ($A \land B$), as it is NP-hard on DNNF (Darwiche and Marquis 2002). The permutation constraint (a conjunction of binary constraints) cannot be encoded as polysize BTPN,
so the conjunction of a set of BTPNs (\( \bigwedge S \)) cannot be computed in polytime. Lemma 6 shows that the disjunction of BTPNs (the \( A \lor B \) and \( \bigvee S \) operations) can be computed in polytime. Then the negation of the permutation constraint can be encoded as polytime BTPN, thus, the negation of a BTPN (\( \neg A \)) cannot be computed in polytime.

The singleton forgetting (SFO) and forgetting (FO) operations can be implemented by the projection operation, while every BTPN (\( X,C \)) also encodes the projection \( \text{sol}(X,C)[V] \) for any variables \( V \subseteq X \). Hence, the SFO and FO operations can be computed in polytime. In addition, the class of BTPNs is conservative, thus, the conditioning (CD) operation can be computed in polytime.

BTPN is strictly more succinct than DNNF, so if NP is not in P, then there is no polytime algorithm to compute the validity (VA), implicant (IM), equivalence (EQ), sentential entailment (SE) and model counting (CT) queries (Darwiche and Marquis 2002). The class of BTPNs is conservative and solvable by AC, thus, the consistency (CO) and clause entailment (CE) queries can be computed in polytime. The time complexity of computing the model enumeration (ME) query on a BTPN \( P \) is polynomial in the sum of \(|\text{sol}(P)|\) and the size of \( P \), as CO and CD are tractable (Darwiche and Marquis 2002).

**Lemma 6.** The conjunction of a set of BTPNs \( \{(X_1, C_1), \ldots, (X_k, C_k)\} \) can be computed in polytime.

**Proof.** Assume \( X = \bigcup_{i=1}^{k} X_i \) and \( X = \{x_1, \ldots, x_n\} \) and for all \( i \in [1,k] \), \( P_i = (X,C_i) \) satisfies BTP w.r.t. an ordering \( O^i \) over \( X \). Then let \( Y = \{y_1, \ldots, y_n\} \) and \( Z = \{z_1, \ldots, z_n\} \). For all \( 1 \leq j \leq n \), \( D(y_j) = \{a^i | a \in D(O^i)_j, i \in [1,k] \} \) merges the domains of the \( k \) variables \( O^1_j, \ldots, O^k_j \), and \( D(z_j) = \{a^i | a \in D(x_j), i \in [1,k] \} \) consists of \( k \) copies of the domain of \( x_j \).

We can construct a BCN \( P \) over \( X \cup Y \cup Z \) where (i) \( \{(y_j, a^{i_1}), (y_j, a^{i_2})\} \) is consistent on \( P \) iff \( i_1 = i_2 \) and \( \{(O^1_j, a), (O^2_j, b)\} \) is consistent on \( P_i \); (ii) \( y_j = a^i \Leftrightarrow z_j = a^i \) for all \( i, j \); (iii) \( z_j \in \{a^i | i \in [1,k] \} \Leftrightarrow x_j = a \) for all \( x_j \in X \) and \( a \in D(x_j) \).

For each \( i \in [1,k] \) and a solution \( \tau \) of \( P_i \), \( \tau \) corresponds to a solution \( \tau' \cup \{(y_j, a^i) | (O^i_j, a) \in \tau \} \cup \{(z_j, a) | (x_j, a) \in \tau \} \) of \( P \), where \( \text{sol}(P)[X] = \bigcup_{i=1}^{k} \text{sol}(P_i) \).

Let \( O \) be an ordering \( y_1 < \cdots < y_n < z_1 < \cdots < z_n \).\( \cdots < z_n \). There is no constraint between the variables in \( Z \), and each variable \( x_j \) is only constrained by \( z_j \) on \( P \), thus, if \( P \) does not satisfy BTP w.r.t \( O \), then \( P \) must have a BT \( \{(y_j, a^1), (y_j, a^2), (y_j, a^3), (y_j, a^4)\} \) where \( v_j \) is in \( \{y_j, z_j\} \) and \( y_j < z_j < v_j \) is a subsequence of \( O \). In addition, \( v_j \) and \( a^i \) are only constrained by a variable \( y_j \) in \( Y \), i.e. constraint (ii), such that \( O^1_j = x_j \). Hence, \( v_j \) is \( y_j \), and the BT is \( \{(y_j, a^1), (y_j, a^2), (y_j, a^3), (y_j, a^4)\} \). However, this implies that \( \{(O^1_j, a^1), (O^2_j, a^2), (O^3_j, a^3), (O^4_j, a^4)\} \) is a BT on \( P_i \) (a contradiction). So \( P \) satisfies BTP w.r.t \( O \).

\( P \) is a polytime BTPN which encodes the disjunction \( \bigvee_{i=1}^{k} P_i \), therefore, the disjunction of BTPNs \( \{(X_1, C_1), \cdots, (X_k, C_k)\} \) can be computed in polytime.

<table>
<thead>
<tr>
<th>Operation</th>
<th>VA</th>
<th>CE</th>
<th>IM</th>
<th>EQ</th>
<th>SE</th>
<th>CT</th>
<th>ME</th>
</tr>
</thead>
<tbody>
<tr>
<td>∧</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>∨</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>¬</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>⊇</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2: Operations and queries on BTPN: ✓(*)(*) means the complexity of computing an operation or query is in polytime (not in polytime, not in polytime unless NP=P).

9 Discussion

Many tractable classes have been proposed to generalize the class of BTPNs (Cohen et al. 2012; Naanaa 2013; Cohen et al. 2015; Jegou and Terrioux 2015; Cooper, Jegou, and Terrioux 2015; Cooper and Živný 2016). In the future, it is worth to investigate whether those tractable classes can improve further the succinctness of BTPN. Usually, the efficiency of propagators is affected by constraint size, thus, an interesting line of research is exploring the most succinct tractable classes having polytime GAC propagators. Moreover, it is also possible to push the study of structural classes forward by identifying more succinct tractable classes, since BCT is already as succinct as the most general known tractable structural class.

We show BTPN is strictly more succinct than DNNF, while it is as powerful as DNNF in terms of computing operations and queries. We propose BTPN as an interesting alternative to DNNF from a knowledge compilation perspective. This allows exploring subclasses of BTPNs to identify valuable alternatives to the subsets of DNNFs in the knowledge compilation map (Darwiche and Marquis 2002).

The circuit complexity of consistency checker is very useful for separating constraint representations. It can separate the DNNF, smart table and segmented table from the permutation constraint and a system of XOR constraints, correspondingly several questions posed in (Fargier and Marquis 2008; Wang and Yap 2023) can be resolved.

In addition, various rules based on BTP and forbidden patterns have been applied to merge values and eliminate variables (Cohen et al. 2013; Cooper et al. 2014; Cooper, El Mouelhi, and Terrioux 2016, 2019). A future research direction is to explore whether these rules can be extended to reduce BTPN.

10 Conclusion

In this paper, we propose to encode constraints as Binary Constraint Networks satisfying Broken Triangle Property (called BTPNs). We prove that the consistency checker of the BTPN constraint can be computed by polytime monotone circuit, thereby, some global constraints cannot be encoded as polytime BTPN, such as the permutation, circuit and linear constraints. Then we show that BTPN is strictly more succinct than the DNNF constraint and all 14 ad-hoc constraints discussed in (Wang and Yap 2023). Moreover, we prove that the BTPN minimization problem is NP-hard. Finally, we also investigate the tractability of various operations and queries on the BTPN constraint.
Acknowledgements

We thank Prof. Roland Yap for his useful comments. This work was supported by the grant T1 251RES2219.

References


