Variable Ordering for the Application of BDDs to the Maximum Independent Set Problem

David Bergman, Andre A. Cire, Willem-Jan van Hoeve, J. N. Hooker

Tepper School of Business, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, U.S.A.
{dbergman,acire,vanhoeve}@andrew.cmu.edu, john@hooker.tepper.cmu.edu [*]

Abstract. The ordering of variables can have a significant effect on the size of the reduced binary decision diagram (BDD) that represents the set of solutions to a combinatorial optimization problem. It also influences the quality of the objective function bound provided by a limited-width relaxation of the BDD. We investigate these effects for the maximum independent set problem. By identifying variable orderings for the BDD, we show that the width of an exact BDD can be given a theoretical upper bound for certain classes of graphs. In addition, we draw an interesting connection between the Fibonacci numbers and the width of exact BDDs for general graphs. We propose variable ordering heuristics inspired by these results, as well as a k-layer look-ahead heuristic applicable to any problem domain. We find experimentally that orderings that result in smaller exact BDDs have a strong tendency to produce tighter bounds in relaxation BDDs.

1 Introduction

In recent years, Binary Decision Diagrams (BDDs) [1, 17, 7] have been regarded as a powerful tool for a variety of purposes in Operations Research. Their application in this domain is primarily as a graphical data structure that aims at a compact representation of the set of feasible solutions to a Constraint Satisfaction Problem (CSP). Examples of applications include the generation of cuts in a Branch-and-cut framework [3], post-optimality analysis for Integer Programming [12, 13], and 0/1 vertex and facet enumeration [4].

This perspective of BDDs is nonetheless associated with inherent difficulties. First, constructing the exact BDD for a CSP instance is in general an NP-hard problem, since this procedure is as hard as deciding the feasibility of the instance. Furthermore, even for problems where feasibility can be determined efficiently, the BDD may grow exponentially large, and thus it is not computationally practical to derive exact representations for most relevant problems.

In light of these difficulties, the work in [2] proposes the use of limited-size BDDs as an *approximate* representation for CSPs. Namely, limited-size BDDs are

[*] This work was supported by NSF under grant CMMI-1130012 and AFOSR under grant FA-95501110180.

constructed in a way that they contain the feasible space of a particular CSP, but may potentially include infeasible solutions due to the imposition of a polynomial bound on their size according to an input parameter. Such approximations have led to research along several directions, such as replacing the traditional domain store relaxation used in Constraint Programming systems [2, 14, 15].

In this context, the work in [5] introduced a systematic approach to generate approximate BDDs for Combinatorial Optimization problems. The authors focused on using BDDs to represent a relaxation of the set of feasible solutions to the Set Covering Problem (SCP). These structures were introduced for the purpose of proving bounds on the objective function for the SCP. It also proposes methods for tightening the bound provided by the relaxed BDDs, analogous to the use of cutting planes in Integer Programming (IP). It was shown by the authors that for structured instances of the SCP, the relaxations based on BDDs can provide substantially tighter bounds than the classical linear relaxation.

We further extend this line of research in the present paper, investigating one of the crucial aspects of BDDs applied to Operations Research problems: How does the ordering of the variables within a BDD affect the quality of the relaxation it provides. In particular, we are interested in identifying the relation between the size of an exact BDD for a CSP, which is directly correlated to the variable ordering applied, and the bounds obtained through its corresponding approximate BDD when an objective function is considered.

The development of good variable orderings requires identifying the underlying cause of the combinatorial explosion of the size of exact BDDs. For this purpose, this work focuses on the Maximum Independent Set Problem (MISP), exploring the following main topics. We first introduce a technique to efficiently construct the BDD representing the family of independent sets of a graph. Next, we provide a thorough study of orderings that yield polynomially-bounded BDD sizes for particular classes of graphs. Through this analysis we uncover an interesting connection between the size of exact BDDs for arbitrary graphs and the Fibonacci numbers, yet another curious property of independent sets [8, 11, 9, 18]. Interestingly, we illustrate how the underlying principles in the proof of these bounds on the width can be used to develop good ordering heuristics. Finally, we experimentally show in this paper that variable orderings that yield small-sized exact BDDs are critical for their application as a bounding technique to optimization problems, resulting in substantially better bounds for the MISP when compared to other orderings.

The contributions of this work potentially go beyond the scope of independent set problems. Namely, we presented the first systematic analysis and empirical evidence of how variable orderings can positively affect approximate BDDs in Combinatorial Optimization. This analysis may be extended to various other problem domains. We particularly reinforce the claim that investigating orderings for particular problem classes can lead to good heuristics that are potentially applicable to other problems. In particular, we introduce the general-purpose variable ordering heuristic *k-stage lookahead*, that yielded the best results for the MISP and can be directly used for any CSP.

This paper is organized as follows. In Section 2 we formally introduce BDDs. In Section 3 we discuss how exact BDDs for the MISP can be constructed. In Section 4 we investigate variable ordering for particular classes of instances of the MISP and prove bounds on the size of the exact BDDs for these problems. In Section 5 we discuss variable ordering heuristics for general graphs. Finally, in Section 6 we provide computational results and conclude in Section 7.

2 Preliminaries and Notation

CSPs A CSP $(X, D, \mathcal{C})$ is defined by a finite set of variables X, a set of discrete domains D such that $D(x) \in D$ restricts the values $x \in X$ can take, and a set of constraints $\mathcal{C}$. A *solution* to a CSP corresponds to an assignment of values to the variables and it is *feasible* if all values are within the variable domains and are consistent with $\mathcal{C}$. A *Constraint Optimization Problem* (COP) is given by a CSP alongside an objective function f to be maximized. For simplicity, we consider here only CSPs with binary domains, *i.e.*, $D(x) = \{0, 1\}$ for all $x \in X$.

BDDs We are interested in representing the set of solutions of a CSP by a *Binary Decision Diagram* (BDD). A BDD $B = (U, A, d)$ is a directed acyclic graph whose nodes U are partitioned into m layers, $U = \cup_{i=1}^{m} L_i$. The layer of a node u is given by $\text{var}(u)$. Layers L_1 and L_m consist of single nodes; the root r and the terminal t, respectively. The *width* ω_j of a layer j is defined as $\omega_j := |L_j|$, and the *width* of B is given by $\omega(B) := \max_j \omega_j$. Let $|B| = |U|$ be the *size* of the BDD. All arcs $a \in A$ are directed from nodes in layer j to nodes in layer $j + 1$, for some $j \in \{1, \ldots, m - 1\}$. The function $d : A \to \{0, 1\}$ associates each arc a with a label $d(a) = d_a \in \{0, 1\}$; a is referred to as a *one-arc* if $d_a = 1$ and as a *zero-arc* if $d_a = 0$. Each node u can have at most one one-arc and at most one zero-arc directed out of it. For any node u, there must exist a directed path from r to u and from u to t. A BDD representing a set of solutions of a CSP $(X, D, \mathcal{C})$, with $n = |X|$, has $m = n + 1$ layers. Each layer L_i is uniquely associated with a variable $x \in X$; we denote this variable by x_i. An arc a directed from layer L_i to L_{i+1} with label d_a identifies an assignment $x_i = d_a$. Hence, a directed path from r to t corresponds to a solution of the CSP. The set of solutions represented by a BDD B (*i.e.*, on all directed paths from r to t) is denoted by $\text{Sol}(B)$.

For a given node $u \in U$, we let $B^+|_u$ be the subgraph of B induced by the subset of nodes composed of u, the root $r \in U$, and all nodes $v \in U$ lying on some directed path from r to u. In addition, we preserve the arc labels as in B; therefore, $B^+|_u$ is also a BDD. Analogously, let $B^-|_u$ be the subgraph of B induced by the subset of nodes composed by u, the terminal $t \in U$, and all nodes $v \in U$ such that there is a directed path from u to t. Also, let $B^+|_{L_j}$ be the digraph induced by $L_1, \ldots, L_j$ and similarly $B^+|_{L_j}$ be the digraph induced by $L_j, \ldots, L_{n+1}$, with $\text{Sol}(B^+|_{L_j}) = \cup_{u \in L_j} \text{Sol}(B^+|_u)$ and $\text{Sol}(B^-|_{L_j}) = \cup_{u \in L_j} \text{Sol}(B^-|_u)$. A *reduced* BDD is one for which $\text{Sol}(B^-|_u) \neq \text{Sol}(B^-|_{u'})$ for any two nodes u and u' on the same layer. It can be shown that for a particular ordering of the variables, that is, how layers are mapped into variables, there is one unique reduced BDD for any set of solutions [7].

MISP In this paper we study variable orderings for the *Maximum Independent Set Problem* (MISP). Let $G = (V, E)$ be a simple undirected graph. An *independent set* of G is a set $I \subseteq V$ such that $(w, v) \notin E$ for any distinct $w, v \in I$. We denote $\mathcal{I}(G)$ as the family of independent sets in G. The MISP consists of finding a set $I \in \mathcal{I}(G)$ with the largest cardinality.

BDD representation for MISP For notation purposes, let $G[W]$ be the graph induced by a subset $W \subseteq V$ and let $\overline{W} := V \backslash W$. A corresponding BDD for the COP above defines a bijection between the vertices $v \in V$ and the layers $L_1, \ldots, L_n$; let v_j be the associated layer of vertex v, with $V_j = \{v_1, \ldots, v_j\}$. With every path $p = (a_1, \ldots, a_n)$ from the root r to the terminal t we associate a subset $I^p \subseteq V$ defined by $I^p := \{v_j : d_{a_j} = 1\}$. Likewise, for a node u, any path $p = (a_1, \ldots, a_{j-1})$ in $B^+|_u$ corresponds to a vertex subset in $G[V_{j-1}]$ and any path $p = (a_j, \ldots, a_n)$ in $B^-|_u$ corresponds to a vertex subset in $G[\overline{V}_{j-1}]$. Note that each solution corresponds to at most one path in any BDD because no node has two arcs with the same label directed out of it.

A BDD B is *exact* for a graph G if $\text{Sol}(B) = \mathcal{I}(G)$, and it is a *relaxation* for G if $\text{Sol}(B) \supseteq \mathcal{I}(G)$. In an exact BDD, $\mathcal{I}(G[V_{j-1}]) = \text{Sol}(B^+|_{L_j})$ and $\mathcal{I}(G[\overline{V}_{j-1}]) = \text{Sol}(B^-|_{L_j})$. In a relaxation BDD, we similarly have $\mathcal{I}(G[V_{j-1}]) \subseteq \text{Sol}(B^+|_{L_j})$ and $\mathcal{I}(G[\overline{V}_{j-1}]) \subseteq \text{Sol}(B^-|_{L_j})$.

By associating a cost d_a to each arc in a BDD B, the longest path from r to t yields a maximum cardinality independent set of G, if B is an exact BDD. If otherwise B is a relaxation BDD, the longest path corresponds to a subset of the vertices whose cardinality is greater than or equal to the size of the maximum independent set, thereby establishes an upper-bound on the value of the optimal solution to the MISP on G. We note that since BDDs are layered graphs, the longest path can be computed in polynomial time in $|B|$.

Additional Notation For a graph $G = (V, E)$, two disjoint subsets $I, J \subset V$ are *independent* if $(w, v) \notin E$ for any $w \in I$, $v \in J$. The *neighborhood* $N(v)$ of $v \in V$ is defined as $N(v) = \{w : (w, v) \in E\}$. A *partial solution with respect to* $W \subseteq V$ corresponds to any subset $I \subseteq W$, which is *feasible* if $I \in \mathcal{I}(G[W])$. Given a partial feasible solution I with respect to W, the *set of feasible completions of* I *with respect to* W is given by $C(I \mid \overline{W}) = \{J \mid J \subseteq \overline{W}, I \cup J \in \mathcal{I}(G)\}$.

Example 1. Consider the MISP on the graph in Figure 1. An exact BDD representation of the feasible set is given next to the graph, where arc (u, v) is solid or dashed if it is labelled as 1 or 0, respectively. Assigning arc costs as described above yields a longest path with value 3 in the BDD.

3 Exact BDD Compilation

A general method for creating exact BDDs for CSPs, known as *top-down compilation*, is presented in [5]. It consists of constructing layers $L_1, \ldots, L_n$ in that order, adding one node at a time at each layer. A node u is removed if the paths from r to u do not correspond to feasible partial solutions (*i.e.*, u is *infeasible*),

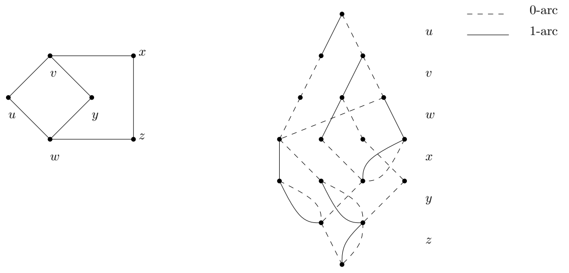


Fig. 1: Example of the exact BDD for a graph G.

and two nodes u, w are merged if all the partial solutions on paths from r to u and from r to w have the same set of feasible completions (*i.e.*, u and w are *equivalent*). The key of this technique is that the infeasibility and equivalence conditions are determined efficiently through the analysis of a *state* associated with each node, which is defined according to the problem constraints.

In order to apply the top-down exact BDD compilation algorithm for the MISP, we first establish a condition for identifying when two independent sets $I_1, I_2 \in \mathcal{I}(G[V_{j-1}])$ have the same set of feasible completions.

Theorem 1. *Given a graph $G = (V, E)$, a subset $\{v_1, \ldots, v_{j-1}\} = V_{j-1} \subseteq V$ of the vertices of G, and two independent sets $I_1, I_2 \subseteq \mathcal{I}(G[V_{j-1}])$,*

$$C(I_1 \mid \overline{V}_{j-1}) = C(I_2 \mid \overline{V}_{j-1}) \iff \overline{V}_{j-1} \setminus \cup_{v \in I_1} N(v) = \overline{V}_{j-1} \setminus \cup_{v \in I_2} N(v).$$

Proof. For $I \in \mathcal{I}(G[V_{j-1}])$, we must have $C(I \mid \overline{V}_{j-1}) = \mathcal{I}(G[\overline{V}_{j-1} \setminus \cup_{v \in I_1} N(v)])$, since $\overline{V}_{j-1} \setminus \cup_{v \in I_1} N(v)$ is exactly the set of remaining vertices in G that are independent of I. Conversely, suppose $\overline{V}_{j-1} \setminus \cup_{v \in I_1} N(v) \neq \overline{V}_{j-1} \setminus \cup_{v \in I_2} N(v)$. Without loss of generality, suppose there exists some $w \in \overline{V}_{j-1} \setminus \cup_{v \in I_1} N(v)$ that is not in $\overline{V}_{j-1} \setminus \cup_{v \in I_2} N(v)$. Then, $w \in C(I_1 \mid \overline{V}_{j-1})$ but $w \notin C(I_1 \mid \overline{V}_{j-1})$, hence $\{v\} \cup I_1$ is an independent set while $\{v\} \cup I_2$ is not, concluding the proof. $\square$

Let $E(I \mid \overline{V}_{j-1}) := \overline{V}_{j-1} \setminus \cup_{v \in I} N(v)$ be the set of *eligible* vertices of $I \in \mathcal{I}(G[V_{j-1}])$; *i.e.*, the vertices $v \in \overline{V}_{j-1}$ for which $I \cup \{v\}$ is an independent set. According to Theorem 1, we can directly use the eligible vertex set as the *state* for each node in order to perform the top-down compilation. Namely, since two independent sets $I_1, I_2 \in \mathcal{I}(G[V_j])$ with the same set of feasible completions have $E(I_1 \mid \overline{V}_j) = E(I_2 \mid \overline{V}_j)$, we label a BDD node $u \in L_j$ with a state $E(u) = E(I \mid \overline{V}_j)$ for any $I \in \text{Sol}(B^+|_u)$, as they must be all equal for any such I. Hence, all paths from r to u correspond to partial solutions with the same set of feasible completions. A node u can only have an one-arc directed out of it if $v_j \in E(u)$ (*infeasibility*), and two nodes u and w are equivalent if $E(u) = E(w)$. These tests are *complete*, *i.e.*, they are necessary and sufficient to determine when a node is infeasible or two nodes are equivalent. Thus, as noted in [5], the top-down compilation using the tests above yields the reduced BDD with respect to a particular variable ordering.

Algorithm 1 Top-Down BDD Compilation for MISP

```
1: Let $L_1 = \{r\}, E(r) = V, V_0 = \emptyset$
2: for $j = 1$ to $n$ do
3:     Choose vertex $v_j \notin V_{j-1}$ and let $V_j := V_{j-1} \cup \{v_j\}, L_{j+1} := \emptyset$
4:     for all $u \in L_j$ do
5:         if $\exists\, w \in L_{j+1}$ with $E(w) = E(u)\backslash\{v_j\}$ then
6:             add arc $(u, w)$ with $d_{u,w} = 0$
7:         else
8:             add node $w$ to $L_{j+1}$ with $E(w) = E(u)\backslash\{v_j\}$ and arc $(u, w)$ with $d_{u,w} = 0$
9:         if $v_j \in E(u)$ then
10:            if $\exists\, w \in L_{j+1}$ with $E(w) = E(u)\backslash(\{v_j\} \cup N(v_j))$ then
11:                add arc $(u, w)$ with $d_{u,w} = 1$
12:            else
13:                add node $w$ to $L_{j+1}$ with $E(w) = E(u)\backslash(\{v_j\} \cup N(v_j))$
14:                add arc $(u, w)$ with $d_{u,w} = 1$
```

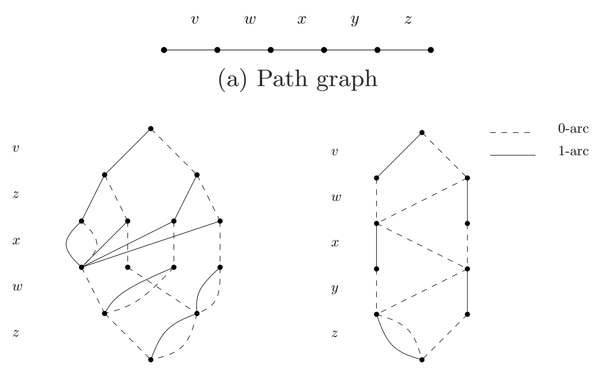
The top-down compilation for the MISP is outlined in Algorithm 1, which is a specialization of the procedure presented in [5]. We start with the root r of B having $E(r) = V$, since $\text{Sol}(B^+|_{L_1}) = \emptyset$ and hence all vertices in G are eligible. The construction is then performed layer by layer: For each node $u \in L_j$, we compute the state of the nodes corresponding to the zero-arcs and the one-arcs extensions of u, according to $E(u)$. If there exists some node $w \in L_{j+1}$ with the same state, we add an arc (u, w) to the BDD. Otherwise, we create a new node w in L_{j+1} and add an arc (u, w). The state of the zero-arc and one-arc extensions of u can be shown to be $E(u) \setminus \{u\}$ and $E(u) \setminus (\{u\} \cup N(u))$, respectively.

4 Variable Ordering for Exact BDD Compilation

The order of variables plays a key role in the size of exact BDDs. The impact of different orderings can be substantial, as shown in Figure 2. The example demonstrates two orderings for the graph presented in Figure 2a. The first ordering is constructed by alternating between the endpoints of the path, yielding a BDD of width 4 as depicted in Figure 2b. If vertices are taken according to the path order, the exact BDD has half the width, as presented in Figure 2c.

An *optimal* ordering minimizes the size of the exact BDD representing a given set of solutions. Previous works have focused on generating procedures to find optimal variable orderings for general BDDs (*e.g.*, [10]). It was shown in [6] that improving a given variable ordering, in terms of reducing the size of an BDD, is in general an NP-hard problem.

In this section we analyze variable orderings for the BDD representing the family of independent sets of a problem. We first examine particular classes of graphs, namely cliques, paths, and trees. We establish polynomial bounds on the widths (and hence size) of the exact BDDs with respect to the graph size. This is achieved by providing an ordering of the vertices that forces the width to be within a certain bound. Finally, we discuss the width for general graphs.



(a) Path graph

(b) Width 4

(c) Width 2

Fig. 2: Graph and exact BDD for two different orderings.

Let $S(L_j)$ be the set of states on nodes in L_j, $S(L_j) = \cup_{u \in L_j} E(u)$. To bound the width of a given layer j, we need only count the number of states that may arise from independent sets on $\{v_1, \ldots, v_{j-1}\}$. This is because each layer will have one and only one node for each possible state, and so there is a one-to-one correspondence between the number of states and the size of a layer. We now show the following Theorems.

Theorem 2. *Let $G = (V, E)$ be a clique. Then, for any ordering of the vertices, the width of the exact reduced BDD will be 2.*

Proof. Consider any layer j. The only possible independent sets on $\{v_1, \ldots, v_{j+1}\}$ are $\emptyset$ or $\{v_i\}, i = 1, \ldots, j-1$. For the former, $E(\emptyset \mid \{v_j, \ldots, v_n\}) = \{v_j, \ldots, v_n\}$ and for the latter, $E(\{v_i\} \mid \{v_j, \ldots, v_n\}) = \emptyset$, establishing the bound. □

Theorem 3. *Let $G = (V, E)$ be a path. Then, there exists an ordering of the vertices for which the width of the exact reduced BDD will be 2.*

Proof. Let the ordering of the vertices be given by the positions in which they appear in the path. Consider any layer j. Of the remaining vertices in G, namely $\{v_j, \ldots, v_n\}$, the only vertex with any adjacencies to $\{v_1, \ldots, v_{j-1}\}$ is v_j. Therefore, for any independent set $I \subseteq \{v_1, \ldots, v_{j-1}\}$, $E(I \mid \overline{V}_{j-1})$ will either be $\{v_j, \ldots, v_n\}$ (when $v_{j-1} \notin I$) and $\{v_{j+1}, \ldots, v_n\}$ (when $v_{j-1} \in I$). Therefore there can be at most 2 states in any given layer. □

Theorem 4. *Let $G = (V, E)$ be a tree. Then, there exists an ordering of the vertices for which the width of the exact reduced BDD will be no larger than n, the number of vertices in G.*

Proof. We proceed by induction on n. For the base case, a tree with 2 vertices is a path, which we already know has width 2. Now let T be a tree on n vertices. Any tree on n vertices contains a vertex v for which the connected components $C_1, \ldots, C_k$ created upon deleted v from T have sizes $|C_i| \leq \frac{n}{2}$ [16]. Each of these connected components are trees with fewer than $\frac{n}{2}$ vertices, so by induction, there exists an ordering of the vertices on each component C_i for which the resulting BDD B_i will have width $\omega(B_i) \leq \frac{n}{2}$. For component C_i, let $v_1^i, \ldots, v_{|C_i|}^i$ be an ordering achieving this width.

Let the final ordering of the vertices in T be $v_1^1, \ldots, v_{|C_1|}^1, v_1^2, \ldots, v_{|C_k|}^k, v$ which we use to create BDD B for the set of independent sets in T. Consider layer $\ell \leq n-1$ of B corresponding to vertex v_j^i. We claim that the only possible states in $S(\ell)$ are $s \cup C_{i+1} \cup \cdots \cup C_k$ and $s \cup C_{i+1} \cup \cdots \cup C_k \cup \{v\}$, for $s \in S^i(j)$, where $S^i(j)$ is the set of states in BDD B_i in layer j. Take any independent set on the vertices $I \subseteq \{v_1^1, \ldots, v_{|C_1|}^1, v_1^2, \ldots, v_{j-1}^i\}$. All vertices in I are independent of the vertices in $C_{i+1}, \ldots, C_k$, and so $E(I \mid \{v_j^i, \ldots, v_{|C_i|}^i\} \cup C_{i+1} \cup \cdots \cup C_k) \supseteq C_{i+1} \cup \cdots \cup C_k$. Now, consider $I_i = I \cap C_i$. I_i is an independent set in the tree induced on the variables in C_i and so it will correspond to some path in B_i from the root of that BDD to layer j, ending at some node u. The state s of node u contains all of the vertices $\{v_j^i, \ldots, v_{|C_i|}^i\}$ that are independent of all vertices in I_i. As $v_1^i, \ldots, v_{j-1}^i$ are the only vertices in the ordering up to layer ℓ in B that have adjacencies to any vertices in C_i, we see that the set of vertices in the state of I from component C_i are exactly s. Therefore, $E(I \mid \{v_j^i, \ldots, v_{|C_i|}^i\} \cup C_{i+1} \cup \cdots \cup C_k) \supseteq s \cup C_{i+1} \cup \cdots \cup C_k$. The only remaining vertex that may be in the state is v, finishing the claim. Therefore, as the only possible states on layer ℓ are $s \cup C_{i+1} \cup \cdots \cup C_k$ and $s \cup C_{i+1} \cup \cdots \cup C_k \cup \{v\}$, for $s \in S^i(j)$, we see that $\omega_\ell \leq \frac{n}{2} \cdot 2 = n$, as desired. The only layers remaining to bound is L_n: the only possible states on layer n are $\{v\}$ and $\emptyset$. □

Theorem 5. *Let $G = (V, E)$ be any graph. There exists an ordering of the vertices for which $\omega_j \leq F_{j+1}$, where F_k is the* k*th Fibonacci number.*

Theorem 5 provides a bound on the width of the exact BDD for any graph. The importance of this theorem goes further than the actual bound provided on the width of the exact BDD for any graph. First, it illuminates another connection between the Fibonacci numbers and the family of independent sets of a graph, as investigated throughout the Graph Theory literature (see for example [8, 11, 9, 18]). In addition to this theoretical consideration, the underlying principles in the proof provide insight into what heuristic ordering for the vertices in a graph could lead to BDDs with small width. We show in Section 6 that finding vertex orderings for which the exact BDD has small width correlates with the bound provided by relaxation BDDs using the same ordering. The ordering inspired by the underlying principle in the proof yields strong relaxation BDDs.

Proof (proof of Theorem 5).

Let $P = P^1, \ldots, P^k$, $P^i = \{v_1^1, \ldots, v_{i_k}^1\}$, be a *maximal path decomposition* of the vertices of G, where by a maximal path decomposition we mean a set of paths that partition V satisfying that v_1^i and $v_{i_k}^i$ are not adjacent to any vertices in $\cup_{j=i+1}^k P^j$. Hence, P^i is a maximal path (in that no vertices can be appended to the path) in the graph induced by the vertices not in the paths, $P^1, \ldots, P^{i-1}$.

Let the ordering of the vertices be given by $v_1^1, \ldots, v_{i_1}^1, v_1^2, \ldots, v_{i_k}^k$, *i.e.*, ordered by the paths and by the order that they appear on the paths. Let the vertices also be labeled, in this order, by $y_1, \ldots, y_n$.

We proceed by induction, showing that if layers L_j and L_{j+1} have widths ω_j and ω_{j+1}, respectively, then the width of layer L_{j+3} is bounded by $\omega_j + 2 \cdot \omega_{j+1}$, thereby proving that each layer L_j is bounded by F_{j+1} for every layer $j = 1, \ldots, n+1$, since $F_{j+3} = F_j + 2 \cdot F_{j+1}$.

First we show that L_4 has width bounded by $F_5 = 5$. We can assume that G is connected and has at least 4 vertices, so that P_1 has at least 3 vertices. $\omega_1 = 1$. Also, $\omega_2 = 2$, with layer L_2 having nodes u_1^2, u_2^2 arising from the partial solutions $I = \emptyset$ and $I = \{w_1\}$, respectively. The corresponding states will be $E(u_1^2) = V \backslash \{y_1\}$ and $E(u_2^2) = V \backslash (\ \{y_1\} \cup N(y_1)\ )$. Now, consider layer L_3. The partial solution ending at node $E(u_2^2)$ cannot have y_2 added to the independent set because y_2 does not appear in $E(u_2^2)$ since $y_2 \in N(w_1)$. Therefore, there will be exact 3 outgoing arcs from the nodes in L_2. If no nodes are combined on the third layer, there will be 3 nodes $u_i^3, i = 1,2,3$ with states $E(u_1^3) = V \backslash \{y_1, y_2\}$, $E(u_2^3) = V \backslash (\ \{y_1, y_2\} \cup N(y_2)\ )$, and $E(u_3^3) = V \backslash (\ \{y_1, y_2\} \cup N(y_1)\ )$. Finally, as P^1 has length at least 3, vertex y_3 is adjacent to y_2. Therefore, we cannot add y_3 under node u_2^3, so layer 4 will have width at most 5, finishing the base case.

Now let the layers of the partially constructed BDD be given by $L_1, \ldots, L_j, L_{j+1}$ with corresponding widths $\omega_i, i = 1, \ldots, j+1$. We break down into cases based on where y_{j+1} appears in the path that it belongs to in P, as follows.

Case 1: y_{j+1} is the last vertex in the path that it belongs to. Take any node $u \in L_{j+1}$ and its associated state $E(u)$. Including or not including y_{j+1} results in state $E(u) \backslash \{y_{j+1}\}$ since y_{j+1} is independent of all vertices $y_i, i \geq j+2$. Therefore, $\omega_{j+2} \leq \omega_{j+1}$ since each arc directed out of u will be directed at the same node, even if the zero-arc and the one-arc are present. And, since in any BDD $\omega_k \leq 2 \cdot \omega_{k-1}$, we have $\omega_{j+3} \leq 2 \cdot \omega_{j+2} \leq 2 \cdot \omega_{j+1} < \omega_j + 2 \cdot \omega_{j+1}$.

Case 2: y_{j+1} is the first vertex in the path that it belongs to. In this case, y_j must be the last vertex in the path that it belongs to. By the reasoning in Case 1, it follows that $\omega_{j+1} \leq \omega_j$. In addition, we can assume that y_{j+1} is not the last vertex in the path that it belongs to because then we are in case 1. Therefore, y_{j+2} is in the same path as y_{j+1} in P. Consider L_{j+2}. In the worst case, each node in L_{j+1} has y_{j+1} in its state so that $\omega_{j+2} = 2 \cdot \omega_{j+1}$. But, any node arising from a one-arc will not have y_{j+2} in its state. Therefore, there are at most ω_{j+1} nodes in L_{j+2} with y_{j+2} in their states and at most ω_{j+1} nodes in L_{j+2} without y_{j+2} in their states. For the set of nodes without y_{j+2} in their states, we cannot make a one-arc, showing that $\omega_{j+3} \leq \omega_{j+2} + \omega_{j+1}$. Therefore, we have $\omega_{j+3} \leq \omega_{j+1} + \omega_{j+2} \leq 3 \cdot \omega_{j+1} \leq \omega_j + 2 \cdot \omega_{j+1}$.

Case 3: y_{j+1} is not first or last in the path that it belongs to. As in case 2, $\omega_{j+1} \leq 2 \cdot \omega_j$, with at most ω_j nodes on layer L_{j+1} with w_{j+2} in it's corresponding state label. Therefore, L_{j+2} will have at most ω_j more nodes in it than layer L_{j+1}. As the same thing holds for layer L_{j+3}, in that it will have at most ω_{j+1} more nodes in it than layer L_{j+2}, we have $\omega_{j+3} \leq \omega_{j+2} + \omega_{j+1} \leq \omega_{j+1} + \omega_j + \omega_{j+1} = \omega_j + 2 \cdot \omega_{j+1}$, as desired, and finishing the proof. $\square$

We note here that using instance `C2000.9` from the benchmark set discussed in Section 6, a maximal path decomposition ordering of the vertices yields widths approximately equal to the Fibonacci numbers, as seen in Table 1.

Table 1: Widths of Exact BDD for `C.2000.9`

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\cdots$
w_j	1	2	3	5	8	13	21	31	52	65	117	182	299	481	624	$\cdots$
$Fib(j+1)$	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	$\cdots$

5 Variable Ordering for Relaxation BDDs

In this section we provide heuristic orderings for the vertices to be used during the top-down compilation of relaxation BDDs. These orderings are based on the Theorems proved in Section 4, with the idea that by examining simple structured problems, we can gain intuition as to what is controlling the width of the exact BDD for general graphs, hopefully yielding tighter upper bounds. First, we describe an alteration to the top-down exact BDD compilation for the purpose of generating relaxation BDDs, and then present the heuristic orderings.

A method for generating relaxation BDDs is developed in [5]. It alters the top-down exact compilation algorithm by restricting the maximum width of the BDD according to an input parameter W. This is done as follows. After a layer L_j is built, the method verifies if its width exceeds W. If so, nodes are merged based on some heuristic choice. Along with merging the nodes, a merging operation defined on the states of the nodes that are to be merged must be defined so that the feasible solutions are preserved during the construction of the remaining layers. For the MISP, the Theorem below defines a proper merging operation.

Theorem 6. *Setting the state of the merged nodes as the union of their original states ensures that a relaxation BDD is created.* $\square$

We now present the following heuristic orderings.

Maximal Path Decomposition (MPD). As show in Theorem 5, such an ordering yields an exact BDD with width bounded by the Fibonnaci numbers, yielding a theoretical worst-case bound on the width for any instance. This ordering can be pre-computed in worst-case time complexity $\mathcal{O}(|V|+|E|)$. We note that different maximal path decompositions may yield different sized BDDs.

Minimum Number of States (MIN). In this ordering, we select the next vertex in the BDD as the vertex which appears in the fewest states of the layer we are currently building. The driving force behind the proof of Theorem 5 is that when constructing a layer, if a vertex does not belong to the state of a node on a previous layer, we cannot include this vertex: *i.e.* we cannot add a one-arc, only the zero-arc. This suggests that selecting a variable appearing the fewest number of times in the states on a layer will yield a small width BDD. The worst-case time complexity to perform this selection is $\mathcal{O}(W|V|)$ per layer.

k-Look Ahead Ordering (kLA). This ordering can be employed for any COP. In *1LA*, after selecting the first j vertices and constructing the top $j+1$ layers, the next chosen vertex is the one that yields the smallest width for layer $j+2$ if it were selected next. This procedure can be generalize for arbitrary $k < n$ by considering subsets of yet to be selected vertices. The worst case running time for selecting a vertex can be shown to be $\mathcal{O}(\binom{n}{k} \cdot W|V|^2 \log |W|)$ per layer.

6 Experimental Results

Our experiments focus on the complement graphs of the well-known DIMACS problem set for the Maximum Clique Problem, which can obtained by accessing http://dimacs.rutgers.edu/Challenges/. The experiments ran on an Intel Xeon E5345 with 8 GB RAM. The BDD was implemented in C++.

6.1 Exact BDDs for Trees

The purpose of the first set of experiments is to demonstrate empirically that variable orderings potentially play a key role in the width of exact BDDs representing combinatorial optimization problems. To this end, we have selected a particular graph structure, namely trees, for which we can define an ordering yielding a polynomial bound on its width (Theorem 4). We then compare the ordering that provides this bound with a set of randomly generated orderings. We also compare with the MPD heuristic, which has a known bound for general graphs according to Theorem 5. The trees were generated from the benchmark problems `C125.9, keller4, c-fat100-1, p_hat300-1, brock200_1`, and `san200_0.7_1` by selecting 5 random trees each on 50 vertices from these graphs. The tree-specific ordering discussed in Theorem 4 is referred to as the *CV* (due to the computation of cut-vertices in the corresponding proof). We generated exact BDDs using 100 uniform-random orderings for each instance, and report the minimum, average, and maximum obtained widths.

The results are shown in Table 2. In all cases, none of the 100 random orderings yielded exact BDDs with width smaller than the ones generated from the CV or MPD orderings. Moreover, the average was consistently more than an order of magnitude worse than either of the structured orderings. This confirms that investigating variable orderings can have a substantial effect on the width of the exact BDDs produced for independent set problems. In addition, we see that also across all instances, the CV ordering, that is specific to trees, outperforms the MPD ordering that can be applied to general graphs, suggesting that investigating orderings specific to particular classes of instances can also have a positive impact on the width of exact BDDs.

Table 2: Random Trees

Instance	Min	Avg	Max	CV	MPD	Instance	Min	Avg	Max	CV	MPD
brock200_1.t-1	2336	22105.1	116736	16	160	C125.9.t-1	768	7530.72	24576	12	228
brock200_1.t-2	672	8532.92	86016	16	312	C125.9.t-2	1600	19070	131072	12	528
brock200_1.t-3	672	7977.92	28608	8	120	C125.9.t-3	1024	8348.04	30720	12	288
brock200_1.t-4	2880	17292.9	67200	16	132	C125.9.t-4	736	4279.62	16704	16	312
brock200_1.t-5	1200	12795.2	55680	8	54	C125.9.t-5	480	18449.3	221184	16	120
c-fat200-1.t-1	896	17764.3	221184	8	112	keller4.t-1	952	9558.76	115200	8	248
c-fat200-1.t-2	1152	10950.9	55040	16	144	keller4.t-2	768	8774.12	71680	12	444
c-fat200-1.t-3	2048	23722.6	150528	10	72	keller4.t-3	2688	16942.1	74240	10	40
c-fat200-1.t-4	624	5883.96	46656	12	180	keller4.t-4	2048	14297.8	77440	16	368
c-fat200-1.t-5	864	7509.66	27648	10	480	keller4.t-5	720	11401.8	73728	8	288
p_hat300-1.t-1	792	15149.3	54720	10	200	san200_0.7_1.t-1	1920	22771.2	139776	10	28
p_hat300-1.t-2	1280	14618.5	86016	16	192	san200_0.7_1.t-2	1024	7841.42	44160	12	92
p_hat300-1.t-3	624	11126.6	69120	12	138	san200_0.7_1.t-3	768	8767.76	36864	8	88
p_hat300-1.t-4	1152	13822.9	73984	16	74	san200_0.7_1.t-4	960	9981.28	43008	16	84
p_hat300-1.t-5	1536	16152	82944	14	160	san200_0.7_1.t-5	1536	9301.92	43008	12	288

6.2 Exact BDD Width versus Relaxation BDD Bound

The second set of experiments aims at providing an empirical evidence to the main hypothesis considered in this paper. Namely, that a problem instance with a smaller exact BDD results in a relaxation BDD that yields a tighter bound. The instances in this test were generated as follows. We first selected 5 instances from the DIMACS benchmark: `brock200_1`, `gen200_p.0.9_55`, `keller4`, `p_hat300-2`, and `san200_0.7_1`. Then, we uniformly at random extracted 5 connected induced subgraphs with 50 vertices for each instance, which is approximately the largest graph size that the exact BDD can be built within our memory limits.

The tests are described next. For each instance and all orderings MPD, MIN, random, and 1LA, we collected the width of the exact BDD and the bound obtained by a relaxation BDD with a maximum width of 10 (the average over 100 orderings for the random procedure). This corresponds to sampling different exact BDD widths and analyzing their respective bounds, since distinct variables orderings may yield BDDs with very different exact widths.

Figure 3 presents a scatter plot of the derived upper bound as a function of the exact widths in log-scale, also separated by the problem class from which the instance was generated. Analyzing each class separately, we observe that the bounds and width increase proportionally, reinforcing our hypothesis. In particular, this proportion tends to be somewhat constant, that is, the points tend to a linear curve for each class. We notice that this shape has different slopes according to the problem class, hence indicating that the effect of the width might be more significant for certain instances.

In Figure 4 we plot the bound as a function of the exact width for a single random instance extracted from `san200_0.7_1`. In this particular case, we applied a procedure that generated 1000 exact BDDs with a large range of widths: the minimum observed BDD width was 151 and the maximum was 27684, and the widths were approximately uniformly distributed in this interval. We then computed the corresponding upper-bounds for a relaxed BDD, constructed using the orderings described above, with width 10. The width is given in a log-scale.

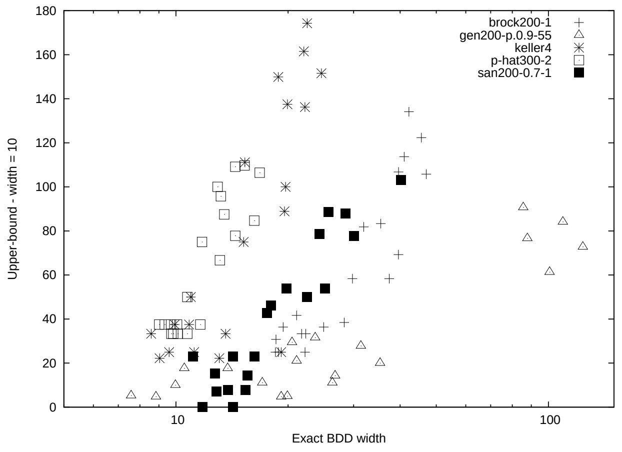


Fig. 3: Bound of relaxation BDD vs. exact BDD width.

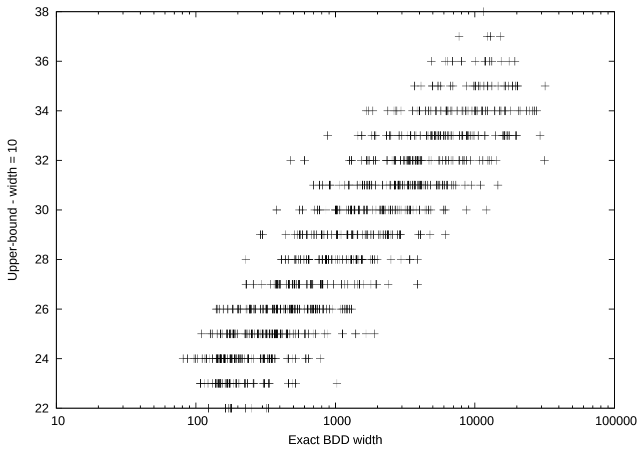


Fig. 4: Bound of relaxation BDD vs. exact BDD width for `san200_0.7_1`.

The Figure also shows a strong correlation between the width and the obtained bound, analogous to the previous set of experiments. A similar behavior is obtained if the same chart is plotted for other instances.

6.3 Relaxation Bounds

We now report the upper bound provided by the relaxation BDD for the original benchmark set, considering all heuristic orderings described in Section 5 for maximum widths 100, 500, and 1000. In addition, we generate 100 random orderings generated uniformly at random, denoted here by RAND, and the bound reported

Table 3: Benchmark Problems Relaxations

Maximum Width		100				500				1000					1000
Instance	OPT	MIN	MAX	RAND	1LA	MIN	MAX	RAND	1LA	MIN	MAX	RAND	1LA	CPLEX(1 minute)	MIN
C1000.9.clq	68	261	419	585.42	259	244	394	528.25	241	240	384	506.63	238	**221.78**	240
C125.9.clq	34	46	55	71.68	44	45	52	64.51	42	43	50	61.78	41	**41.2846**	43
C2000.5.clq	16	153	353	368.34	152	121	249	252.27	120	110	218	218	110	1.00E+75	**110**
C2000.9.clq	77	480	829	1170.91	479	447	788	1055.26	447	436	767	1012.4	433	1.00E+75	**436**
C250.9.clq	44	80	107	144.84	78	74	99	130.46	73	72	98	125.21	72	**70.9322**	72
C4000.5.clq	18	281	708	736.31	280	223	497	504.46	223	202	429	435.31	203	1.00E+75	**202**
C500.9.clq	57	142	215	291.48	142	134	203	262.57	133	132	198	251.8	131	**123.956**	132
gen200_p0.9_44.clq	44	62	84	115.69	62	61	79	103.98	59	59	78	99.78	56	**44**	59
gen200_p0.9_55.clq	55	67	88	116.39	65	63	84	104.88	62	61	81	100.57	59	**55**	61
gen400_p0.9_55.clq	55	100	168	233.15	100	99	161	210.21	96	94	156	201.84	94	**55**	94
gen400_p0.9_65.clq	65	112	168	233.63	110	105	161	210.55	105	103	159	202.11	101	**65**	103
gen400_p0.9_75.clq	75	118	170	234.23	118	109	164	211.2	109	108	158	202.73	105	**75**	108
brock200_1.clq	21	42	64	72.12	41	36	54	58.61	36	34	50	54.01	35	38.9817	**34**
brock200_2.clq	12	22	35	35.6	22	17	24	24.68	18	16	22	21.69	16	22.3764	**16**
brock200_3.clq	15	28	48	48.87	29	24	36	36.22	25	23	33	32.39	23	28.3765	**23**
brock200_4.clq	17	32	53	56.61	32	29	42	43.32	27	26	37	39.12	25	31.5437	**26**
brock400_1.clq	27	72	127	145.81	71	63	108	118.75	63	60	102	109.32	61	67.2201	**60**
brock400_2.clq	29	75	128	147.35	72	63	107	119.47	61	61	101	110.16	60	67.9351	**61**
brock400_3.clq	31	72	127	146.19	73	64	109	118.63	64	60	102	109.12	60	67.4939	**60**
brock400_4.clq	33	70	129	146.43	71	63	110	119.54	63	63	106	109.59	61	67.3132	**63**
brock800_1.clq	23	99	204	222.01	100	85	160	168.39	86	79	145	151.21	78	136.103	**79**
brock800_2.clq	24	101	201	224.38	100	86	162	170.65	85	79	145	153.29	79	136.538	**79**
brock800_3.clq	25	101	203	222.61	100	84	164	169.05	84	81	149	151.31	79	130.832	**81**
brock800_4.clq	26	101	205	223.41	100	84	161	169.81	84	80	145	152.66	78	132.696	**80**
c-fat200-1.clq	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
c-fat200-2.clq	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
c-fat200-5.clq	58	58	58	58	58	58	58	58	58	58	58	58	58	61.6953	**58**
c-fat500-1.clq	14	14	15	16.62	14	14	14	14	14	14	14	14	14	230.513	**14**
c-fat500-10.clq	126	126	126	126	126	126	126	126	126	126	126	126	126	246	**126**
c-fat500-2.clq	26	26	26	26	26	26	26	26	26	26	26	26	26	240	**26**
c-fat500-5.clq	64	64	64	64	64	64	64	64	64	64	64	64	64	244.5	**64**
hamming10-2.clq	512	512	512	892.69	515	512	512	871.68	512	512	512	862.99	512	512	512
hamming10-4.clq	40	106	91	456.63	105	96	76	385.13	93	79	72	359.76	79	206.047	**79**
hamming6-2.clq	32	32	32	37.01	32	32	32	34.03	32	32	32	33.28	32	32	32
hamming6-4.clq	4	4	4	5.98	4	4	4	4	4	4	4	4	4	5.33333	**4**
hamming8-2.clq	128	128	128	194.42	128	128	128	184.51	128	128	128	180.71	128	128	128
hamming8-4.clq	16	20	21	62.23	19	18	18	45.66	18	17	17	40.56	17	**16**	17
johnson16-2-4.clq	8	11	11	38.75	11	9	9	29.24	9	8	8	25.64	8	8	8
johnson32-2-4.clq	16	40	35	250.07	42	38	29	215.06	39	35	25	202.36	40	**16**	35
johnson8-2-4.clq	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
johnson8-4-4.clq	14	14	15	24.57	14	14	14	19.82	14	14	14	18.54	14	14	14
keller4.clq	11	19	22	43.38	18	16	17	31.24	16	15	16	27.54	15	**14.75**	15
keller5.clq	27	58	98	280.74	59	56	77	225.75	55	48	72	207.08	49	**32.875**	48
keller6.clq	59	171	417	1503.26	174	142	332	1277.98	144	123	307	1197.76	125	1.00E+75	**123**
MANN_a27.clq	126	142	138	327.2	135	140	137	318.93	137	139	137	315.25	136	**133.331**	139
MANN_a45.clq	345	371	365	954.51	366	368	362	942.45	363	368	362	937.06	365	**357.162**	368
MANN_a81.clq	1100	1154	1143	3186.21	1141	1150	1143	3166.06	1143	1148	1143	3158.78	1141	**1131.82**	1148
MANN_a9.clq	16	18	18	27.21	17	16	16	23.9	16	16	16	22.88	16	17	**16**
p_hat1000-1.clq	10	47	86	88.73	48	35	52	52.71	36	31	43	43.37	31	413.5	**31**
p_hat1000-2.clq	46	130	210	225.57	129	116	171	178.1	112	112	159	163.47	108	376.5	**112**
p_hat1000-3.clq	68	202	324	383.76	197	187	286	322.62	179	179	272	302.07	175	245.674	**179**
p_hat1500-1.clq	12	68	136	139.02	68	51	83	83.08	51	46	69	68.33	45	1.00E+75	**46**
p_hat1500-2.clq	65	199	344	357.01	193	176	285	286.03	174	168	267	263.95	163	1.00E+75	**168**
p_hat1500-3.clq	94	298	511	594.04	296	277	452	502.22	270	272	433	470.91	266	1.00E+75	**272**
p_hat300-1.clq	8	17	27	26.05	18	14	16	15.89	14	12	13	13.39	12	18.2278	**12**
p_hat300-2.clq	25	48	64	66.46	45	42	51	52.29	40	40	48	47.83	39	**35.2878**	40
p_hat300-3.clq	36	70	99	114.66	67	65	89	95.93	61	62	84	89.86	60	**55.2598**	62
p_hat500-1.clq	9	28	45	45.33	27	21	28	27.3	21	18	23	22.7	19	158	**18**
p_hat500-2.clq	36	77	112	116.55	72	69	92	92.8	64	66	84	85.54	63	160.25	**66**
p_hat500-3.clq	50	111	172	195.67	109	106	155	165.35	102	104	147	154.88	99	**90.7331**	104
p_hat700-1.clq	11	36	62	63.27	36	27	39	37.83	27	24	31	31.33	24	272.5	**24**
p_hat700-2.clq	44	101	155	163.03	99	90	128	130.39	88	85	118	120.19	83	272.5	**85**
p_hat700-3.clq	62	153	234	272.83	147	142	208	230.14	141	137	198	215.93	134	160.333	**137**
san1000.clq	15	28	184	202.02	26	21	101	104.09	19	19	78	79.84	19	462.5	**19**
san200_0.7_1.clq	30	32	66	73.67	31	30	57	60.3	30	30	52	55.37	30	30	30
san200_0.7_2.clq	18	23	58	71.76	21	20	48	56.2	20	19	46	50.23	18	**18**	19
san200_0.9_1.clq	70	71	86	118.89	70	70	82	108.56	70	70	81	105.13	70	70	70
san200_0.9_2.clq	60	68	86	116.48	64	64	83	105.39	60	60	81	101.05	60	60	60
san200_0.9_3.clq	44	57	84	115	54	55	78	103.23	53	51	77	99	52	**44**	51
san400_0.5_1.clq	13	17	66	69.02	18	14	35	35.6	14	13	28	28.31	13	13	13
san400_0.7_1.clq	40	50	142	160.35	51	46	127	136.08	43	42	119	126.86	41	**40**	42
san400_0.7_2.clq	30	44	129	147.55	45	38	108	119.96	39	37	103	109.84	35	**30**	37
san400_0.7_3.clq	22	36	118	137.72	38	29	98	108.29	31	29	91	97.98	29	**22**	29
san400_0.9_1.clq	100	117	175	236.22	118	109	169	214.05	108	108	164	205.73	108	**100**	108
sanr200_0.7.clq	18	34	58	63	36	31	46	49.56	32	30	44	45.18	29	34.5339	**30**
sanr200_0.9.clq	42	67	86	114.78	66	63	83	103.25	60	61	80	98.89	61	**59.5252**	61
sanr400_0.5.clq	13	40	70	73.32	39	33	50	50.5	31	29	45	43.73	29	43.1544	**29**
sanr400_0.7.clq	21	64	115	128.44	64	55	96	101.06	54	52	89	91.69	52	62.078	**52**

is obtained by taking the average over the 100 generated orderings. The average compilation time for maximum width 100, 500 and 1000 were 0.21, 1.49, and 3.01 seconds, respectively, for the MIN ordering (which was similar to RAND and MPD), while the average time for maximum width 100, 500, and 1000 were 65.01, 318.68, and 659.02, respectively, for the 1LA ordering. For comparison purposes, we have also included the upper bound obtained by considering the IP formulation of the MISP, since this corresponds to a well-known bounding technique for general domains. We ran these instances with CPLEX 12.2 with default settings and took the resulting bound obtained after the root node was computed. We impose a time limit of 60 seconds so that the results were comparable to the MIN ordering with width 1000 since the longest time to create any relaxation BDD with these parameters was `C.4000.5`, which took 50.42 seconds.

The results are presented in Table 3. We report for each instance the optimal or the best known feasible solution and the bounds, where *CPLEX* is the bound obtained by the root node relaxation using CPLEX (the symbol $+\infty$ indicates that a bound was not obtained in the 60 seconds time-limit). By first comparing the results obtained between orderings, we see that the MIN ordering and the general purpose 1LA heuristic provide the best bounds for most instances. We highlight here that the MIN and 1LA were the heuristics that provided the smallest BDD widths for the instances tested in Section 6.2. We note that MIN is generates BDDs an average of an order of magnitude faster than 1LA.

To compare the obtained bounds with CPLEX, we consider the *relative bound* measure, which is given by (upper bound/optimum). The average relative bound for CPLEX (omitting the instances for which CPLEX was unable to provide a bound) is given by 3.85, while for MIN and 1LA they are given by 2.34 and 2.32, respectively, for a width of 100; and 1.92 and 1.90, respectively, for a width of 1000 (the averages are not significantly different at the 5% level between MIN and 1LA). The average relative ordering for RAND was 5.51 and 4.25 for widths of 100 and 1000, respectively. This indicates that variable orderings are crucial to obtain tighter and relevant bounds, which showed to be particularly significant for larger instances when comparing with CPLEX, explaining the smaller average relative bound. We further observe that, since times were very small for the structured heuristics, the bounds obtained here can be improved using the general purpose bound improving procedures in [5].

7 Conclusion

In this paper we analyzed the impact of variable ordering on the quality of the relaxation provided by binary decision diagrams. We focus on the Maximum Independent Set Problem, providing theoretical bounds on the BDD width for general and particular classes of graphs. In addition, we utilize the developed theory to propose specific and general-purpose variable ordering heuristics. Experimental results indicate that there is a strong correlation between variable ordering heuristics that yield small-sized exact BDDs and the bounds obtained by relaxed BDDs that use these orderings.

References

1. S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C-27:509–516, 1978.
2. H. R. Andersen, T. Hadzic, J. N. Hooker, and P. Tiedemann. A constraint store based on multivalued decision diagrams. In C. Bessière, editor, *Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *Lecture Notes in Computer Science*, pages 118–132. Springer, 2007.
3. B. Becker, M. Behle, F. Eisenbrand, and R. Wimmer. BDDs in a branch and cut framework. In S. Nikoletseas, editor, *Experimental and Efficient Algorithms, Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA 05)*, volume 3503 of *Lecture Notes in Computer Science*, pages 452–463. Springer, 2005.
4. Markus Behle and Friedrich Eisenbrand. 0/1 vertex and facet enumeration with bdds. In *ALENEX*. SIAM, 2007.
5. David Bergman, Willem Jan van Hoeve, and John N. Hooker. Manipulating mdd relaxations for combinatorial optimization. In Tobias Achterberg and J. Christopher Beck, editors, *CPAIOR*, volume 6697 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2011.
6. Bollig and Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEETC: IEEE Transactions on Computers*, 45, 1996.
7. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.
8. Neil J. Calkin and Herbert S. Wilf. The number of independent sets in a grid graph. *SIAM J. Discrete Math.*, 11(1):54–60, 1998.
9. Martin E. Dyer, Alan M. Frieze, and Mark Jerrum. On counting independent sets in sparse graphs. *SIAM J. Comput.*, 31(5):1527–1541, 2002.
10. Rudiger Ebendt, Wolfgang Gunther, and Rolf Drechsler. An improved branch and bound algorithm for exact BDD minimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(12):1657–1663, 2003.
11. Florence Forbes and Bernard Ycart. Counting stable sets on cartesian products of graphs. *Discrete Mathematics*, 186(1-3):105–116, 1998.
12. T. Hadzic and J. N. Hooker. Postoptimality analysis for integer programming using binary decision diagrams, presented at GICOLAG workshop (Global Optimization: Integrating Convexity, Optimization, Logic Programming, and Computational Algebraic Geometry), Vienna. Technical report, Carnegie Mellon University, 2006.
13. T. Hadzic and J. N. Hooker. Cost-bounded binary decision diagrams for 0-1 programming. Technical report, Carnegie Mellon University, 2007.
14. T. Hadzic, J. N. Hooker, B. O'Sullivan, and P. Tiedemann. Approximate compilation of constraints into multivalued decision diagrams. In P. J. Stuckey, editor, *Principles and Practice of Constraint Programming (CP 2008)*, volume 5202 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2008.
15. S. Hoda, W.-J. van Hoeve, and John N. Hooker. A systematic approach to mdd-based constraint programming. In *Proceedings of the 16th International Conference on Principles and Practices of Constraint Programming*, Lecture Notes in Computer Science. Springer, 2010.
16. C. Jordan. Sur les assemblages de lignes. *J. Reine Angew Math*, 70:185–190, 1869.
17. C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal*, 38:985–999, 1959.
18. Yufei Zhao. The number of independent sets in a regular graph. *Combinatorics, Probability & Computing*, 19(2):315–320, 2010.