# Compact MDDs for Pseudo-Boolean Constraints with At-Most-One Relations in Resource-Constrained Scheduling Problems

**Miquel Bofill, Jordi Coll, Josep Suy** and **Mateu Villaret**
University of Girona, Spain
{miquel.bofill, jordi.coll, josep.suy, mateu.villaret}@imae.udg.edu

## Abstract

Pseudo-Boolean (PB) constraints are usually encoded into Boolean clauses using compact Binary Decision Diagram (BDD) representations. Although these constraints appear in many problems, they are particularly useful for representing resource constraints in scheduling problems. Sometimes, the Boolean variables in the PB constraints have implicit at-most-one relations. In this work we introduce a way to take advantage of these implicit relations to obtain a compact Multi-Decision Diagram (MDD) representation for those PB constraints. We provide empirical evidence of the usefulness of this technique for some Resource-Constrained Project Scheduling Problem (RCPSP) variants, namely the Multi-Mode RCPSP (MRCPSP) and the RCPSP with Time-Dependent Resource Capacities and Requests (RCPSP/t). The size reduction of the representation of the PB constraints lets us decrease the number of Boolean variables in the encodings by one order of magnitude. We close/certify the optimum of many instances of these problems.

## 1 Introduction

Scheduling problems consist of finding execution times for each of the activities of a project while respecting some constraints. The most frequent involve precedence relations between activities and limitations over the use of shared resources with finite capacity. Usually, the goal is to find a schedule that minimizes the makespan, i.e., the total duration of the project. These problems are NP-hard. Among scheduling problems, the Resource-Constrained Project Scheduling Problem (RCPSP) is a well-studied one which comprises non-preemptive activities and renewable constraints. Many extensions of this problem have been proposed [Brucker *et al.*, 1999]. One of the key points for solving these problems is how the resource constraints are handled.

In this work we use SAT Modulo Theories (SMT) encodings for scheduling problems: we use Integer Difference Logic (IDL) to enforce precedences, and SAT encodings of pseudo-Boolean (PB) constraints to control resources usage. Interestingly, in some cases the variables of such PB constraints have at-most-one (AMO) relations enforced by other constraints of the global encoding. We show how to take advantage of this fact to simplify the encodings.

The main contribution of this paper is to provide a specific way to use Multi-Decision Diagrams (MDD) to represent PB constraints with AMO relations between their variables, which we call AMO-PB constraints. AMO-PBs represent partial functions which do not include in their domain some assignments which are assumed to be forbidden due to AMO relations. Having a partial domain allows us to build compact MDDs which are translated into small SAT encodings. We show that such encodings are very efficient compared to BDD-based SAT encodings of PB constraints. Our approach can be applied to any problem in which PBs and AMO constraints between their variables appear together, but in this work we focus on scheduling problems. The use of AMO-PBs turns out to be decisive in some problems, where we are able to outperform state-of-the-art exact solvers.

In the problems we consider, there occur several AMO relations that we can exploit to compact PB constraints into AMO-PBs. Interestingly, in the formulation of our problems sometimes we do not need to include any SAT encoding of the AMOs, since these are enforced implicitly. On the one hand, we consider the RCPSP, where AMO relations arise from precedences between activities. On the other hand, we consider two extensions of it: the Multi-Mode RCPSP (MRCPSP) [Brucker *et al.*, 1999] where each activity can be executed in one of many modes, and the RCPSP with Time-Dependent Resource Capacities and Requests (RCPSP/t) [Hartmann, 2013]. The impact of taking into account AMOs is higher in these extensions, because the variables introduced to deal with modes or with resource requirement variations have additional AMO relations which make AMO-PB encodings much smaller than PB encodings. Our results show a reduction in the size of the encodings of one order of magnitude in terms of Boolean variables.

## 2 Related Work

Many exact approaches have been proposed to tackle the RCPSP and the MRCPSP, based on Integer Linear Programming (ILP) [Zhu *et al.*, 2006; Koné *et al.*, 2011], Failure-Directed Search (FDS) [Vilím *et al.*, 2015] or Lazy Clause Generation (LCG) [Schutt *et al.*, 2013; Szeredi and Schutt,

2016]. SMT has also been shown to be a competitive approach to solve such scheduling problems [Ansótegui *et al.*, 2011; Bofill *et al.*, 2016]. The latter demonstrated the good performance of PB constraints to encode resource constraints. In [Hartmann, 2015] it was introduced a genetic algorithm to solve the RCPSP/t, but to our knowledge no exact approach for this problem has been published.

Regarding the field of PB constraints, there have been many works on representing them as Binary Decision Diagrams (BDDs) and translating them to SAT encodings [Eén and Sorensson, 2006; Abío *et al.*, 2012]. In [Abío *et al.*, 2016] there were presented SAT encodings for the generalized case of MDDs. [Abío *et al.*, 2015] presented an algorithm to construct MDDs from PB constraints with implication chains. There, it is shown how this approach can be used to capture AMO relations between variables, but unlike our approach, it requires some transformations of the original constraints which introduce many additional clauses and fresh variables.

In [Abío and Stuckey, 2014] it was provided an algorithm to construct MDDs representing Linear Integer (LI) constraints. MDDs were also used in [Cire and van Hoeve, 2012] in the field of scheduling to model the *disjunctive* global constraint.

## 3 MDDs for Pseudo-Boolean Constraints with AMO Relations

Before going into the definition of AMO-PBs, we will introduce the concepts of PB and BDD, and illustrate how BDDs can represent PBs.

**Definition 3.1.** A *pseudo-Boolean* (PB) constraint has the form:

$$q_1 \cdot x_1 + \cdots + q_n \cdot x_n \# K$$

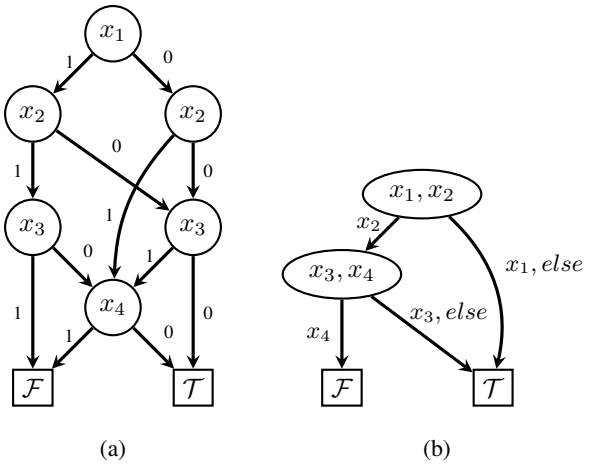
where $q_i$ and $K$ are integer constants, $x_i$ are $0/1$ ($true$/$false$) variables, and $\# \in \{<, \leq, =, \geq, >\}$. As usual in the literature and w.l.o.g. we will assume that $\#$ is $\leq$ [Eén and Sorensson, 2006].

**Definition 3.2.** A *Binary Decision Diagram* (BDD) is a rooted, directed, acyclic graph which represents a Boolean function. BDDs have two terminal nodes, namely $\mathcal{F}$-terminal and $\mathcal{T}$-terminal. Each non-terminal node has associated a Boolean variable (*selector*), and has two outgoing edges, representing the true and the false assignment of the selector. Every truth assignment of the variables follows a path from the root to the $\mathcal{T}$-terminal if it satisfies the formula, or to the $\mathcal{F}$-terminal otherwise.

A BDD is called *ordered* if different variables appear in the same order on all paths from the root. A BDD is said to be *reduced* if the following two rules have been applied until fix point:

- Merge any isomorphic subgraphs.
- Eliminate any node whose two children are isomorphic.

A *Reduced Ordered Binary Decision Diagram* (ROBDD) is canonical (unique) for a particular function and variable order.




(a) (b)

Figure 1: (a) ROBDD for $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$
(b) ROMDD for $\langle 2, 3\rangle \cdot \langle x_1, x_2\rangle + \langle 4, 5\rangle \cdot \langle x_3, x_4\rangle \leq 7$

(RO)BDDs can in particular represent PB constraints [Eén and Sorensson, 2006]. Figure 1a contains the ROBDD representation of the PB constraint $C : 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$ with the variable order $x_1 \prec x_2 \prec x_3 \prec x_4$. As seen in the picture, a BDD can be organized in different *layers*, and at each layer it is considered a different selector variable. For instance, in all the nodes of the second layer we choose whether to set $x_2$ to $true$ or to $false$.

In this work we deal with PB constraints whose variables satisfy AMO relations. For example, consider again the PB constraint $C : 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$, and assume that $AMO(x_1, x_2)$ and $AMO(x_3, x_4)$ hold, i.e., these constraints are already enforced. A possible truth assignment of $C$ would be $x_1 = true, x_2 = true$, but we know that such an assignment is already forbidden. Therefore, there is no need to handle this assignment in an encoding of the PB constraint. This motivates the definition of PB constraints with AMO relations as follows.

**Definition 3.3.** We refer to an integer linear expression $q_1 \cdot x_1 + \cdots + q_m \cdot x_m$ over 0/1 variables $x_1, \ldots, x_m$, subject to the fact that at most one $x_i$ is true, as an *AMO-product*. We conveniently express AMO-products as $QX$, where $Q = \langle q_1, \ldots, q_m\rangle$ and $X = \langle x_1, \cdots, x_m\rangle$.

**Definition 3.4.** We refer to an expression of the form $Q_1X_1 + \cdots + Q_nX_n \leq K$, where $Q_iX_i$ are AMO-products and $K$ is an integer constant, as a *PB constraint with AMO relations (AMO-PB)*.

Note that an AMO-PB can be seen as a partial function, whose value is undefined if the AMO relation does not hold for some $X_i$.

AMO-PBs generalize PBs, since the particular case in which each AMO-product contains only one variable is indeed a PB constraint. Now, still assuming that $AMO(x_1, x_2)$ and $AMO(x_3, x_4)$ hold, we can redefine $C$ as an AMO-PB $C' : \langle 2, 3\rangle \cdot \langle x_1, x_2\rangle + \langle 4, 5\rangle \cdot \langle x_3, x_4\rangle \leq 7$. This has to be read as: the sum of either $2x_1$ or $3x_2$ or non of them, plus $4x_3$ or $5x_4$ or none of them, is $\leq 7$. Unlike $C$, $C'$ is not defined for assignments such as $x_1 = true, x_2 = true$,

i.e., the assignments that do not satisfy the AMO relation between the variables of some AMO-product do not belong to the domain of $C'$. The key point of our approach is constructing decision diagram representations of AMO-PBs which do not handle such assignments; this lets us obtain more compact structures than the classical BDD representations of PB constraints. Obviously, it has to be guaranteed that the AMO constraint of each AMO-product is satisfied but, interestingly, it may be the case that these constraints are implied by other constraints, as we will see in Section 4. Now we show how to represent an AMO-PB constraint with an MDD.

**Definition 3.5.** A *Multi-Decision Diagram* (MDD) is a generalization of a BDD. It is a rooted, directed, acyclic multigraph which has two terminal nodes, namely $\mathcal{F}$-terminal and $\mathcal{T}$-terminal. Each non-terminal node can have an arbitrary number of outgoing edges, each one corresponding to a different decision. The same notions of ordered and reduced can be applied to MDDs, thus having also ROMDDs.

In their classical definition, MDDs stand for *Multi-valued Decision Diagrams* and can be seen as a generalization of BDDs which have a multi-valued selector variable in each node instead of a Boolean variable [Srinivasan *et al.*, 1990]. However, and especially in the context of SAT encodings of MDDs, a set of Boolean variables can be used as selectors, each variable representing a different decision.

In our setting, each non-terminal node has associated a list of Boolean selector variables $\langle x_1, \ldots, x_m \rangle$, and has an outgoing edge for each variable. Moreover, there is an additional outgoing edge, which we denote as the *else* edge. Each one of the $m+1$ outgoing edges corresponds to a different decision, namely assigning to $true$ exactly one of $x_i$, $1 \leq i \leq m$, or assigning all of them to $false$, hence choosing the *else* edge.

AMO-PBs can be represented with such MDDs, where each node handles an AMO-product, and all paths from the root to a terminal node choose (assign to $true$) at most one of the variables of each AMO-product. Every one of these truth assignments follows a path from the root to the $\mathcal{T}$-terminal if it satisfies the AMO-PB, or to the $\mathcal{F}$-terminal otherwise. Figure 1b contains the ROMDD representation of $C'$ with the order $\langle x_1, x_2 \rangle \prec \langle x_3, x_4 \rangle$. Unlike BDDs for PB constraints, where each layer considers a single variable, each layer of an ROMDD considers the set of variables of a different AMO-product. Note the significant size reduction with respect to the ROBDD representation of $C$ in Figure 1a. In particular, we have now just 2 non-terminal nodes instead of the 6 we had originally, and the depth is reduced from 5 to 3.

### 3.1 The Particular Case of Exactly-One

The particular case where the variables of an AMO-product satisfy an exactly-one (EO) constraint is already supported by AMO-PBs. However, in this case we can further simplify the AMO-product as follows. Consider an AMO-PB $Q_1X_1 + \cdots + Q_nX_n \leq K$. Let $X_i = \langle x_1, \ldots, x_m \rangle$ such that an EO relation holds on its variables, and $Q_i = \langle q_1, \ldots, q_m \rangle$. We can simplify $X_i$ into $X_i' = \langle x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_m \rangle$ and $Q_i$ into $Q_i' = \langle q_1 - q_j, \ldots, q_{j-1} - q_j, q_{j+1} - q_j, \ldots, q_m - q_j \rangle$, for some $j$, and replace the original AMO-PB by $Q_1X_1 + \cdots + Q_i'X_i' + \cdots + Q_nX_n \leq K - q_j$. Moreover, by choosing a $q_j$ which appears more than once, more than one variable can be removed from the AMO-product.

**Example 3.1.** *Let $C : \langle 2, 2, 3, 4 \rangle \cdot \langle x_1, x_2, x_3, x_4 \rangle \leq 3$, and assume that $EO(x_1, x_2, x_3, x_4)$ holds. By removing all $x_j$ with coefficient $q_j = 2$ in the previously described way, we obtain $C' : \langle 1, 2 \rangle \cdot \langle x_3, x_4 \rangle \leq 1$, and $AMO(x_3, x_4)$ holds. Note that we can replace $C$ by $C'$ because the following holds:*

$$C \wedge EO(x_1, x_2, x_3, x_4) \Leftrightarrow C' \wedge EO(x_1, x_2, x_3, x_4)$$

### 3.2 MDD Construction

In [Abío *et al.*, 2012], it was presented an algorithm to construct a ROBDD representing a PB constraint with a given variable ordering, whose running time is polynomial w.r.t. the size of the resulting ROBDD. Now we present a generalization of this algorithm to construct a ROMDD representing an AMO-PB with a given order on the sets of variables of the AMO-products. We highlight the main aspects of this generalized algorithm, and refer the reader to [Abío *et al.*, 2012] for further details.

Before describing the algorithm, we introduce the idea of interval for AMO-PB constraints.

**Definition 3.6.** Let $C$ be of the form $Q_1X_1 + \cdots + Q_nX_n \leq K$. The *interval* of $C$ are all integers $K'$ such that $Q_1X_1 + \cdots + Q_nX_n \leq K'$, interpreted as a Boolean function, is equivalent to $C$. The set of valid $K'$ is always an interval that we denote as $[\beta, \gamma]$.

For example, given the AMO-PB $\langle 1, 5, 4 \rangle \cdot \langle x_1, x_2, x_3 \rangle + \langle 4 \rangle \cdot \langle x_4 \rangle \leq 6$, its interval is $[5, 7]$, because its truth table is the same for $K' = 5$, $K' = 6$ and $K' = 7$, and different for $K' = 4$ and $K' = 8$ (taking into account that this AMO-PB is undefined for some assignments). The ROMDD representation of an AMO-PB constraint is the same for all the values of its interval.

At layer $i$ of the global ROMDD, each node is the root of the ROMDD representing the AMO-PB $Q_iX_i + \cdots + Q_nX_n \leq K$, for some different $K$ at each node. The algorithm ensures that the ROMDD is reduced by maintaining a set $L_i$ of tuples $([\beta, \gamma], \mathcal{M})$ for each layer $i$ of the global ROMDD, where $[\beta, \gamma]$ is an interval and $\mathcal{M}$ its associated ROMDD. We denote $\mathcal{L} = (L_1, \ldots, L_{n+1})$ the list of all $L_i$ sets of a ROMDD. By means of this dynamic programming method, the ROMDD representation for a particular interval of an AMO-PB constraint is constructed only once, and reused as many times as needed.

The main procedure is described in Algorithm 1. It receives as input an AMO-PB $C : Q_1X_1 + \cdots + Q_nX_n \leq K$, and the ordering used for the ROMDD is $X_1 \prec \cdots \prec X_n$. It starts by computing, for each layer of the final ROMDD, the interval of the $\mathcal{T}$-terminal ROMDD and the interval of the $\mathcal{F}$-terminal ROMDD. Then, it calls the recursive procedure *MDDBuild* (Algorithm 2). This inserts and searches ROMDDs in the lists $L_i$ while recursively constructing the global ROMDD. More precisely, each call creates (if it was not already constructed and stored in $L_i$) the ROMDD for a given AMO-PB $C : Q_iX_i + \cdots + Q_nX_n \leq K$ and layer $i$. To construct this ROMDD, the child for each possible decision

**Algorithm 1** Construction of ROMDD algorithm

**Require:** constraint $C : Q_1X_1 + \cdots + Q_nX_n \leq K$
**Ensure:** returns $\mathcal{M}$ the ROMDD of $C$
  **for all** $i$ such that $1 \leq i \leq n+1$ **do**
    $L_i \leftarrow \left\{ ((-\infty, -1], \mathcal{F}), ([\sum_{j \in [i,n]} max(Q_j), \infty), \mathcal{T}) \right\}$
  **end for**
  $\mathcal{L} \leftarrow \langle L_1, \ldots, L_{n+1} \rangle$
  $([\beta, \gamma], \mathcal{M}) \leftarrow$ **MDDBuild**$(1, C, \mathcal{L})$
  **return** $\mathcal{M}$

is recursively constructed, taking into account that choosing $x_j$ contributes with $q_j$ to the left-hand-side sum of $C$, and that choosing *else* has a 0 contribution. If all the children are the same ROMDD, i.e., all the outgoing edges point to the same ROMDD, no new node is created but this unique child is already the representation of $C$. Otherwise, the root node for the ROMDD representing $C$ is created, and its children are properly linked. Finally, the obtained ROMDD and its corresponding interval are inserted in $L_i$.

Algorithm 2 uses the following functions:

**search**$(K, L_i)$**:** If there is a tuple $(I, \mathcal{M})$ in $L_i$, such that $K \in I$, it is returned. Otherwise, an empty interval is returned in the first component of the tuple.

**insert**$((I, \mathcal{M}), L_i)$**:** Inserts $(I, \mathcal{M})$ into the set $L_i$.

**indexOfMax**$(Q_i)$**:** Returns the position of the maximum coefficient in list $Q_i$.

**mdd**$(\langle x_1, \ldots, x_m \rangle, \langle \mathcal{M}_1, \ldots, \mathcal{M}_m \rangle, \mathcal{M}_{else})$**:** Constructs an MDD with a new node as root, $\mathcal{M}_j$ as child for selector variable $x_j$, and $\mathcal{M}_{else}$ as the *else* child.

### 3.3 SAT Encodings of MDDs for AMO-PBs

The SAT encoding for MDDs that we use was firstly introduced in [Abío *et al.*, 2012] for BDDs representing monotonic PB constraints. This encoding is generalized arc-consistent. A generalized version for MDDs can be found in [Abío *et al.*, 2016]. Many other encodings were presented in the latter, not restricted to monotonic functions, but we have observed that in the resource constraints considered in this paper they generate larger encodings both in number of variables and clauses, and have a worse runtime performance.

In the AMO-PBs that we use in Section 4, all the coefficients are positive, and therefore we adapt the encoding for monotonic MDDs of [Abío *et al.*, 2016] to take into account the *else* case. We add a fresh auxiliary Boolean variable $v$ for each node of the MDD. Each non-terminal node of the MDD has as selector variables $\langle x_1, \ldots, x_m \rangle$, and $v$ as auxiliary variable. Each selector $x_i$ of a node has associated an outgoing edge to a child node with auxiliary variable $v_i$. We assert the following clauses for each non-terminal node:

$$\neg v_i \wedge x_i \rightarrow \neg v \qquad \forall i \in [1, m], v_i \neq v_{else}$$

$$\neg v_{else} \rightarrow \neg v$$

Moreover, we assert the clauses $\{v_r\}$, $\{v_{\mathcal{T}}\}$ and $\{\neg v_{\mathcal{F}}\}$, where $v_r, v_{\mathcal{T}}, v_{\mathcal{F}}$, are the auxiliary variables of the root, the $\mathcal{T}$-terminal and the $\mathcal{F}$-terminal nodes respectively. Note that a node can have more than one selector pointing to the same child, and if it happens with the *else* child, there is no need to add the clauses corresponding to selector variables pointing to the *else* child. We remark that this encoding does not enforce AMO constraints on the variables of the AMO-products, but we have constructed the AMO-PBs assuming that such constraints hold. For the soundness of the whole encoding of a problem, these constraints must be already enforced either implicitly, as shown in Section 4, or explicitly.

**Algorithm 2** Procedure MDDBuild

**Require:** integer $i \in \{1, \ldots, n+1\}$,
  constraint $C : Q_iX_i + \cdots + Q_nX_n \leq K$ and list $\mathcal{L}$
**Ensure:** returns $[\beta, \gamma]$ interval of $C$, $\mathcal{M}$ its ROMDD
  $([\beta, \gamma], \mathcal{M}) \leftarrow$ **search**$(K, L_i)$
  **if** $[\beta, \gamma] \neq \emptyset$ **then**
    **return** $([\beta, \gamma], \mathcal{M})$
  **else**
    $\langle x_1, \ldots, x_m \rangle \leftarrow X_i$
    $\langle q_1, \ldots, q_m \rangle \leftarrow Q_i$
    $\alpha \leftarrow$ **indexOfMax**$(Q_i)$
    **for all** $j$ such that $1 \leq j \leq m$ **do**
      $C' \leftarrow Q_{i+1}X_{i+1} + \cdots + Q_nX_n \leq K - q_j$
      $([\beta_j, \gamma_j], \mathcal{M}_j) \leftarrow$ **MDDBuild**$(i+1, C', \mathcal{L})$
    **end for**
    $C' \leftarrow Q_{i+1}X_{i+1} + \cdots + Q_nX_n \leq K$
    $([\beta_{else}, \gamma_{else}], \mathcal{M}_{else}) \leftarrow$ **MDDBuild**$(i+1, C', \mathcal{L})$
    **if** $[\beta_\alpha, \gamma_\alpha] = [\beta_{else}, \gamma_{else}]$ **then**
      $\mathcal{M} \leftarrow \mathcal{M}_\alpha$
      $[\beta, \gamma] \leftarrow [\beta_\alpha + q_\alpha, \gamma_\alpha]$
    **else**
      $\mathcal{M} \leftarrow$ **mdd**$(\langle x_1, \ldots, x_m \rangle, \langle \mathcal{M}_1, \ldots, \mathcal{M}_m \rangle, \mathcal{M}_{else})$
      $[\beta, \gamma] \leftarrow [\beta_{else}, \gamma_{else}] \cap \bigcap_{j \in [1,m]} [\beta_j + q_j, \gamma_j + q_j]$
    **end if**
    **insert**$(([\beta, \gamma], \mathcal{M}), L_i)$
    **return** $([\beta, \gamma], \mathcal{M})$
  **end if**

## 4 Application to Scheduling Problems

The *Resource-Constrained Project Scheduling Problem* (RCPSP) consists of finding a start time for each activity (a schedule) in a project while respecting precedence and resource usage constraints. The schedule must minimize the total duration of the project (makespan). It is defined by a tuple $(V, p, E, R, B, b)$ where:

- $V = \{A_0, A_1, \ldots, A_n, A_{n+1}\}$ is a set of activities. Activities $A_0$ and $A_{n+1}$ are dummy activities representing, by convention, the start and the end of the schedule, respectively. They don't consume resources and have duration 0.
- $p$ is a vector of naturals, where $p_i$ is the duration of $A_i$.
- $E$ is a set of pairs of activities representing precedence relations. Concretely, $(A_i, A_j) \in E$ iff the execution of activity $A_i$ must precede that of activity $A_j$, i.e., activity $A_j$ must start after activity $A_i$ has finished. We

assume that we are given a precedence activity-on-node graph $G(V, E)$ that contains no cycles, since otherwise the precedence relation is inconsistent.

- $R = \{R_1, \ldots, R_v\}$ is a set of renewable resources.
- $B \in \mathbb{N}^v$ is a vector of naturals, where $B_k$ is the available amount of each resource $R_k$.
- $b$ is a matrix of naturals corresponding to the resource demands of activities, and $b_{i,k}$ represents the amount of resource $R_k$ that activity $A_i$ is using per time step during its execution.

A schedule is a vector of naturals $S = (S_0, S_1, \ldots, S_n, S_{n+1})$ where $S_i$ denotes the start time of activity $A_i$.

It is common in the literature to compute the transitive closure $E^*$ of $E$, which contains a pair of activities $(A_i, A_j)$ iff there is a path from $A_i$ to $A_j$ in the precedence graph. We will denote $G^* = (A, E^*)$ as the extended precedence graph.

In ([Bofill *et al.*, 2016]) it is proposed an SMT encoding for the MRCPSP which models the precedences as IDL expressions, and enforces the resource constraints with BDD encodings of PB constraints. The overall idea of the encoding is to capture, for each activity $A_i \in V$, and for each discrete time instant $t \in TW(A_i)$, whether $A_i$ is running at time $t$. Here $TW(A_i)$ is the range of time instants in which $A_i$ can be running according to some preprocessed information [Artigues *et al.*, 2007]. In the current work we use the precedence graph to infer earliest start times and latest close times. Then, the encoding enforces the resource constraints at all time instants $t \in [0, H]$, where $H$ is a large enough time horizon to ensure the feasibility of the project. The following equations are a subset of this encoding properly adapted for the case of RCPSP. We do not reproduce the whole encoding, but the principal equations involved in the contributions of this paper.

$$S_j - S_i \geq p_i \qquad \forall (A_i, A_j) \in E \tag{1}$$

$$x_{i,t} \leftrightarrow (S_i \leq t < S_i + p_i) \quad \forall A_i \in V, \forall t \in TW(A_i) \tag{2}$$

$$\left( \sum_{A_i \in V, t \in TW(A_i), b_{i,k} > 0} b_{i,k} \cdot x_{i,t} \right) \leq B_k \qquad \forall R_k \in \{R_1, \ldots, R_v\}, \forall t \in [0, H] \tag{3}$$

$S_i$ are integer variables denoting the start time of activity $A_i$, and $x_{i,t}$ are Boolean variables which are true iff $A_i$ is running at time $t$. Constraints (1) enforce the precedence relations, Constraints (2) define $x_{i,t}$, and PB Constraints (3) ensure that the capacity of $R_k$ is not surpassed at time $t$.

### 4.1 Implicit AMO from Precedences

A precedence $(A_i, A_j) \in E$ introduces an incompatibility between these two activities, i.e., they can never be running at the same time. It is also the case for any pair of activities connected by a path in the precedence graph. For this reason, for a given time instant $t$ there is a mutual exclusion between any pair of variables $(x_{i,t}, x_{j,t})$ such that $(A_i, A_j) \in E^*$. This means that all the variables $x_{i,t}$ of activities in a path are pairwise mutually exclusive, or in other words, they satisfy the AMO relation. Note that there are no clauses explicitly enforcing the AMO over variables $x_{i,t}$ in a path, but it is guaranteed by Equations (1) and (2).

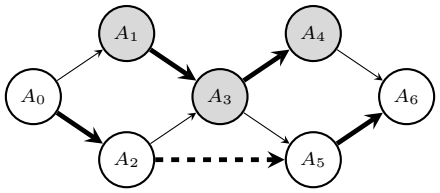



Figure 2: An example precedence graph with a minimum path cover of its extended precedence graph.

What we will do is to compute a path cover $\mathcal{P} = \langle P_1, ..., P_l \rangle$ in $G^*$, i.e., $P_i = \langle A_{i_1}, \ldots, A_{i_k} \rangle$ is a path in $G^*$ and every activity belongs to one and only one path.

For every path $P_i \in \mathcal{P}$ we will have an AMO-product in an AMO-PB constraint, and thus a layer in the resulting ROMDD. For this reason we are interested in finding a minimum path cover, which can be done in polynomial time in directed acyclic graphs [Fulkerson, 1956]. Note the importance of computing the minimum path cover in $G^*$ instead of $G$. Figure 2 depicts a precedence graph of a project with 7 activities. For the sake of simplicity, the picture does not contain all the additional extended precedences but only $(A_2, A_5) \in E^*$. Thanks to this extended edge, we can find $\mathcal{P} = \langle \langle A_1, A_3, A_4 \rangle, \langle A_0, A_2, A_5, A_6 \rangle \rangle$ which only has 2 paths. Otherwise, using the non-extended precedence graph a minimum path cover would contain 3 paths.

### 4.2 AMO-PB for Resource Constraints

Now we show how to use minimum path covers on $G^*$ to reformulate the resource constraints (3) as AMO-PB constraints. For some time instant $t$:

1. Compute $G^*(t)$ the subgraph from $G^*$ that contains all the nodes of the activities $A_i$ such that $t \in TW(A_i)$, i.e., $G^*(t) = (V(t), E^*(t))$, where $V(t) = \{A_i \,|\, A_i \in V, t \in TW(A_i)\}$, and $E^*(t) = \{(A_i, A_j) \,|\, (A_i, A_j) \in E^*, A_i \in V(t), A_j \in V(t)\}$.
2. Compute a minimum path cover $\mathcal{P}(t) = \langle P_1, ..., P_l \rangle$ of $G^*(t)$.
3. Formulate the AMO-PB constraint by defining an AMO-product for each $P_i \in \mathcal{P}(t)$

The first two steps remain the same for many generalizations of the RCPSP, and in particular for the ones considered in this paper. The implementation of the third step for each particular problem is described below.

**RCPSP**

We substitute PB constraints (3) for the following AMO-PB constraints:

$$\sum_{P_i \in \mathcal{P}(t)} Q(P_i, k) \cdot X(P_i, k, t) \leq B_k \qquad \forall R_k \in \{R_1, \ldots, R_v\}, \forall t \in [0, H] \tag{4}$$

Where $Q(P_i, k)$ is the list of coefficients:

$$Q(P_i, k) = \langle b_{j,k} \mid A_j \in P_i \wedge b_{j,k} > 0 \rangle$$

And $X(P_i, k, t)$ is the list of Boolean variables:

$$X(P_i, k, t) = \langle x_{j,t} \mid A_j \in P_i \wedge b_{j,k} > 0 \rangle$$

Note the AMO constraint among the variables of the AMO-products is implicitly enforced by the precedences in the paths. This also happens in MRCPSP and RCPSP/t.

**MRCPSP**

The *Multi-Mode Resource-Constrained Project Scheduling Problem* (MRCPSP) introduces different execution modes for the activities. The durations and the demands of the activities depend on their execution mode. In addition to the start times, solutions have also to assign a single execution mode to each activity.

We denote the number of possible execution modes of $A_i$ as $M_i$. For an activity $A_i$, running in mode $o \in [1, M_i]$, its demand over resource $R_k$ is $b_{i,k,o}$, and its duration is $p_{i,o}$. As done in [Bofill *et al.*, 2016], we add the Boolean variables $m_{i,o}$ which are true iff $A_i$ runs in mode $o$. It has to be ensured that each activity runs in exactly one mode. We encode this constraint (5) with the classical quadratic encoding. The dependence of the durations and the demands on the execution modes slightly modifies the precedence constraints (6), and requires the substitution $x_{i,t}$ for $x_{i,t,o}$ (7), now meaning that $A_i$ is running at time $t$ in execution mode $o$:

$$exactly\_one_{o \in [1, M_i]}(m_{i,o}) \qquad \forall A_i \in V \tag{5}$$

$$m_{i,o} \rightarrow (S_j - S_i \geq p_{i,o}) \qquad \forall (A_i, A_j) \in E, \forall o \in [1, M_i] \tag{6}$$

$$x_{i,t,o} \leftrightarrow (S_i \leq t \wedge t < S_i + p_i \wedge m_{i,o}) \qquad \forall A_i \in V, \forall t \in TW(A_i), \forall o \in [1, M_i] \tag{7}$$

We update Constraints (4) by redefining $Q$ and $X$:

$$Q(P_i, k) = \langle b_{j,k,o} \mid A_j \in P_i \wedge o \in [1, M_j] \wedge b_{j,k,o} > 0 \rangle$$

$$X(P_i, k, t) = \langle x_{j,t,o} \mid A_j \in P_i \wedge o \in [1, M_j] \wedge b_{j,k,o} > 0 \rangle$$

Moreover, in this problem there can be non-renewable resources, whose demanded capacity is not recovered after the demanding activity ends its execution. Letting $\{R_{v+1}, \ldots, R_w\}$ be the non-renewable resources, the AMO-PB constraint which guarantees that their capacity is not exceeded is the following:

$$\sum_{A_i \in V} Q(A_i, k) \cdot X(A_i, k) \leq B_k \qquad \forall R_k \in \{R_{v+1}, \ldots, R_w\} \tag{8}$$

Having:

$$Q(A_i, k) = \langle b_{i,k,o} \mid o \in [1, M_i] \rangle$$
$$X(A_i, k) = \langle m_{i,o} \mid o \in [1, M_i] \rangle$$

Since variables $m_{i,o}$ maintain an EO relation for a fixed $A_i$ (5), we apply the reduction explained in Section 3.1 to Constraints (8). In order to preserve the monotonicity required in the SAT encoding that we use for the MDDs, we take the minimum $q_j$ in each AMO-product. Thanks to this, we can remove at least one variable from each AMO-product. This turns out to be very significant in the benchmark instances considered in Section 5, which contain activities with 3 execution modes, and hence the number of variables in AMO-PB constraints is reduced at least to $2/3$.

Note that the AMO constraint among the variables of AMO-products is preserved both in renewable and non-renewable resource constraints. This follows from Constraints (5) and (7).

An interesting property of using AMO-PBs in MRCPSP is that the inclusion of multiple execution modes for the activities in the resource constraints do not yield an increase of the depth of the MDDs with respect to the single-mode case, because all the variables of possible modes of a particular activity are included in a same AMO-product (and hence handled in the same MDD layer). The BDD representation of resource PB constraints in [Bofill *et al.*, 2016] multiplies the depth of the BDDs by the number of modes of the activities.

**RCPSP/t**

In this generalization of the RCPSP, the resources' capacities can be different at each time instant, and the requirements of an activity can be different at each time instant of its execution. Now we denote the capacity of resource $R_k$ at time $t$ as $B_{k,t}$, and the demand of activity $A_i$ at its $e$-th execution time instant as $b_{i,k,e}$. In order to encode this problem, we redefine variables $x_{i,t}$ to have a different meaning than in RCPSP. Now $x_{i,t}$ is true iff activity $A_i$ starts at time $t$:

$$x_{i,t} \leftrightarrow S_i = t \qquad \forall A_i \in V, \forall t \in TW'(A_i) \tag{9}$$

Where $TW'(A_i)$ is the range of time instants at which $A_i$ can start. Note that $TW'(A_i) \subseteq TW(A_i)$. Now we can redefine the resource constraints (4) as:

$$\sum_{P_i \in \mathcal{P}(t)} Q(P_i, k, t) \cdot X(P_i, k, t) \leq B_{k,t} \qquad \forall R_k \in \{R_1, \ldots, R_v\}, \forall t \in [0, H] \tag{10}$$

Having:

$$Q(P_i, k, t) = \langle b_{j,k,e} \mid A_j \in P_i \wedge e \in [0, p_j) \wedge \wedge\, t - e \in TW'(A_j) \wedge b_{j,k,e} > 0 \rangle$$

$$X(P_i, k, t) = \langle x_{j,t-e} \mid A_j \in P_i \wedge e \in [0, p_j) \wedge \wedge\, t - e \in TW'(A_j) \wedge b_{j,k,e} > 0 \rangle$$

The main idea of Constraints (10) is that an activity can only be running at time $t$ because it has started at some time $t - e$ for $e \in [0, p_j)$. Note that the AMO relation among all the variables $x_{i,t}$ of a given $A_i$ is guaranteed by the domain of $S_i$ and Constraints (9), i.e., an activity has only one start time.

## 5 Experiments

We have run our experiments on a 8GB Intel® Xeon® E3-1220v2 machine at 3.10 GHz. In all experiments we use

| set | tool | 25% | 50% | 75% | avg | t.o. |
|---|---|---|---|---|---|---|
| **MRCPSP** | | | | | | |
| j30 (640) | FDS | 0.03 | 0.07 | 6.4 | 38.6 | 15 |
| | LCG | 0.11 | 0.21 | 0.6 | 29.6 | 25 |
| | PB | 1.04 | 2.00 | 4.4 | 23.7 | 14 |
| | APB | 0.33 | 0.49 | 0.8 | 14.7 | 9 |
| M50 (540) | FDS | 0.04 | 1.33 | t.o. | 173.4 | 140 |
| | LCG | 0.59 | 6.59 | 338.0 | 162.4 | 115 |
| | PB | 3.35 | 7.94 | 127.3 | 143.1 | 105 |
| | APB | 1.07 | 1.93 | 24.2 | 108.2 | 88 |
| **RCPSP/t** | | | | | | |
| j30 (2880) | PB | 0.26 | 1.04 | 3.1 | 3.1 | 0 |
| | APB | 0.05 | 0.16 | 0.4 | 0.4 | 0 |
| j120 (3600) | PB | 29.61 | t.o. | t.o. | 368.1 | 2085 |
| | APB | 7.16 | 56.18 | t.o. | 253.8 | 1390 |

Table 1: Solving time results. All values are in seconds, with a timeout of 600 seconds. The first column contains two values: on top there is the name of the dataset, which specifies the number of non-dummy activities of each instance; below that, between parenthesis, there is the number of instances in the dataset. Column 2 specifies, for each row, the tool used to solve the instances of the dataset. Columns 25%, 50% and 75% specify the first, second, and third quartile of the running times of solving each instance of the dataset, and *t.o.* means timeout. Column *avg* contains the average run time, counting timeouts as 600 seconds. The last column contains the number of instances that timed out without having found an optimal solution and proven its optimality.

Yices 2.4.2 [Dutertre and de Moura, 2006] as the core SMT solver. We have also implemented the optimization procedure described in [Bofill *et al.*, 2016]. We have tested our proposal on benchmark datasets of different hardness available in PSPLib [Kolisch and Sprecher, 1997] and MMLIB [Van Peteghem and Vanhoucke, 2014].[1]

First of all we compare the time performance of our system (*APB* in short) with the best known exact methods for these problems. Namely, for RCPSP and MRCPSP we consider Failure-Directed Search (*FDS*) [Vilím *et al.*, 2015], Lazy Clause Generation (*LCG*) [Schutt *et al.*, 2013; Szeredi and Schutt, 2016], and SMT with PB constraints (*PB*) [Bofill *et al.*, 2016]. We have been able to run all the solvers in the described environment except LCG for the RCPSP, for which we have considered results from [Schutt *et al.*, 2013]. To the best of our knowledge there is no exact approach to tackle RCPSP/t. In order to evaluate the impact of using AMO-PB constraints we have also codified this problem by transforming AMO-PBs into PBs in the obvious way (i.e., splitting all AMO-products into single variable products). These results are given in Table 1.

In the MRCPSP, APB is able to solve to optimality within the timeout many more instances than the other approaches, especially for MMLIB50 (M50 in the table), which is the hardest set of instances. Both in MRCPSP and RCPSP/t there is a significant speedup using AMO-PB constraints w.r.t. using PB constraints. One possible cause for this can be found in Table 2. It contains, for PB and APB, the average size of the encoding of a dataset in terms of number of Boolean variables and number of clauses. We consider the hardest set of each problem, i.e., MMLIB50 for MRCPSP and j120 for RCPSP/t. It can be seen that the reduction is an order of magnitude for the number of Boolean variables and halving for the number of clauses.

[1]Our solvers and detailed results can be downloaded at http://imae.udg.edu/recerca/LAP

| problem | tool | vars | clauses |
|---|---|---|---|
| MRCPSP | PB | 281 | 569 |
| | APB | 45 | 277 |
| RCPSP/t | PB | 3450 | 5605 |
| | APB | 360 | 3012 |

Table 2: Comparison of the number of Boolean variables and clauses. The values are expressed in thousands. We consider the set MMLIB50 for MRCPSP and j120 for RCPSP/t.

Regarding RCPSP, we have tested our system with sets j30, j60, j90 and j120 from PSPLib. We do not include detailed results for lack of space, but we have observed that APB average runtimes are very similar with the ones of FDS and LCG, and the number of unsolved instances is equal or smaller in all cases. Overall, we certify the optimality of 6 new instances and decrease the upper bound of 2 instances for the MMLIB50 dataset with respect to the best reported results until now [Bofill *et al.*, 2016; Geiger, 2013]. Finally, as mentioned earlier, to our knowledge no previous results of exact methods have been published for the RCPSP/t datasets, but they have been tackled with heuristic methods in [Hartmann, 2013] and [Hartmann, 2015]. We have certified the optimality, or infeasibility for the time horizon specified, of 5090 instances, and improved the upper bound of 1763 instances with respect to the heuristic methods.

## 6 Conclusions

In this work we have shown that the codification into SAT of PB constraints which have AMO relations among their variables can be enhanced by using MDD representations. Although our technique can be used in any problem with such kind of constraints, we have shown that in the particular case of scheduling problems it performs very well. This suggests that the presented approach could be successfully used in other domains. In addition, we have presented SMT encodings for scheduling problems which have shown to provide very good performance, and that can be handled by any off-the-shelf SMT solver.

## Acknowledgments

Work supported by grants TIN2015-66293-R (MINECO/FEDER, UE), MPCUdG2016/055 (UdG), and *Ayudas para Contratos Predoctorales 2016* (grant number BES-2016-076867, funded by MINECO and co-funded by FSE). We thank the authors of [Vilím *et al.*, 2015; Szeredi and Schutt, 2016] for sharing with us the FDS and LCG solvers, and the author of [Hartmann, 2013; 2015] for sharing the heuristics results for the RCPSP/t. We are also grateful to Dr. Alan Frisch for his helpful comments.

## References


[Abío and Stuckey, 2014] Ignasi Abío and Peter J Stuckey. Encoding linear constraints into SAT. In *International Conference on Principles and Practice of Constraint Programming*, pages 75–91. Springer, 2014.

[Abío *et al.*, 2012] Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Valentin Mayer-Eichberger. A new look at BDDs for pseudo-Boolean constraints. *Journal of Artificial Intelligence Research*, pages 443–480, 2012.

[Abío *et al.*, 2015] Ignasi Abío, Valentin Mayer-Eichberger, and Peter J Stuckey. Encoding linear constraints with implication chains to CNF. In *International Conference on Principles and Practice of Constraint Programming*, pages 3–11. Springer, 2015.

[Abío *et al.*, 2016] Ignasi Abío, Graeme Gange, Valentin Mayer-Eichberger, and Peter J Stuckey. On CNF Encodings of Decision Diagrams. In *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, page 1. Springer, 2016.

[Ansótegui *et al.*, 2011] Carlos Ansótegui, Miquel Bofill, Miquel Palahí, Josep Suy, and Mateu Villaret. Satisfiability Modulo Theories: An Efficient Approach for the Resource-Constrained Project Scheduling Problem. In *Proceedings of the Ninth Symposium on Abstraction, Reformulation, and Approximation (SARA)*, pages 2–9. AAAI, 2011.

[Artigues *et al.*, 2007] Christian Artigues, Sophie Demassey, and Emmanuel Neron. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. ISTE Ltd., 2007.

[Bofill *et al.*, 2016] Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. Solving the multi-mode resource-constrained project scheduling problem with SMT. In *28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2016, San Jose, CA, USA, November 6-8, 2016*, pages 239–246, 2016.

[Brucker *et al.*, 1999] Peter Brucker, Andreas Drexl, Rolf Mhring, Klaus Neumann, and Erwin Pesch. Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. *European Journal of Operational Research*, 112(1):3 – 41, 1999.

[Cire and van Hoeve, 2012] Andre A Cire and Willem Jan van Hoeve. MDD Propagation for Disjunctive Scheduling. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 11–19, 2012.

[Dutertre and de Moura, 2006] B. Dutertre and L. de Moura. The Yices SMT Solver. Technical report, Computer Science Laboratory, SRI International, 2006. Available at `http://yices.csl.sri.com`.

[Eén and Sorensson, 2006] Niklas Eén and Niklas Sorensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.

[Fulkerson, 1956] Delbert Ray Fulkerson. Note on Dilworths decomposition theorem for partially ordered sets. In *Proc. Amer. Math. Soc*, volume 7, pages 701–702, 1956.

[Geiger, 2013] Martin Josef Geiger. Iterated variable neighborhood search for the resource constrained multi-mode multi-project scheduling problem. *CoRR*, abs/1310.0602, 2013.

[Hartmann, 2013] Sönke Hartmann. Project scheduling with resource capacities and requests varying with time: a case study. *Flexible Services and Manufacturing Journal*, 25(1-2):74–93, 2013.

[Hartmann, 2015] Sönke Hartmann. Time-varying resource requirements and capacities. In *Handbook on Project Management and Scheduling Vol. 1*, pages 163–176. Springer, 2015.

[Kolisch and Sprecher, 1997] Rainer Kolisch and Arno Sprecher. PSPLIB - A Project Scheduling Problem Library. *European Journal of Operational Research*, 96(1):205–216, 1997.

[Koné *et al.*, 2011] Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Event-Based MILP Models for Resource-Constrained Project Scheduling Problems. *Computers & Operations Research*, 38:3–13, January 2011.

[Schutt *et al.*, 2013] Andreas Schutt, Thibaut Feydy, and Peter J Stuckey. Explaining time-table-edge-finding propagation for the cumulative resource constraint. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 234–250. Springer, 2013.

[Srinivasan *et al.*, 1990] Arvind Srinivasan, Timothy Ham, Sharad Malik, and Robert K Brayton. Algorithms for discrete function manipulation. In *Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pages 92–95. IEEE, 1990.

[Szeredi and Schutt, 2016] Ria Szeredi and Andreas Schutt. Modelling and solving multi-mode resource-constrained project scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pages 483–492. Springer, 2016.

[Van Peteghem and Vanhoucke, 2014] Vincent Van Peteghem and Mario Vanhoucke. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, 235(1):62–72, 2014.

[Vilím *et al.*, 2015] Petr Vilím, Philippe Laborie, and Paul Shaw. Failure-directed search for constraint-based scheduling. In *Integration of AI and OR Techniques in Constraint Programming*, pages 437–453. Springer, 2015.

[Zhu *et al.*, 2006] Guidong Zhu, Jonathan F. Bard, and Gang Yu. A Branch-and-Cut Procedure for the Multimode Resource-Constrained Project-Scheduling Problem. *INFORMS J. on Computing*, 18(3):377–390, January 2006.