Duplex Encoding of Staircase At-Most-One Constraints for the Antibandwidth Problem

Katalin Fazekas[1] (✉), Markus Sinnl[2], Armin Biere[1], and Sophie Parragh[2]

[1] Institute for Formal Models and Verification
katalin.fazekas@jku.at, armin.biere@jku.at
Johannes Kepler University, Linz, Austria
[2] Institute of Production and Logistics Management
markus.sinnl@jku.at, sophie.parragh@jku.at
Johannes Kepler University, Linz, Austria

Abstract. Decision and optimization problems can be tackled with different techniques, such as Mixed Integer Programming, Constraint Programming or SAT solving. An important ingredient in the success of each of these approaches is the exploitation of common constraint structures with specialized (re-)formulations, encodings or other techniques. In this paper we present a new linear SAT encoding using binary decision diagrams over multiple variable orders as intermediate representation of a special form of constraints denoted as staircase at-most-one-constraints. The use of these constraints is motivated by recent work on the antibandwidth problem, where an iterative solution procedure using feasibility-mixed integer programs based on such constraints was most effective. In a computational study we compare the effectiveness of our new encoding against traditional SAT-encodings for staircase at-most-one-constraints. Additionally we compare against previous exact solution methods for the antibandwidth problem, such as a constraint programming approach and the one based on feasibility-mixed integer programs.

1 Introduction

An important ingredient in the success of computational approaches, such as Mixed Integer Programming (MIP), Constraint Programming (CP) or propositional satisfiability solving (SAT), for solving optimization and decision problems is the exploitation of common constraint structures with specialized encodings, (re-)formulations or other techniques (see e.g. [1–3]).

In this paper we present a new and specialized SAT encoding of problems where an at-most-one constraint slides over a sequence of Boolean variables. We denote this special case of sliding sequence constraints [4–7] as *staircase at-most-one constraint* (SCAMO) and illustrate the reason for this name with the following example.

Example 1. Given a sequence of variables $X = \langle x_1\, x_2 \cdots x_{10} \rangle$, the staircase at-most-one constraint set of width 4 is the following formula:

$$
\begin{array}{ccccccccccccccccccc}
x_1 & + & x_2 & + & x_3 & + & x_4 & & & & & & & & & & & \leq 1 \wedge \\
& & x_2 & + & x_3 & + & x_4 & + & x_5 & & & & & & & & & \leq 1 \wedge \\
& & & & x_3 & + & x_4 & + & x_5 & + & x_6 & & & & & & & \leq 1 \wedge \\
& & & & & & x_4 & + & x_5 & + & x_6 & + & x_7 & & & & & \leq 1 \wedge \\
& & & & & & & & x_5 & + & x_6 & + & x_7 & + & x_8 & & & \leq 1 \wedge \\
& & & & & & & & & & x_6 & + & x_7 & + & x_8 & + & x_9 & \leq 1 \wedge \\
& & & & & & & & & & & & x_7 & + & x_8 & + & x_9 & + \; x_{10} \leq 1.
\end{array}
$$

This research is motivated by recent work [8] of the second author on the *antibandwidth problem* (ABP). The ABP is a *graph labeling problem* (see e.g. [9] for more on such problems) where the goal is to maximize the smallest difference between labels of neighbouring nodes. It has various applications, such as scheduling [10], obnoxious facility location [11], radio frequency assignment [12] and map-coloring [13]. It has been studied from a theoretical point of view (see e.g. [14–19]), and several heuristics and metaheuristics (e.g. [20–23]) have been designed for it. In [21], aside from a metaheuristic, also a MIP approach was presented to solve the ABP exactly.

In [8] new MIP formulations were presented, and based on one of them, an iterative solution procedure, which repeatedly solved feasbility-MIPs, was designed. For a given number k, these MIPs encode the question whether there exists a solution with antibandwidth greater than k. This iterative procedure actually proved to be the most effective one in the computational study of [8].

Our proposed encoding can be used for more difficult problem structures than the one given in Example 1. In the ABP, for example, the difference of labels of neighbouring nodes is restricted by combining two SCAMO constraints on two sequences of variables. Aside from the ABP (and other labeling problems), the SCAMO constraints can potentially be used in many further application contexts, such as scheduling problems (see e.g. [24–26]) or in staff rostering [27, 28] and car sequencing problems [29, 30], when at most one variable is allowed to take a given value in every sequence of variables.

As at-most-one constraints are ubiquitous in applications of SAT they are featured prominently in the literature, see e.g. [31–36]. They are forming a special case of cardinality constraints [37–39], which in turn are instances of Pseudo-Boolean constraints [40–43] and thus 0/1 integer linear programs. Encoding constraints (for an overview see [36]) instead of handling them natively (as in [38]) allows to make full use of the power of SAT solving. For some applications mixed strategies [44] are better though. In practice, size is the most important criteria to evaluate such encodings, while at least in theory also propagation strength is considered. See [45] for a discussion of these trade-offs. In particular, the path based encoding of binary decision diagrams introduced in [45] has the goal to improve propagation. However, as the authors point out, it can not be used for encoding shared constraints, which is the main reason of the efficiency in our encoding. Thus we also provide a new set of benchmarks for which such sharing occurs naturally.

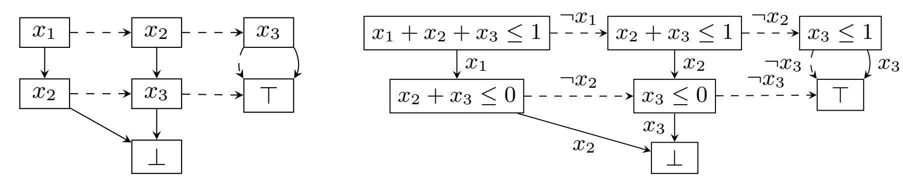
2 Preliminaries

A propositional formula in conjunctive normal form (CNF) consists of a set of clauses, where each clause C is a disjunction of literals, which are Boolean (also called 0/1) variables (e.g. x) or their negation ($\neg x$ or $1-x$). A truth assignment T maps truth values (0/1 values) to Boolean variables and can be represented by a set of consistent literals; it satisfies a literal ℓ (i.e. assigns value 1 to ℓ) if $\ell \in T$, and falsifies it (assigns value 0 to ℓ) if $\neg\ell \in T$, where $\neg\ell = \neg x$ if $\ell = x$ and $\neg\ell = x$ if $\ell = \neg x$. The satisfiability problem (SAT) for a formula in CNF asks whether there is a truth assignment such that all clauses contain at least one satisfied literal. A truth assignment satisfying a formula is also called a model.

An *at-most-one* (AMO) constraint is an expression of the form $\sum_{i=1}^{n} x_i \leq 1$, where $x_1, x_2, \ldots, x_n$ are Boolean variables. Similarly, we can formulate *at-most-zero* (AMZ) constraints (as $\sum_{i=1}^{n} x_i \leq 0$), which actually states that each variable must be false (i.e. assigned value 0). Further, an exactly-one (EO) constraint is an expression of the form $\sum_{i=1}^{n} x_i = 1$. Notice that we define and use these constraints over Boolean variables, but they are trivially extensible to literals.

A binary decision diagram (BDD, see e.g. [46, 47]) is a rooted, directed, acyclic graph with at most two leafs, labeled with $\bot$ (false or 0) and $\top$ (true or 1). Every non-leaf (also called nonterminal) node of a BDD is labeled with a Boolean variable and has exactly two outgoing edges (called *low* and *high* in [46]). In this paper we use BDDs to represent AMO and AMZ constraints. Figure 1a depicts an example BDD of an AMO constraint over variables x_1, x_2 and x_3. Each path from the root of the BDD that ends in the true leaf ($\top$) is a model of $x_1 + x_2 + x_3 \leq 1$. Whenever the low or high child (marked with dashed resp. solid line in Fig. 1) of a node labeled with variable x is taken, it means that x is assigned to be false (true respectively) on that path. Since all our BDDs represent AMO or AMZ constraints, we will depict them rather in an expanded form where each node contains the whole Boolean expression represented by the sub-graph starting from it, as it can be seen on Fig. 1b. To emphasize the decision variables of the nodes, we mark them explicitly on the edges. Further, beyond the non-terminal (i.e. non-leaf) nodes we distinguish non-unit nodes that are representing a constraint over more than one variable. For example, the BDD of Fig. 1b contains two leaf nodes ($\top$ and $\bot$), two unit nodes (over x_3) and three non-unit non-leaf nodes. The ordering of the variables appearing in BDDs is fixed (e.g. $x_1 < x_2 < x_3$ in Fig. 1), i.e. we use ordered BDDs (OBDD in short). Even though we merge isomorphic subtrees in our BDDs, they are not reduced because nodes with identical children are kept (see e.g. x_3 in Fig. 1). Thus we use partially reduced ordered BDDs (ROBDD) over multiple variable orders.

Given a graph $G = (V, E)$, a feasible solution to the antibandwidth problem consists of assigning each node $v \in V$ a unique label from the range $1, \ldots, |V|$. Given such a labeling f, the antibandwidth $AB_f(v)$ of a node v is defined as $\min\{|f(v) - f(v')| : \{v, v'\} \in E\}$, and the antibandwidth $AB_f(G)$ is defined as $\min\{AB_f(v) : v \in V\}$. The goal of the ABP is to find a labeling f^*, such that $f^* = \arg\max_{f \in \mathcal{F}(G)} AB_f(G)$, where $\mathcal{F}(G)$ denotes the set of all labelings of G.



(a) BDD of $(x_1 + x_2 + x_3 \leq 1)$ (b) Expanded BDD of $(x_1 + x_2 + x_3 \leq 1)$

Fig. 1: Different BDD representations of AMO constraint $(x_1 + x_2 + x_3 \leq 1)$.

We briefly discuss previous work [8] on which our new SAT solution is based. Let binary variables $x_i^\ell = 1$ if and only if vertex i is assigned label ℓ (i.e. $f_i = \ell$). For a given k, the question, whether there exists a solution with $\mathrm{AB}(G) \geq k+1$, can be formulated as MIP as follows. We will denote this formulation as $F_e(k)$.

$$\max \ 0$$

$$\sum_{i \in V} x_i^\ell = 1 \qquad \forall \ell \in \{1, \ldots, |V|\} \qquad \text{(Labels)}$$

$$\sum_{\ell \in \{1,\ldots,|V|\}} x_i^\ell = 1 \qquad \forall i \in V \qquad \text{(Vertices)}$$

$$\sum_{\lambda \leq \ell \leq \lambda+k} (x_i^\ell + x_{i'}^\ell) \leq 1 \qquad \forall \{i, i'\} \in E,\ 1 \leq \lambda \leq |V| - k \qquad (\text{Obj}_k)$$

$$x_i^\ell \in \{0,1\} \qquad \forall i \in V,\ \forall \ell \in \{1, \ldots, |V|\}$$

Constraints (Labels) make sure that each label is used only once and constraints (Vertices) ensure that each node $i \in V$ gets assigned one label. Thus, the solution encoded by these constraints corresponds to a labeling. Constraints (Obj_k) describe that for each edge $\{i, i'\}$, the labels f_i, $f_{i'}$ are not allowed to be within a range of k. Thus, any solution of the above constraints corresponds to a labeling with antibandwidth at least $k+1$. The iterative algorithm of [8] starts with a value of k obtained by a heuristic, which constructs a feasible labeling, and then iteratively solves $F_e(k)$ and increases k by one, until either $F_e(k)$ becomes infeasible (proving optimality of k) or a time limit is reached.

3 Staircase At-Most-One Constraint Sets

As a first step we define and illustrate the main concept of our paper, the so-called staircase AMO constraint set (SCAMO). Following that, in the next section we demonstrate step-by-step our proposed SAT encoding of these constraints.

Definition 1. *Given a sequence of Boolean variables $X = \langle x_1\, x_2 \cdots x_n \rangle$ and a width w s.t. $1 < w \leq n$, a* staircase constraint set *is formulated as follows:*

$$\mathrm{SCAMO}(X, w) = \bigwedge_{i=0}^{(n-w)} \left(\sum_{j=i+1}^{(i+w)} x_j \leq 1 \right) \ \textit{where } n = |X|.$$

Notice that this constraint is a special sub-case of SEQUENCE constraints (see e.g. [4–7]) and so could be formulated as SEQUENCE$(0, 1, w, X, \{1\})$.

In Example 1 we saw, that there is an ordering of the constraints in that problem such that each constraint differs only slightly from the previous one. For instance, in Example 1 the 1st and 2nd constraints both include the sum of x_2, x_3 and x_4 while the 2nd and 3rd both contain the sub-expression $x_3 + x_4 + x_5$. Since addition is associative, the sum of the variables can be calculated regardless of the grouping of the variables. However, if we would like to reuse previous calculations, it is more beneficial to evaluate the first AMO constraint for example as $x_1 + (x_2 + x_3 + x_4)$ instead of considering any other variable grouping (e.g. $(x_1 + x_2) + (x_3 + x_4)$). Doing so, the second constraint can just simply consider the result of $(x_2 + x_3 + x_4)$ together with x_5. Continuing the evaluation with the next constraint, we could reuse (x_3+x_4) from $(x_2+x_3+x_4)$, in case we calculated it as $x_2 + (x_3 + x_4)$, to decide $x_3 + x_4 + x_5 + x_6 \leq 1$ by combining it with $(x_5 + x_6)$. In general, each constraint shares a sub-sum over $w-1$ variables with the previous and at the same time with the next constraint.

Evaluating the very first constraint in this example in a right associative way allows us to reuse (at least once) all its sub-expression in the following three (i.e. $w-1$) constraints. However, in order to reuse these sub-expressions we need a left associative grouping of variables in the constraint $x_5 + x_6 + x_7 + x_8 \leq 1$, since in the second constraint we need x_5, then (x_5+x_6) and then $(x_5+x_6+x_7)$ to complement the reused sub-sums of $x_1 + x_2 + x_3 + x_4$.

All in all, considering only the first w constraints, we see that we need a right associative evaluation of the first constraint and a left associative grouping of the $(w + 1)$'th constraint. Figure 2 depicts how these variable groupings can be "bonded" together to reconstruct the original constraints of Example 1. Extending this pattern to the whole set of constraints, we can see that each w consecutive constraints need to be considered once left associative to combine with the previous w constraints' sub-expressions and once right associative, to combine with the next w constraints. Thus, in Fig. 2 the sum over variables x_5, x_6, x_7 and x_8 is actually considered twice, once with a left and once with a right associative variable ordering. This duplicate view of constraints is the main concept behind our proposed duplex encoding.

4 Duplex Encoding of Staircase Constraint Sets

Our goal is to exploit sharing of sub-expressions between constraints to obtain a compact encoding. Again, the main idea of our approach can be seen in Fig. 2 where we identified common sub-sums. In our concrete encoding we have to go one step further though and actually have to share sub-constraints. This is achieved by decomposing longer AMO constraints into two smaller ones using the following proposition. While the original longer constraints may be used only once, smaller constraints potentially can be shared and reused multiple times.

Proposition 1. *A constraint* $x_1 + \cdots + x_n \leq 1$ *holds iff for all* $1 \leq i < n$

$$(x_1+\ldots+x_i \leq 1)\wedge(x_{i+1}+\ldots+x_n \leq 1)\wedge(x_1+\ldots+x_i \leq 0\vee x_{i+1}+\ldots+x_n \leq 0).$$

$$
\begin{array}{lcll}
(x_1 + (x_2 + (x_3 + (x_4)))) & & & \leq 1 \land \\
(x_2 + (x_3 + (x_4))) & + & (x_5) & \leq 1 \land \\
(x_3 + (x_4)) & + & ((x_5) + x_6) & \leq 1 \land \\
(x_4) & + & (((x_5) + x_6) + x_7) & \leq 1 \land \\
 & & ((((x_5) + x_6) + x_7) + x_8) &
\end{array}
$$

$$
\begin{array}{lcll}
(x_5 + (x_6 + (x_7 + (x_8)))) & & & \leq 1 \land \\
(x_6 + (x_7 + (x_8))) & + & (x_9) & \leq 1 \land \\
(x_7 + (x_8)) & + & ((x_9) + x_{10}) & \leq 1 \\
(x_8) & & &
\end{array}
$$

Fig. 2: Decomposition of the staircase AMO constraint set of Example 1.

4.1 Sub-Constraint Construction

As a first step, given a sequence of variables $X = \langle x_1 \cdots x_n \rangle$ and width w, we partition the variables into $M = \lceil \frac{n}{w} \rceil$ consecutive *windows* $\omega_1, \omega_2, \ldots, \omega_M$, where ω_1 contains variables $x_1, \ldots, x_w$, ω_2 contains $x_{w+1}, \ldots, x_{2w}$ etc. Note that unless $(n \bmod w) = 0$, the very last window contains fewer than w variables.

Example 2. Continuing the previous example, our width $w = 4$ splits X into three windows: $\omega_1 = \{x_1, x_2, x_3, x_4\}$, $\omega_2 = \{x_5, x_6, x_7, x_8\}$ and $\omega_3 = \{x_9, x_{10}\}$.

To encode a SCAMO set of constraints as compositions of smaller constraints, we build two BDDs for each window with two different variable orderings (hence the name "duplex"). Notice that any SAT encoding technique of AMO constraints could be employed instead of BDDs (as long as we do duplex encoding by considering both directions). However, beyond the smaller AMO constraints, we further need AMZ constraints in order to connect the parts together (see the binary clause in Prop. 1). One benefit of BDDs is that we get these constraints automatically already by encoding the AMO constraints. Thus in this paper we will focus only on this BDD based approach.

Given window ω_i over variables $X_i = \{x_{i_1}, \ldots x_{i_w}\}$, we construct two two-rooted BDDs, both representing the same two constraints $x_{i_1} + \cdots + x_{i_w} \leq 1$ and $x_{i_1} + \cdots + x_{i_w} \leq 0$. The first BDD, which we call *forward* BDD, considers the AMO and AMZ constraints with a right associative variable grouping (i.e. with variable ordering $x_{i_1} < x_{i_2} < \ldots < x_{i_w}$). The other BDD, called *backward* BDD, represents the same constraints but with a left associative variable grouping (i.e. with variable ordering $x_{i_w} < x_{i_{w-1}} < \ldots < x_{i_1}$).

Abío et al. in [42] proposed a generalized arc-consistent, polynomial size ROBDD-based encoding for Pseudo-Boolean constraints. In our setting the constraints are all AMO or AMZ constraints without coefficients, and thus applying their approach leads to small and simple BDDs. The recursive algorithms in Fig. 3 present the main steps of this building process. In these procedures $\langle x_i \cdots x_j \rangle$ means an ordered sequence of consecutive variables and function `if-then-else` builds a BDD node with the given decision variable and high and

`BDD-AMO` (consecutive variables $\langle x_i \cdots x_j \rangle$)

1 $\mathcal{B}$:= `Search-AMO`($\langle x_i \cdots x_j \rangle$)
2 **if** $\mathcal{B} = \emptyset$ **then**
3   **if** $|\langle x_i \cdots x_j \rangle| = 1$ **then**
4         $\mathcal{B}_T, \mathcal{B}_F := \top, \top$
5   **else**
6         $\mathcal{B}_T$:= `BDD-AMZ`($\langle x_{i+1} \cdots x_j \rangle$)
7         $\mathcal{B}_F$:= `BDD-AMO`($\langle x_{i+1} \cdots x_j \rangle$)
8     $\mathcal{B}$:= **if-then-else**($x_i, \mathcal{B}_T, \mathcal{B}_F$)
9 **return** $\mathcal{B}$

`BDD-AMZ` (consecutive variables $\langle x_i \cdots x_j \rangle$)

1 $\mathcal{B}$:= `Search-AMZ`($\langle x_i \cdots x_j \rangle$)
2 **if** $\mathcal{B} = \emptyset$ **then**
3   **if** $|\langle x_i \cdots x_j \rangle| = 1$ **then**
4         $\mathcal{B}_T, \mathcal{B}_F := \bot, \top$
5   **else**
6         $\mathcal{B}_T := \bot$
7         $\mathcal{B}_F$:= `BDD-AMZ`($\langle x_{i+1} \cdots x_j \rangle$)
8     $\mathcal{B}$:= **if-then-else**($x_i, \mathcal{B}_T, \mathcal{B}_F$)
9 **return** $\mathcal{B}$

Fig. 3: Algorithms `BDD-AMO` and `BDD-AMZ` to construct binary decision diagrams for constraints over a given sequence of consecutive Boolean variables.

low BDD nodes. Building the forward BDDs of a window ω_i simply means to call `BDD-AMO` and `BDD-AMZ` with $\langle x_{i_1} \cdots x_{i_w} \rangle$ as parameter. To build the backward BDDs, we need to call the methods with $\langle x_{i_w} \cdots x_{i_1} \rangle$ as argument. The result in both cases (see Ex. 3) will be a two-rooted BDD with height of at most $(w+1)$.

Consider the following *layers* of these constructed BDDs. A non-leaf layer l_j (where $1 \leq j \leq w$) of a forward BDD (backward BDD) consists of two nodes, one capturing the AMO and another node representing the AMZ constraint over variables $\langle x_{i_j} \cdots x_{i_w} \rangle$ (respectively $\langle x_{i_{w-(j-1)}} \cdots x_{i_1} \rangle$ for the backward BDD).

Example 3. The upper part of Fig. 4 shows what the forward BDD of ω_1 in Example 2 looks like. The BDD is the result of calling `BDD-AMO`($\langle x_1\, x_2\, x_3\, x_4 \rangle$) and `BDD-AMZ`($\langle x_1\, x_2\, x_3\, x_4 \rangle$). Notice that due to the search for already existing BDDs at the beginning of each method (`Search-AMO` and `Search-AMZ`), the two calls result in a single shared structure (i.e. we have a partially reduced ordered BDD). Further notice that though node $x_4 \leq 1$ could be reduced simply to $\top$, we kept this node in the representation. In this BDD we can distuinguish four layers ($l_1 - l_4$) that refer to four sub-constraints of the root expressions.

The lower part of the figure depicts the backward BDD of ω_2 in Example 2, resulting from calls `BDD-AMO`($\langle x_8\, x_7\, x_6\, x_5 \rangle$) and `BDD-AMZ`($\langle x_8\, x_7\, x_6\, x_5 \rangle$). The variable ordering here is $x_8 < x_7 < x_6 < x_5$. Notice that the structure of the two BDDs are identical, they just talk about different variables in different orders.

4.2 CNF Encoding of BDDs

During BDD construction (e.g. after Line 5 in both algorithms of Fig. 3), or later in an independent traversal, we can assign new Boolean variables to each non-unit non-leaf node. Notice that top nodes of the forward and backward BDDs over the same variables can use the same Boolean variable.

Now, given a node with auxiliary Boolean variable b, that decides on variable x_i and has a true child node with variable t and a false child node with variable f, we introduce clauses to encode $x_i \rightarrow (b \leftrightarrow t)$ and $\neg x_i \rightarrow (b \leftrightarrow f)$. However, there are several simplification possibilities due to the structure of our BDDs and

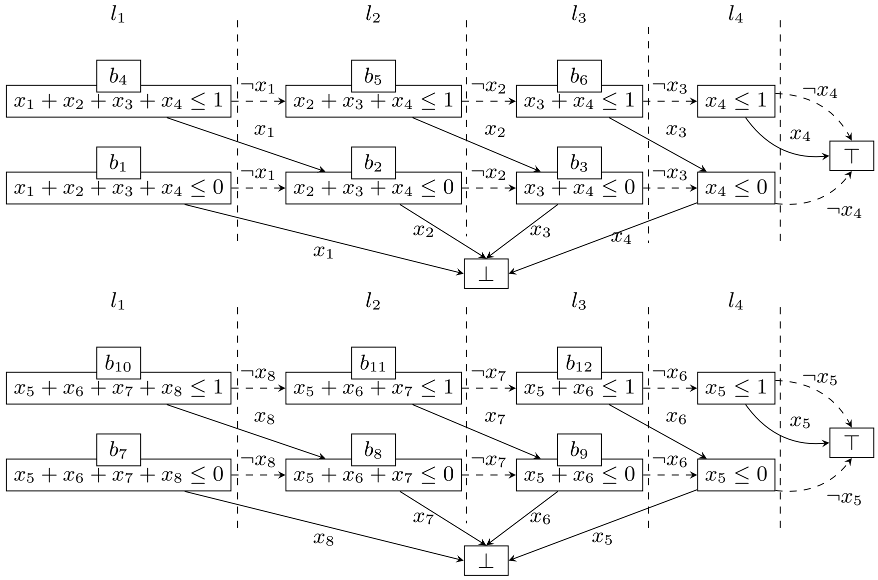


Fig. 4: Forward BDD of ω_1 with variable ordering $x_1 < x_2 < x_3 < x_4$ and backward BDD of ω_2 with ordering $x_8 < x_7 < x_6 < x_5$. Two-rooted partially reduced OBDDs to represent constraints $x_1 + x_2 + x_3 + x_4 \leq K$ with right and $x_5 + x_6 + x_7 + x_8 \leq K$ with left associative variable groupings, where $K \in \{0, 1\}$.

our problem. For instance, all AMZ nodes have $\perp$ as a true child (see Fig. 4) and all AMO nodes are assumed as unit clauses (due to using them with Prop. 1). Nodes of a constraint $x_i \leq 1$ are simply encoded as $\top$, while nodes of constraints $x_i \leq 0$ are encoded as $\neg x_i$ in the clausal representation of the parent nodes.

Example 4. On Fig. 4 the introduced new Boolean variables are represented together with their nodes. For example, variable b_6 belongs to the node of constraint $x_3 + x_4 \leq 1$. The introduced clause regarding this node is $(\neg x_3 \vee \neg x_4)$.

4.3 Bonding Stairs

An AMO constraint of a SCAMO set is either a root node of one of our BDDs or can be described by combining two layers of two BDDs via Prop. 1. As last step of encoding a whole SCAMO set of constraints, we traverse the forward BDD of each window (denoted as ω_i^f-BDD with $i \in \{1, \ldots, M-1\}$) and combine its nodes with those of the backward BDD of the next window (ω_{i+1}^b-BDD). Thus, we combine layer l_j of ω_i^f with layer $l_{(w-j)+2}$ of ω_{i+1}^b for each $j = 2, \ldots, w$. At the end, the bonding of two consecutive BDDs yields the following formula:

$$\mathrm{BOND}(\omega_i^f, \omega_{i+1}^b) \;=\; \omega_i^f\text{-}l_1\text{-AMO} \wedge$$

$$\bigwedge_{j=2}^{w} \Big(\omega_i^f\text{-}l_j\text{-AMO} \wedge \omega_{i+1}^b\text{-}l_{(w-j)+2}\text{-AMO} \wedge (\omega_i^f\text{-}l_j\text{-AMZ} \vee \omega_{i+1}^b\text{-}l_{(w-j)+2}\text{-AMZ}) \Big) .$$

Forward BDD of ω_1

ω_1^f-l_1	$(x_1 + (x_2 + (x_3 + (x_4)))) \leq 1$			
ω_1^f-l_2	$(x_2 + (x_3 + (x_4))) \leq 1$	$\wedge$	$(x_5) \leq 1$	ω_2^b-l_4
ω_1^f-l_3	$(x_3 + (x_4)) \leq 1$	$\wedge$	$((x_5) + x_6) \leq 1$	ω_2^b-l_3
ω_1^f-l_4	$(x_4) \leq 1$	$\wedge$	$(((x_5) + x_6) + x_7) \leq 1$	ω_2^b-l_2
			$((((x_5) + x_6) + x_7) + x_8) \leq 1$	ω_2^b-l_1

Backward BDD of ω_2

Fig. 5: Combining forward and backward BDDs to encode SCAMO constraints.

Example 5. We continue the running example. At this point we have seen how to construct a BDD for each small stair structure in Fig. 2. Next we combine them using Prop. 1 to capture all AMO constraints. Fig. 5 depicts how the layers of the constructed BDDs are meant to be paired with each other. Applying Prop. 1 on layers of ω_1^f-BDD and ω_2^b-BDD yields the following formula:

$$
\begin{gathered}
(x_1 + x_2 + x_3 + x_4 \leq 1) \wedge \\
(x_2 + x_3 + x_4 \leq 1) \wedge (x_5 \leq 1) \wedge (x_2 + x_3 + x_4 \leq 0 \vee x_5 \leq 0) \wedge \\
(x_3 + x_4 \leq 1) \wedge (x_5 + x_6 \leq 1) \wedge ((x_3 + x_4 \leq 0) \vee (x_5 + x_6 \leq 0)) \wedge \\
(x_4 \leq 1) \wedge (x_5 + x_6 + x_7 \leq 1) \wedge ((x_4 \leq 0) \vee (x_5 + x_6 + x_7 \leq 0)),
\end{gathered}
$$

that translates to the clauses $b_4 \wedge b_5 \wedge \top \wedge (b_2 \vee \neg x_5) \wedge b_6 \wedge b_{12} \wedge (b_3 \vee b_9) \wedge \top \wedge b_{11} \wedge (\neg x_4 \vee b_8)$. Notice that with this set of clauses, together with the BDD clauses, we encoded the first four AMO constraints of our SCAMO problem.

4.4 Arc Consistency of Duplex Encoding

Notice that AMO, AMZ and SCAMO constraints are all monotonic decreasing Boolean functions, i.e. setting any of the variables to false does not restrict any other variables. Thus setting a variable to true affects only those variables that share at least one AMO constraint with it. Note that decomposing each AMO constraint of a SCAMO set based on Prop. 1 results in an equivalent problem. Although our constructed BDDs for this decomposition share most of their nodes with each other (due to the chosen variable orders), our method is still a BDD-based translation of each AMO and AMZ constraint into clauses. Thus, applying an arc consistent encoding [48, 49] on each BDD node (e.g. the one in Minisat+ [41]) makes our encoding arc consistent as well.

In fact, notice that our bonding clauses contain a unit clause for each AMO constraint in order to enforce the output of the corresponding (sub-)BDD to be true. Beyond that, it is not hard to see that setting an input variable to true falsifies the output variable of each AMZ-BDD containing it. Thus the binary clauses of the bonding clauses enforce the root-node of each respective AMZ constraint to be true, and in turn unit propagation, the main inference rule of SAT solvers, falsifies all the variables in them.

5 Comparing Encodings of Staircase Constraints

In this section we discuss commonly used existing SAT encodings of AMO constraints and possible SEQUENCE encodings of SCAMO constraints. We compare them to our proposed duplex encoding in the context of SCAMOs.

Let $N = (n-w)+1$ be the number of AMO constraints in a staircase problem set over n variables and width w. A *naive* (also called *pair-wise* or *binomial*) encoding of a w-long AMO constraint is $\bigwedge_{i=1}^{(w-1)} \bigwedge_{j=i+1}^{(w)} (\neg x_i \vee \neg x_j)$. Although this approach does not require any additional Boolean variable, the number of clauses constructed with that encoding over N w-long AMO constraints is $N \cdot ((w-1) + (w-2) + \ldots + (w-(w-1)) = N \cdot \frac{(w-1)\cdot w}{2}$.

Using the naive encoding on the SCAMO constraint set would produce more than once many of the binary clauses. Eliminating duplicated clauses yields the *reduced naive* encoding with $\frac{(w-1)\cdot w}{2} + (N-1)\cdot(w-1)$ unique clauses.

Sinz introduced in [37] a *sequential* counter encoding for Boolean cardinality constraints. Applying it to an AMO constraint over w variables produces $3 \cdot w - 5$ binary clauses and introduces $w-2$ auxiliary variables. With N AMO constraints this gives $N \cdot (3 \cdot w - 5)$ clauses and $N \cdot (w-2)$ new variables.

The BDD-*based* encoding for Pseudo-Boolean constraints [41, 42] applied to AMO constraints is comparable to the sequential counter encoding. However, for a fixed variable order, the BDD built for each w-long AMO constraint of a SCAMO set, will always either contain a variable that does not occur in any other constraint or will miss a variable needed in other constraints. Thus for this approach using a fixed single variable order the amount of sharing of BDD nodes among constraints is rather restricted. On the other hand the approach does not require bonding clauses. With a simplified clausal representation of the BDD nodes, the naive BDD encoding uses at most $N \cdot (3 \cdot (w-2) + 2 \cdot (w-1) - 1)$ clauses and introduces $N \cdot (2 \cdot w - 3)$ new variables to encode a SCAMO set.

The so-called *2-product* encoding [32] relies on the same decomposition rule as Prop. 1. This approach breaks an AMO constraint over w variables into a product of two AMO constraints over p and q variables, where $p * q \geq w$. To simplify the calculation we use $p = \lceil \sqrt{w} \rceil$ and $q = \lceil w/p \rceil$ as recommended in [32] and assume recursive 2-product encoding of the resulting smaller constraints. Even though this approach can efficiently encode a single AMO constraint, making use of shared sub-expressions is not straightforward. Thus, based on the estimations given in [32], the number of clauses is $N \cdot (2 \cdot w + 4 \cdot \sqrt{w} + O(\sqrt[4]{w}))$. Further, the number of newly introduced variables is $N \cdot (2 \cdot \sqrt{w} + O(\sqrt[4]{w}))$ again following [32].

Instead of focusing on specialized AMO encodings, it is also possible to encode a complete SCAMO set with more generic approaches, like the ones in [6]. For example, encoding SCAMO as a REGULAR constraint yields similar results as a naive BDD-based approach with a single variable order (i.e. $\mathcal{O}(n \cdot w)$ size).

Another encoding (also from [6]) based on cumulative sums or difference constraints requires an internal representation which is at least quadratic size in the worst case. Similarly, partial sums (again see [6]) would consider every possible sub-sums which also yields $\mathcal{O}(n \cdot w^2)$ constraints.

Table 1: Comparison of size of SAT encodings of w-long SCAMO sets over n variables. Columns #NewVars and #Clauses show the number of additional variables and clauses of each approach, where $N = (n - w) + 1$ and $M = \lceil \frac{n}{w} \rceil$.

Encoding	#NewVars	#Clauses	WorstCase
Naive	0	$N \cdot \frac{(w-1)\cdot w}{2}$	$\mathcal{O}(n^3)$
Reduced	0	$\frac{(w-1)\cdot w}{2} + (N-1)\cdot(w-1)$	$\mathcal{O}(n^2)$
Sequential	$N \cdot (w-2)$	$N \cdot (3 \cdot (w-2) + 1)$	$\mathcal{O}(n^2)$
BDD	$N \cdot (2 \cdot w - 3)$	$N \cdot (3 \cdot (w-2) + 2 \cdot (w-1) - 1)$	$\mathcal{O}(n^2)$
2-Product	$N \cdot (2 \cdot \sqrt{w} + O(\sqrt[4]{w}))$	$N \cdot (2 \cdot w + 4 \cdot \sqrt{w} + O(\sqrt[4]{w}))$	$\mathcal{O}(n^2)$
Duplex	$4 \cdot M \cdot (w-1)$	$13 \cdot M \cdot w - 14 \cdot M - 3 \cdot w + 2$	$\mathcal{O}(n)$

The size-wise most competitive sequence encoding from [6] is the log-based approach where a SCAMO set could be represented as $\mathcal{O}(n \cdot log\, w)$ constraints.

5.1 Duplex Encoding

For a given constraint set over n variables of width w we construct two BDDs of the same size (each having $2 \cdot (w+1)$ nodes) for $M = \lceil \frac{n}{w} \rceil$ windows. To simplify the calculation, we will assume that each BDD has the same size (even though the last window is most of the time way smaller) and that we encode the first and last windows in both directions. Thus, we provide here just an upper bound on the actual values. With these assumptions we have $2 \cdot M$ BDDs. For each BDD we construct three clauses for the non-unit non-leaf AMZ nodes and at most two clauses for the non-unit non-leaf AMO nodes. Beyond these clauses, we need to bond together each layer of the neighbouring forward and backward BDDs, resulting in $M - 1$ bond-clause sets, each consisting of two unit and a binary clause. All in all, the final number of clauses in the encoding is as follows:

$$\#\text{BDD-clauses} \leq 2 \cdot M \cdot (3 \cdot (w-1) + 2 \cdot (w-1) - 1) = 10 \cdot M \cdot w - 12 \cdot M$$
$$\#\text{BOND-clauses} \leq (M-1) \cdot (3 \cdot (w-1) + 1) = 3 \cdot M \cdot w - 2 \cdot M - 3 \cdot w + 2$$
$$\#\text{BDD} + \#\text{BOND-clauses} \leq 13 \cdot M \cdot w - 14 \cdot M - 3 \cdot w + 2$$

The number of new variables at the very end of the encoding is at most $4 \cdot M \cdot (w-1)$ introducing one for each non-leaf non-unit node of our BDDs.

5.2 Comparison Summary

Table 1 summarizes the sizes of different SAT encodings expressed as functions over the number n of all variables in a SCAMO constraint set and the width w of the individual AMO constraints, combined into $N = (n - w) + 1$ (the number of AMO constraints) and $M = \lceil \frac{n}{w} \rceil$ (the number of windows in duplex encoding). The columns capture the number of auxiliary variables and number of clauses of the encodings. Notice that M is significantly smaller than N. The last column gives the worst case of each approach, assuming $w = n/2$, where N is approximately $n/2$ too. In this scenario existing encodings are quadratic or

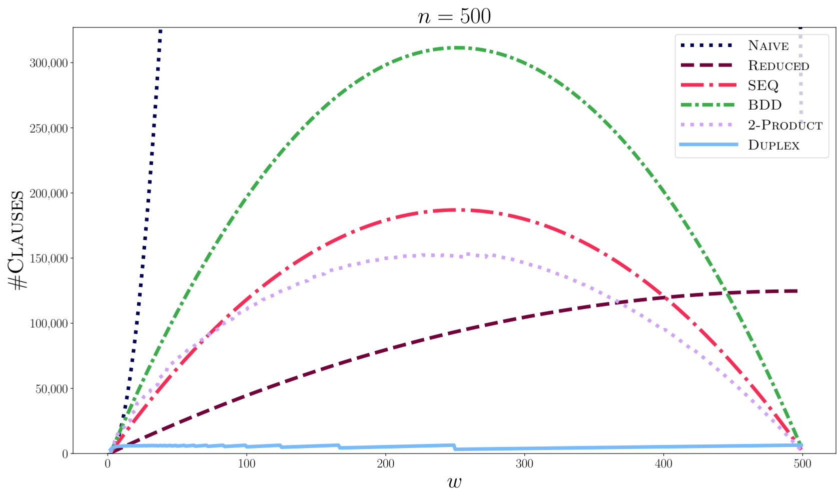


Fig. 6: Comparison of number of clauses for different encodings of a single SCAMO constraint set on $n = 500$ variables and width w between 2 and 500.

even cubic. However, in our duplex encoding we have $M = 2$ in that case and thus it remains linear.

Figure 6 visualizes the difference between SAT encodings for the fixed number of variables $n = 500$. The horizontal axis ranges over all possible widths w. Note that the naive encoding is only partially shown here, and further, that in our application $n/2$ is an upper bound on the width w, and thus only the left part of Fig. 6 is interesting up to the middle $w = n/2 = 250$.

The asymptotic behavior of the last column of Table 1 can be observed in Fig. 6 too. Again, the largest difference between the encodings occurs for $w = n/2$. According to Fig. 6 the reduced naive encoding turns out to be the best SAT-based alternative to our approach in terms of number of clauses. Though Fig. 6 focuses only on SAT encodings, note that the smallest sequence-based alternative (in [6]) would have size $\mathcal{O}(n \cdot log\, n)$ when $w = n/2$, that is smaller than most SAT encodings but larger than our proposed linear encoding.

6 Experimental Evaluation

Formulating the antibandwidth problem iteratively, as it was proposed in [8] (see Sect. 2), asks whether there exists a labelling for a graph $G = (V, E)$ s.t. $\mathrm{AB}(G) \geq k + 1$. The question has $2 \cdot |V|$ pieces of $|V|$-long exactly-one constraints (as (Labels) and (Vertices)) and for each edge of the graph (i.e. $|E|$ times) a $(|V| - k)$ big set of AMO constraints, each over $2 \cdot k$ variables (as (Obj_k)).

An off-the-shelf SAT solution could encode each of the AMO and exactly-one constraints one-by-one (e.g. as in Sect. 5). However, for a given edge between nodes i, i' (i.e. $\{i, i'\} \in E$) constraint ($\textsc{Obj}_k$) can be reformulated as

$$\bigwedge_{\lambda=1}^{(|V|-k)} \left(\sum_{\ell=\lambda}^{(\lambda+k)} x_i^\ell + x_{i'}^\ell \leq 1 \right) \overset{Prop.\ 1}{\equiv}$$

$$\bigwedge_{\lambda=1}^{(|V|-k)} \left(\sum_{\ell=\lambda}^{(\lambda+k)} x_i^\ell \leq 1 \wedge \sum_{\ell=\lambda}^{(\lambda+k)} x_{i'}^\ell \leq 1 \wedge \left(\sum_{\ell=\lambda}^{(\lambda+k)} x_i^\ell \leq 0 \vee \sum_{\ell=\lambda}^{(\lambda+k)} x_{i'}^\ell \leq 0 \right) \right).$$

In that form we have exactly two SCAMO sets of width $k+1$, one over the variables of node i and another over variables of i'. The third component of the decomposition takes the disjunction of AMZ constraints that can be constructed easily by combining our smaller AMZ nodes corresponding to the SCAMO sets.

The staircase structure in ($\textsc{Obj}_k$) allows to apply our new duplex encoding by simply encoding a SCAMO set of width $k+1$ for each node of the graph (i.e. $|V|$ times) and combining the corresponding AMZ constraints (with less than $4 \cdot (|V| - k)$ binary clauses for each edge). This encodes all AMO constraints of the problem. Also note that we can reuse the Boolean variables representing the root nodes of the constructed AMO BDDs to encode the (Vertices) constraints.

Experimental Results

We implemented a framework to compare off-the-shelf SAT encodings in practice to our proposed SCAMO based duplex encoding on the antibandwidth problem (as formulated in Sect. 2). Beyond SAT encodings, we also compared our approach against alternative exact methods to solve the problem, like Constraint Programming or the iterative method presented in [8] based on feasibility-MIPs.

The experiments considered 24 matrices of the Harwell-Boeing Sparse Matrix Collection [50], containing 12 relatively small and 12 rather large graphs (as in [8]). For each graph lower bounds (by a construction heuristic) and theoretical upper bounds of the antibandwidth were provided in [8]. These values were reused in our experiment as starting and ending points for the iterative methods and as lower bounds in the CP approaches. All reported results were experimented on our cluster with Intel Xeon E5-2620 v4 @ 2.10GHz CPUs.

Table 2 summarizes our results.[3] For each graph it shows the number of nodes and edges, the starting width or lower bound and last width to check of the solving methods (columns $|V|, |E|$ and LB,UB). Then for each solving technique we report the best found solution together with the time (in seconds) and memory consumption (in MB). Each approach was limited to 1800 seconds and 120 GB memory. This rather high main memory limit is due to trying to solve the alternative SAT encodings with a large number of clauses as well, while the other methods never exceeded 4 GB.

We compare the 2-product [32] and reduced naive AMO encodings to our proposed duplex SCAMO encoding as the first three techniques in Table 2.

[3] Source code, data and benchmarks are available at http://fmv.jku.at/duplex/.

All three techniques are implemented in the same framework and follow the same method: encode (considering LB as width of SCAMO or as k of the AMO constraints) and solve the SAT representation of the problem with a SAT solver (we used CaDiCaL 1.2.1 [51]). If it is satisfiable, increase the width and start again to encode and solve the new problem. If it is unsatisfiable or the width is UB, it means that the optimal solution of ABP was found and the process ends. At the moment when the 1800 seconds or 120 GB is exceeded, the method stops (with TO or MO respectively). The reported solutions are the highest widths with what the formula was still successfully constructed and solved. In case even the first formula was too hard to solve, it is marked with "-".

While the 2-product encoding of the largest instance had a memory out during solving the first formula (after a successful encoding), the reduced naive approach required less memory and even solved a few of the larger problems with more than one width in 1800 seconds. The duplex encoding required significantly less memory and was faster in encoding and solving the problems compared to the other SAT approaches. It performed well also compared to further techniques.

The next two approaches, $F_e(k)$ and CP-CPLEX, are taken from [8] as is, and were executed on our cluster for comparison. Note that while CP-CPLEX knows LB, $F_e(k)$ constructs it internally. The last reported approach is based on Chuffed [44, 52] via the MiniZinc language [53]. This hybrid solver employs lazy clause generation and combines the strengths of SAT and finite domain solving techniques. Note that both CP approaches encode the ABP naively as a labeling problem to maximize smallest neighbour-distances, using state-of-the-art solvers off-the-shelf. All in all we can see that the SCAMO based duplex encoding of the ABP is comparable and most of the time even better than other approaches.

7 Conclusion and Outlook

In this paper we have proposed a new SAT encoding for at-most-one constraints with a staircase structure, i.e. where consecutive constraints share sequences of sub-expressions in a structured way. This structure is exploited in an encoding which relies on binary decision diagrams using two variable orderings. Compared to alternative encodings for the ABP, our encoding outperforms the existing ones.

In the future we plan to integrate and interleave the MIP based approach of [8] and the SAT approach proposed here. Further, we want to apply the proposed method to other problems featuring at-most-one constraints with a staircase structure. Another intriguing direction for future work is to explore how symbolic optimization techniques using decision diagrams [54] can take advantage of multiple variable orders simultaneously, which is essential to keep our encoding compact.

Acknowledgments. This research has been supported by the Austrian Science Fund (FWF) under projects W1255-N23, S11408-N23 and by the LIT AI Lab funded by the State of Upper Austria. The authors would like to thank the reviewers for their useful suggestions and helpful comments.

Table 2: Results of different approaches to solve the antibandwidth problem (TO = 1800 seconds and MO = 120 GB).

Instance	$\lvert V\rvert$	$\lvert E\rvert$	LB	UB	2-Product			Reduced Naive			Duplex			F_e(k) [8]			CP-CPLEX [8]			CP-MZ-Chuffed		
					Obj.	Time	MB	Obj.	Time	MB	Obj.	Time	MB	Obj.	Time	MB	Obj.	Time	MB	Obj.	Time	MB
A-pores_1	30	103	6	8	**6**	206.85	80	**6**	166.48	68	**6**	185.52	52	**6**	23.71	29	6-8	TO	57	**6**	5.97	11
B-ibm32	32	90	9	9	**9**	14.06	51	**9**	46.03	47	**9**	1.30	11	**9**	28.57	29	**9**	7.35	20	**9**	17.4	11
C-bcspwr01	39	46	16	17	**17**	83.12	69	**17**	56.02	59	**17**	3.85	13	**17**	6.64	28	**17**	18.78	21	17	TO	11
D-bcsstk01	48	176	8	9	**9**	14.41	139	**9**	8.59	47	**9**	0.25	14	**9**	62.28	36	**9**	20.15	21	**9**	6.35	12
E-bcspwr02	49	59	21	22	**21**	36.17	76	**21**	53.01	80	**21**	3.37	13	**21**	774.02	205	**21**	22.84	19	**21**	673.44	11
F-curtis54	54	124	12	13	**13**	20.89	139	**13**	1.02	41	**13**	1.33	18	**13**	12.56	32	**13**	34.66	21	**13**	2.14	11
G-will57	57	127	12	14	**13**	108.79	164	**13**	26.8	79	**13**	0.57	19	**13**	15.4	33	**13**	44.75	21	**13**	2.69	11
H-impcol_b	59	281	8	8	**8**	5.51	173	**8**	0.47	52	**8**	0.54	22	**8**	0.47	24	8-22	TO	63	**8**	23.3	12
I-ash85	85	219	19	27	21	TO	794	21	TO	658	**23**	TO	331	20	TO	133	22-31	TO	37	21	TO	12
J-nos4	100	247	32	40	32	TO	1037	32	TO	911	**35**	585.33	190	-	TO	106	34-47	TO	31	-	TO	12
K-dwt__234	117	162	46	58	47	TO	924	47	TO	957	49	TO	477	48	TO	264	**51-57**	TO	33	-	TO	11
L-bcspwr03	118	179	39	39	**39**	22.82	662	**39**	6.92	436	**39**	0.99	58	**39**	0.52	21	**39**	110.92	22	**39**	26.42	12
M-bcsstk06	420	3720	28	72	-	TO	53392	29	TO	22076	**34**	TO	1621	33	TO	625	-	TO	20	-	TO	35
N-bcsstk07	420	3720	28	72	-	TO	53392	29	TO	22097	**34**	TO	1621	33	TO	634	-	TO	20	-	TO	35
O-impcol_d	425	1267	91	173	-	TO	30306	92	TO	22285	**99**	TO	1043	95	TO	691	-	TO	18	-	TO	24
P-can__445	445	1682	78	120	-	TO	41572	-	TO	27030	-	TO	1581	-	TO	644	-	TO	18	-	TO	24
Q-494_bus	494	586	219	246	-	TO	25944	-	TO	29640	-	TO	1167	**220**	TO	905	-	TO	18	-	TO	21
R-dwt__503	503	2762	46	71	-	TO	56611	47	TO	35227	**62**	TO	1680	52	TO	911	-	TO	19	-	TO	31
S-sherman4	546	1341	256	272	-	TO	73031	-	TO	59860	-	TO	1129	-	TO	1033	-	TO	19	-	TO	24
T-dwt__592	592	2256	103	150	-	TO	85816	-	TO	62936	-	TO	2253	-	TO	1068	-	TO	20	-	TO	37
U-662_bus	662	906	219	220	-	TO	63844	-	TO	68402	**220**	319.73	1564	-	TO	1320	-	TO	19	-	TO	28
V-nos6	675	1290	326	337	-	TO	101724	-	TO	90129	-	TO	1571	-	TO	1434	-	TO	20	-	TO	28
W-685_bus	685	1282	136	136	-	TO	76110	-	TO	72839	**136**	14.33	1428	**136**	9.24	37	-	TO	20	-	TO	29
X-can__715	715	2975	112	142	-	686.23	MO	-	TO	106462	-	TO	3312	-	TO	1468	-	TO	21	-	TO	39

References

1. Achterberg, T.: SCIP: solving constraint integer programs. Mathematical Programming Computation **1**(1) (2009) 1–41
2. Vielma, J.P.: Mixed integer linear programming formulation techniques. Siam Review **57**(1) (2015) 3–57
3. Bofill, M., Coll, J., Suy, J., Villaret, M.: SAT encodings of pseudo-boolean constraints with at-most-one relations. In Rousseau, L., Stergiou, K., eds.: Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4-7, 2019, Proceedings. Volume 11494 of LNCS., Springer (2019) 112–128
4. Beldiceanu, N., Contejean, E.: Introducing global constraints in CHIP. Mathematical and Computer Modelling **20** (03 1996) 97–123
5. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: SLIDE: A useful special case of the CARDPATH constraint. In Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N.M., eds.: ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings. Volume 178 of Frontiers in Artificial Intelligence and Applications., IOS Press (2008) 475–479
6. Brand, S., Narodytska, N., Quimper, C., Stuckey, P.J., Walsh, T.: Encodings of the SEQUENCE constraint. In Bessiere, C., ed.: Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings. Volume 4741 of LNCS., Springer (2007) 210–224
7. van Hoeve, W.J., Pesant, G., Rousseau, L., Sabharwal, A.: Revisiting the sequence constraint. In Benhamou, F., ed.: Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings. Volume 4204 of LNCS., Springer (2006) 620–634
8. Sinnl, M.: A note on computational approaches for the antibandwidth problem. CoRR **abs/1910.03367** (2019)
9. Gallian, J.A.: A dynamic survey of graph labeling. The Electronic Journal of Combinatorics **16**(6) (2009) 1–219
10. Leung, J.Y., Vornberger, O., Witthoff, J.D.: On some variants of the bandwidth minimization problem. SIAM Journal on Computing **13**(3) (1984) 650–667
11. Cappanera, P.: A survey on obnoxious facility location problems (1999)
12. Hale, W.K.: Frequency assignment: Theory and applications. Proceedings of the IEEE **68**(12) (1980) 1497–1514
13. Gansner, E.R., Hu, Y., Kobourov, S.: Gmap: Visualizing graphs and clusters as maps. In: Visualization Symposium (PacificVis), 2010 IEEE Pacific, IEEE (2010) 201–208
14. Miller, Z., Pritikin, D.: On the separation number of a graph. Networks **19**(6) (1989) 651–666
15. Yixun, L., JinJiang, Y.: The dual bandwidth problem for graphs. Journal of Zhengzhou University **35**(1) (2003)
16. Raspaud, A., Schröder, H., Sỳkora, O., Torok, L., Vrto, I.: Antibandwidth and cyclic antibandwidth of meshes and hypercubes. Discrete Mathematics **309**(11) (2009) 3541–3552
17. Wang, X., Wu, X., Dumitrescu, S.: On explicit formulas for bandwidth and antibandwidth of hypercubes. Discrete Applied Mathematics **157**(8) (2009) 1947–1952

18. Dobrev, S., Královič, R., Pardubská, D., Török, L., Vrto, I.: Antibandwidth and cyclic antibandwidth of hamming graphs. Discrete Applied Mathematics **161**(10-11) (2013) 1402–1408
19. Bekos, M.A., Kaufmann, M., Kobourov, S., Veeramoni, S.: A note on maximum differential coloring of planar graphs. Journal of Discrete Algorithms **29** (2014) 1–7
20. Bansal, R., Srivastava, K.: Memetic algorithm for the antibandwidth maximization problem. Journal of Heuristics **17**(1) (2011) 39–60
21. Duarte, A., Martí, R., Resende, M.G., Silva, R.M.: GRASP with path relinking heuristics for the antibandwidth problem. Networks **58**(3) (2011) 171–189
22. Lozano, M., Duarte, A., Gortázar, F., Martí, R.: Variable neighborhood search with ejection chains for the antibandwidth problem. Journal of Heuristics **18**(6) (2012) 919–938
23. Scott, J., Hu, Y.: Level-based heuristics and hill climbing for the antibandwidth maximization problem. Numerical Linear Algebra with Applications **21**(1) (2014) 51–67
24. Akker, van den, J.: LP-based solution methods for single-machine scheduling problems. PhD thesis, Technische Universiteit Eindhoven - Department of Mathematics and Computer Science (1994)
25. Boland, N., Kalinowski, T., Waterer, H., Zheng, L.: Mixed integer programming based maintenance scheduling for the hunter valley coal chain. Journal of Scheduling **16**(6) (2013) 649–659
26. Maravelias, C.T.: On the combinatorial structure of discrete-time MIP formulations for chemical production scheduling. Computers & Chemical Engineering **38** (2012) 204–212
27. Burke, E.K., Causmaecker, P.D., Berghe, G.V., Landeghem, H.V.: The state of the art of nurse rostering. J. Scheduling **7**(6) (2004) 441–499
28. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research **153**(1) (2004) 3–27
29. Dincbas, M., Simonis, H., Hentenryck, P.V.: Solving the car-sequencing problem in constraint logic programming. In Kodratoff, Y., ed.: 8th European Conference on Artificial Intelligence, ECAI 1988, Munich, Germany, August 1-5, 1988, Proceedings, Pitmann Publishing, London (1988) 290–295
30. Solnon, C., Cung, V., Nguyen, A., Artigues, C.: The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem. European Journal of Operational Research **191**(3) (2008) 912–927
31. Prestwich, S.D.: CNF encodings. In Biere, A., Heule, M., van Maaren, H., Walsh, T., eds.: Handbook of Satisfiability. Volume 185 of Frontiers in Artificial Intelligence and Applications. IOS Press (2009) 75–97
32. Chen, J.: A new sat encoding of the at-most-one constraint. Proc. Constraint Modelling and Reformulation (2010)
33. Manthey, N., Heule, M., Biere, A.: Automated reencoding of boolean formulas. In Biere, A., Nahir, A., Vos, T.E.J., eds.: Hardware and Software: Verification and Testing - 8th International Haifa Verification Conference, HVC 2012, Haifa, Israel, November 6-8, 2012. Revised Selected Papers. Volume 7857 of LNCS., Springer (2012) 102–117
34. Hölldobler, S., Nguyen, V.H.: On SAT-Encodings of the at-most-one constraint. In Katsirelos, G., Quimper, C.G., eds.: Proc. The Twelfth International Workshop

on Constraint Modelling and Reformulation, Uppsala, Sweden, September 16-20. (2013) 1–17
35. Knuth, D.E.: The Art of Computer Programming, Volume 4B, Fascicle 6: Satisfiability. Addison-Wesley (2015)
36. Nguyen, V.: SAT encodings of finite-csp domains: A survey. In: Proceedings of the Eighth International Symposium on Information and Communication Technology, Nha Trang City, Viet Nam, December 7-8, 2017, ACM (2017) 84–91
37. Sinz, C.: Towards an optimal CNF encoding of boolean cardinality constraints. In van Beek, P., ed.: Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings. Volume 3709 of LNCS., Springer (2005) 827–831
38. Liffiton, M.H., Maglalang, J.C.: A cardinality solver: More expressive constraints for free - (poster presentation). In Cimatti, A., Sebastiani, R., eds.: Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings. Volume 7317 of LNCS., Springer (2012) 485–486
39. Biere, A., Berre, D.L., Lonca, E., Manthey, N.: Detecting cardinality constraints in CNF. In Sinz, C., Egly, U., eds.: Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings. Volume 8561 of LNCS., Springer (2014) 285–301
40. Roussel, O., Manquinho, V.M.: Pseudo-boolean and cardinality constraints. In Biere, A., Heule, M., van Maaren, H., Walsh, T., eds.: Handbook of Satisfiability. Volume 185 of Frontiers in Artificial Intelligence and Applications. IOS Press (2009) 695–733
41. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. JSAT **2**(1-4) (2006) 1–26
42. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E., Mayer-Eichberger, V.: A new look at BDDs for pseudo-boolean constraints. J. Artif. Intell. Res. **45** (2012) 443–480
43. Philipp, T., Steinke, P.: PBLib - A library for encoding pseudo-boolean constraints into CNF. In Heule, M., Weaver, S.A., eds.: Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings. Volume 9340 of LNCS., Springer (2015) 9–16
44. Feydy, T., Stuckey, P.J.: Lazy clause generation reengineered. In Gent, I.P., ed.: Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings. Volume 5732 of LNCS., Springer (2009) 352–366
45. Abío, I., Gange, G., Mayer-Eichberger, V., Stuckey, P.J.: On CNF encodings of decision diagrams. In Quimper, C., ed.: Integration of AI and OR Techniques in Constraint Programming - 13th International Conference, CPAIOR 2016, Banff, AB, Canada, May 29 - June 1, 2016, Proceedings. Volume 9676 of LNCS., Springer (2016) 1–17
46. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers **35**(8) (1986) 677–691
47. Bryant, R.E.: Binary decision diagrams. In Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R., eds.: Handbook of Model Checking. Springer (2018) 191–217
48. Gent, I.P.: Arc consistency in SAT. In van Harmelen, F., ed.: Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002, IOS Press (2002) 121–125

49. Bacchus, F.: GAC via unit propagation. In Bessiere, C., ed.: Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings. Volume 4741 of LNCS., Springer (2007) 133–147
50. Rodriguez-Tello, E., Romero-Monsivais, H., Ramrez-Torres, J., Lardeux, F.: Harwell-Boeing graphs for the CB problem. https://www.researchgate.net/publication/272022702_Harwell-Boeing_graphs_for_the_CB_problem (2015)
51. Biere, A.: CaDiCaL at the SAT Race 2019. In Heule, M., Järvisalo, M., Suda, M., eds.: Proc. of SAT Race 2019 – Solver and Benchmark Descriptions. Volume B-2019-1 of Department of Computer Science Series of Publications B., University of Helsinki (2019) 8–9
52. Stuckey, P.J.: Lazy clause generation: Combining the power of SAT and CP (and MIP?) solving. In Lodi, A., Milano, M., Toth, P., eds.: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010, Bologna, Italy, June 14-18, 2010. Proceedings. Volume 6140 of LNCS., Springer (2010) 5–9
53. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: Towards a standard CP modelling language. In Bessiere, C., ed.: Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings. Volume 4741 of LNCS., Springer (2007) 529–543
54. Bergman, D., Ciré, A.A., van Hoeve, W., Hooker, J.N.: Decision Diagrams for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer (2016)