

Simplifying Pseudo-Boolean Constraints in Residual Number Systems^{*}

Yoav Fekete and Michael Codish

Department of Computer Science, Ben-Gurion University, Israel

Abstract. We present an encoding of pseudo-Boolean constraints based on decomposition with respect to a residual number system. We illustrate that careful selection of the base for the residual number system, and when bit-blasting modulo arithmetic, results in a powerful approach when solving hard pseudo-Boolean constraints. We demonstrate, using a range of pseudo-Boolean constraint solvers, that the obtained constraints are often substantially easier to solve.

1 Introduction

Pseudo-Boolean constraints take the form $a_1x_1 + a_2x_2 + \dots + a_nx_n \# k$, where $a_1, \dots, a_n$ are integer coefficients, $x_1, \dots, x_n$ are Boolean literals (i.e., Boolean variables or their negation), $\#$ is an arithmetic relation ($\leq$, $<$, $=$, etc.), and k is an integer. Pseudo-Boolean constraints are well studied and arise in many different contexts, for example in verification [9] and in operations research [7]. Typically we are interested in the satisfiability of a conjunction of Pseudo-Boolean constraints, often in conjunction with additional CNF clauses involving (among others also) Boolean variables from the Pseudo-Boolean constraints. Since 2005 there is a series of Pseudo-Boolean Evaluations [16] which aim to assess the state of the art in the field of Pseudo-Boolean constraint solvers.

There are variety of types of pseudo-Boolean constraint solvers. Most of these either encode constraints directly to SAT and then use a SAT solver to solve them, or else adapt methods applied in SAT solvers, such as conflict analysis and constraint learning, to deal directly with pseudo-Boolean constraints.

From the encoding based solvers, many are designed in terms of a BDD (Binary Decision Diagram) interpretation. These are typically the fastest when applicable. However, in some cases BDD based encodings are exponential in size (this is unavoidable unless $\text{NP} = \text{co-NP}$) and lead to memory problems in the encoding phase (even before solving) [2]. `MINISAT+` [14] provides solvers based on three different encoding techniques. In addition to the BDD based encoding, two alternatives are provided and applied when the BDD encoding is too large: one based on binary adders and the other on a mixed radix representation and unary sorting networks. The approach based on adders results in the most concise encodings, but it has the weakest propagation properties and often leads to high SAT solving times. `Sat4j` [6] provides two different categories of direct pseudo-Boolean solvers: one based on resolution, and the other on cutting planes.

^{*} Supported by the Israel Science Foundation, grant 182/13.

This paper introduces a new encoding technique for solving pseudo-Boolean constraints. It is based on the well-studied residual number system and its correctness is justified by the Chinese Remainder Theorem [12].¹ Our approach is based on a transformation of pseudo-Boolean constraints to equi-satisfiable conjunctions of pseudo-Boolean constraints with smaller coefficients. As such, it can be coupled with any solver for pseudo-Boolean constraints. Each constraint in the transformed system concerns the same Boolean variables as the original, and possibly a few additional. To evaluate our approach we illustrate the impact of applying the transformation, comparing for a given solver, performance with and without the proposed transformation. We consider both types of solvers: based on SAT encodings as implemented in MINISAT⁺, and based on a direct approach as implemented in Sat4j [6].

In a residual number system, given a (finite) base ρ of co-prime natural values, all natural numbers $n < \pi_\rho$, where π_ρ denotes the multiplication of the elements in ρ , are uniquely represented in terms of their residuals with respect to the elements of ρ . In our approach, given a pseudo-Boolean constraint C , we select a suitable residual base $\rho = \{m_1, \dots, m_k\}$, and transform c to an equi-satisfiable conjunction of pseudo-Boolean constraints $\{C_1, \dots, C_k\}$ in which the coefficients are smaller than those in C . The choice of the residual base ρ for the given constraint C is an important factor in the quality of the encoding and an important attribute of our contribution.

Another major theme in our approach is “bit-blasting”. This is a term referring to the encoding of finite-precision arithmetic to a CNF formula based on its bit-level representation (see for example [10]). The traditional notion of bit-blasting assumes a binary representation of integers. Recent research, e.g. [18, 17], demonstrates the potential when considering unary representations for bit-blasting. In this paper we propose to consider also mixed radix representations.

The mixed radix number system is also at the heart of the sorting network encodings applied in MINISAT⁺. In brief, the sum on the left side of the pseudo-Boolean constraint C is represented in terms of a (finite) mixed radix base $\mu = \{r_1, \dots, r_k\}$, and a network of $k+1$ sorting networks is constructed, so that there is one sorting network to represent the value of each digit in the representation. For details, see [14]. However these are not essential for our presentation. What is important, is that the quality of this encoding is governed by the choice of the mixed radix base μ . The objective is to find a base which minimizes the size of the sorting networks. Roughly this means, to minimize the sum of the digits of the coefficients in C when represented in the number system determined by μ . For experimental evaluation, when considering pseudo-Boolean constraint solvers based on sorting networks, we apply an improved version of MINISAT⁺, which we call MINISAT⁺⁺, that is configured to encode the constraints to SAT using an optimal mixed radix base. We note that the standard MINISAT⁺ tool does not always apply an optimal base and ours is based on the application of the optimal base algorithm described in [11].

¹ The Chinese Remainder Theorem is introduced in a fifth century book by Chinese mathematician Sun Tzu. We cite the more accessible textbook presentation.

Our technique builds on recent work by Aavani *et al.*[1] where the authors also propose to apply a residual number system to encode pseudo-Boolean constraints. Similar to their work, we first apply the Chinese Remainder Theorem to derive a conjunction of “pseudo-Boolean modulo” constraints and then consider how to encode these constraints. Our work differs in two main points. First, we define new criteria on the selection of the residual base, and make an optimal choice with respect to these criteria. Second, we bit-blast pseudo-Boolean modulo constraints to standard pseudo-Boolean constraints instead of encoding them to SAT. We consider unary, binary and mixed-radix bit-blasts. The resulting constraints can be solved using any pseudo-Boolean constraint solver. We illustrate the impact of these contributions when solving hard pseudo-Boolean constraints.

We assume that pseudo-Boolean constraints are in Pseudo-Boolean normal form [5] and follow [1] (Proposition 1) to further assume that constraints are also normalized so that the relation between the sum on the left hand side and the constant on the right hand side is an equality.

2 Preliminaries

Underlying the techniques presented in this paper are concepts concerning the residual- and the mixed-radix- number systems. In this section we provide the basic definitions for these number systems and state the corresponding optimal base problems: the problem of selecting an “optimal” base for a given encoding task. Both tasks concern finding a suitable representation for the multiset of coefficients occurring on the left side in a given pseudo-Boolean constraint. The following introduces two pseudo-Boolean constraints as running examples.

Example 1. We demonstrate two pseudo-Boolean constraints:

$$621x_1 + 459x_2 + 323x_3 + 7429x_4 = 7888 \quad (1)$$

$$20x_1 + 493x_2 + 561x_3 + 1071x_4 = 1942 \quad (2)$$

Constraint (1) involves the coefficients $s_1 = \{323, 459, 621, 7429\}$ and is satisfiable as illustrated by the assignment $\{x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1\}$. Constraint (2) involves the coefficients $s_2 = \{20, 493, 561, 1071\}$ and is not satisfiable.

Mixed radix base: A finite mixed radix base, $\mu = \langle r_0, r_1, \dots, r_{k-1} \rangle$, is a sequence of k integer values, called *radices*, where for each radix, $r_i > 1$. A number in base μ has $k + 1$ digits (possibly zero padded) and takes the form $\mathbf{d} = \langle d_0, \dots, d_k \rangle$. For each digit, except the last, $0 \leq d_i < r_i$ ($i < k$). There is no bound on the value of the most significant digit, $d_k \geq 0$. The integer value associated with $\mathbf{d}$ is $v = d_0w_0 + d_1w_1 + d_2w_2 + \dots + d_kw_k$ where $\langle w_0, \dots, w_k \rangle$, called the *weights* for μ , are defined by $w_0 = 1$ and for $0 \leq i < k$, $w_{i+1} = w_i r_i$.

Example 2. Consider Constraint (1) with coefficients $s_1 = \{323, 459, 621, 7429\}$ represented in bases $\mu_1 = \langle 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 \rangle$ (a binary base) and $\mu_2 = \langle 17, 3, 3, 2, 2, 2, 2, 3 \rangle$. The columns in the two tables (below) correspond to the digits in the corresponding mixed radix representations.

$\mu_1 = \langle 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 \rangle$													
$s_1 \backslash \mathbf{d}$	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}
323	1	1	0	0	0	0	1	0	1	0	0	0	0
459	1	1	0	1	0	0	1	1	0	0	0	0	0
621	1	0	1	1	0	1	1	0	0	1	0	0	0
7429	1	0	1	0	0	0	0	0	1	0	1	1	1

$\mu_2 = \langle 17, 3, 3, 2, 2, 2, 2, 2, 3 \rangle$												
$s_1 \backslash \mathbf{d}$	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8			
323	0	1	0	0	1	0	0	0	0			
459	0	0	0	1	1	0	0	0	0			
621	9	0	0	0	0	1	0	0	0			
7429	0	2	1	0	0	0	0	0	1			

Let $\mu = \langle r_0, r_1, \dots, r_{k-1} \rangle$ be a mixed radix base. We denote by $v_{(\mu)} = \langle d_0, d_1, \dots, d_k \rangle$ the representation of a natural number v in base μ . We let $ms(\mathbb{N})$ denote the set of finite non-empty multisets of natural numbers. For $S \in ms(\mathbb{N})$ with n elements we denote the $n \times (k+1)$ matrix of digits of elements from S in base μ as $S_{(\mu)}$. So, the i^{th} row in $S_{(\mu)}$ is the vector $S(i)_{(\mu)}$. We denote the sum of the digits in $S_{(\mu)}$ by $|S|_{\mu}^{sod}$. When clear from the context we omit the superscript. Observe in the context of Example 2 that $|s_1|_{\mu_1}^{sod} = 22$ and $|s_1|_{\mu_2}^{sod} = 18$. These are the sums of the mixed radix digits in the corresponding tables.

The following optimal base problem is introduced in [11]. That work introduces an algorithm to find an optimal base and demonstrates that when solving pseudo-Boolean constraints encoded as networks of sorting networks it is beneficial to prefer a construction defined in terms of an optimal base.

Definition 1. Let $S \in ms(\mathbb{N})$. We say that a mixed radix base μ is an optimal base for S , if for all bases μ' , $|S|_{\mu}^{sod} \leq |S|_{\mu'}^{sod}$. The corresponding optimal base problem is to find an optimal base μ for S .

Residual number systems: An RNS base is a sequence, $\rho = \langle m_1, \dots, m_k \rangle$, of co-prime integer values, called *moduli*, where for each modulo, $m_i > 1$, and we denote $\pi_{\rho} = m_1 \times \dots \times m_k$. Given the base ρ every natural number $n < \pi_{\rho}$ is uniquely represented as $n_{(\rho)} = \langle (n \bmod m_1), \dots, (n \bmod m_k) \rangle$. That this is a unique representation is a direct consequence of the Chinese remainder theorem.

Let $\rho = \langle m_1, \dots, m_k \rangle$ be an RNS base and consider natural numbers a, b and $c = a \text{ op } b$ where $\text{op} \in \{+, *\}$ and such that $a, b, c < \pi_{\rho}$. From the Chinese remainder theorem, $c_{(\rho)} = \langle (a_1 \text{ op } b_1) \bmod m_1, \dots, (a_k \text{ op } b_k) \bmod m_k \rangle$. This implies that one can implement operations of addition and multiplication on RNS representations with no carry between the RNS digits. In particular, this implies that corresponding CNF encodings based on RNS representations result in a formula with small depth. The downside of the RNS representation is that encoding other arithmetic operations as well as encoding the comparison of RNS representations (except equality) is difficult.

The following two examples illustrate the basic idea underlying the application of an RNS to the encoding of pseudo-Boolean constraints. These examples also illustrate the first phase in the approach suggested in [1].

Example 3. Consider the RNS bases $\rho_1 = \langle 2, 3, 5, 7, 11, 13 \rangle$ and $\rho_2 = \langle 17, 3, 19, 23 \rangle$ and again Constraint (1). Base ρ_1 , consists of the first 6 prime numbers and ρ_2 consists of carefully selected prime numbers as explained in the sequel. From the Chinese remainder theorem we know that solving Constraint (1) is the same as solving the system of equivalences that derive from its residuals modulo the given base elements. For ρ_1 this means solving the system of 6 modulo-equivalences on the left, and for ρ_2 , the 4 modulo-equivalences on the right:

$$\begin{array}{ll}
1x_1 + 1x_2 + 1x_3 + 1x_4 \equiv 0 \pmod{2} & 9x_1 \equiv 0 \pmod{17} \\
2x_3 + 1x_4 \equiv 1 \pmod{3} & 2x_3 + 1x_4 \equiv 1 \pmod{3} \\
1x_1 + 4x_2 + 3x_3 + 4x_4 \equiv 3 \pmod{5} & 13x_1 + 3x_2 \equiv 3 \pmod{19} \\
5x_1 + 4x_2 + 1x_3 + 2x_4 \equiv 6 \pmod{7} & 22x_2 + 1x_3 \equiv 22 \pmod{23} \\
5x_1 + 8x_2 + 4x_3 + 4x_4 \equiv 1 \pmod{11} & \\
10x_1 + 4x_2 + 11x_3 + 6x_4 \equiv 10 \pmod{13} &
\end{array}$$

In both cases the coefficients in the modulo-equivalences are smaller than those in the original constraint. In the sequel we will see how to encode modulo-equivalences in terms of pseudo-Boolean constraints. Meanwhile, observe that the system on the right is extremely easy to solve. The first equivalence implies that $x_1 = 0$, then the third that $x_2 = 1$, then the fourth that $x_3 = 0$, and finally the second that $x_4 = 1$.

Example 4. Consider again Constraint (2) and the RNS bases $\rho_1 = \langle 2, 3, 5, 7, 11 \rangle$ and $\rho_2 = \langle 17, 3, 2, 5, 7 \rangle$. Base, ρ_1 consists of the first 5 prime numbers and ρ_2 consists of carefully selected prime numbers as explained in the sequel. Just as in Example 3, the given constraint is equivalent to the system of modulo-equivalences obtained with respect to either of the two given bases. The residuals with respect to ρ_1 are given on the left and with respect to ρ_2 , on the right:

$$\begin{array}{ll}
1x_2 + 1x_3 + 1x_4 \equiv 0 \pmod{2} & 3x_1 \equiv 4 \pmod{17} \\
2x_1 + 1x_2 \equiv 1 \pmod{3} & 2x_1 + 1x_2 \equiv 1 \pmod{3} \\
3x_2 + 1x_3 + 1x_4 \equiv 2 \pmod{5} & 1x_2 + 1x_3 + 1x_4 \equiv 0 \pmod{2} \\
6x_1 + 3x_2 + 1x_3 \equiv 3 \pmod{7} & 3x_2 + 1x_3 + 1x_4 \equiv 2 \pmod{5} \\
9x_1 + 9x_2 + 4x_4 \equiv 6 \pmod{11} & 6x_1 + 3x_2 + 1x_3 \equiv 3 \pmod{7}
\end{array}$$

Observe that the system on the right is easy to solve as the first equivalence (modulo 17) is not satisfiable, and hence also the original constraint.

The next example illustrates that RNS decomposition is not arc consistent.

Example 5. Consider the constraint $7x_1 + 3x_2 + 1x_3 = 5$ and the partial assignment $\alpha = \{x_1 = \text{true}\}$ which renders the constraint false. Now consider the RNS decomposition given the base $B = \langle 3, 5 \rangle$ resulting in the two modulo constraints $1x_1 + 1x_3 \equiv 2 \pmod{3}$ and $2x_1 + 3x_2 + 1x_3 \equiv 0 \pmod{5}$. One cannot deduce the conflict with partial assignment α on the two modulo constraints.

We conclude this section with a statement of two questions that this paper addresses: (1) Given a pseudo-Boolean constraint, how to best choose an RNS base for its encoding? In each of Examples 3 and 4, one of the bases considered results in a system of modulo-equivalences that is easy to solve. But how to select such a base? (2) Given a system of pseudo-Boolean modulo-equivalences, such as those derived in Examples 3 and 4, how to solve or encode them?

To answer these questions Section 3 proposes an encoding of pseudo-Boolean modulo-equivalences to pseudo-Boolean constraints. As such, we can apply any pseudo-Boolean constraint solver to solve the pseudo-Boolean modulo equivalences. Section 4 proposes two new measures on RNS bases, introduces corresponding optimal base problems, and provides simple algorithms to select optimal RNS bases with respect to these measures for a given pseudo-Boolean constraint.

3 The RNS Pseudo-Boolean Transformation

Let $C = (a_1x_1 + a_2x_2 + \dots + a_nx_n = c)$ be a pseudo-Boolean constraint with $S = \langle a_1, \dots, a_n \rangle$ the multiset of its coefficients. We say that ρ is an RNS base for C if it is an RNS base and $\pi_\rho > \Sigma S$. In this case, we also say that ρ is a base for (sums from) S . We say that ρ is non-redundant for C if for every $p \in \rho$, $p < \max(S)$; and no prefix of ρ is also a base for C . Not all pseudo-Boolean constraints have a non-redundant RNS base, as is the case when the sum of the elements in S is larger than the multiplication of the primes smaller than $\max(S)$. For example, cardinality constraints do not have a non-redundant RNS base. We assume that C has a non-redundant base, and we describe an encoding of C to an equivalent system of “smaller” pseudo-Boolean constraints derived from ρ . In the following we denote $\rho = \langle p_1, \dots, p_m \rangle$, $a_{ij} = a_i \bmod p_j$, and $c_j = c \bmod p_j$ for $1 \leq i \leq n$ and $1 \leq j \leq m$. We detail the encoding in three phases.

The first phase, demonstrated in Examples 3 and 4, is formalized as Equation (3). The pseudo-Boolean constraint on the left is equivalent to the conjunction of pseudo-Boolean modulo-equivalences, on the right. Stated as a transition: to encode the left side, it is sufficient to encode the right side.

$$\sum_{i=1}^n a_i x_i = c \quad \Leftrightarrow \quad \bigwedge_{j=1}^m \left(\sum_{i=1}^n a_{ij} x_i \equiv c_j \pmod{p_j} \right) \quad (3)$$

In the second phase, we apply the definition of integer division modulo p_j to replace the equivalence modulo p_j on the left side with an equality, on the right side, introducing integer variables t_j and constants $c_j < p_j$ for $1 \leq j \leq m$.

$$\bigwedge_{j=1}^m \left(\sum_{i=1}^n a_{ij} x_i \equiv c_j \pmod{p_j} \right) \quad \Leftrightarrow \quad \bigwedge_{j=1}^m \left(\sum_{i=1}^n a_{ij} x_i = t_j p_j + c_j \right) \quad (4)$$

The right side of Equation (4) is not yet a system of pseudo-Boolean constraints due to the integer variables t_j ($1 \leq j \leq m$). However, these variables take values from a finite domain, as we have, recalling that $a_{ij} < p_j$ and $c_j < p_j$:

$$0 \leq t_j = \left\lfloor \frac{(\sum a_{ij} x_i) - c_j}{p_j} \right\rfloor \leq \left\lfloor \frac{(\sum a_{ij}) - c_j}{p_j} \right\rfloor \leq n \quad (5)$$

In the third phase we bit-blast the integer variables in the right side of Equation (4) to obtain a proper pseudo-Boolean constraint. We consider three alternative bit-blasting techniques: binary, unary, and mixed radix and obtain Equation (6) where $\llbracket t_j \rrbracket_{bb}$ indicates the corresponding bit-blast of integer variable t_j with bb equal to **b** (binary), **u** (unary), or μ (mixed radix). Once substituting $\llbracket t_j \rrbracket_{bb}$, the right side of Equation (6) is a pseudo-Boolean constraint.

$$\bigwedge_{j=1}^m \left(\sum_{i=1}^n a_{ij} x_i = p_j t_j + c_j \right) \quad \Leftrightarrow \quad \bigwedge_{j=1}^m \left(\sum_{i=1}^n a_{ij} x_i = p_j \llbracket t_j \rrbracket_{bb} + c_j \right) \quad (6)$$

For all three bit-blasting alternatives, assume that t is an integer variable taking values in the range $[0, \dots, n]$.

$$\begin{array}{lll}
\llbracket t \rrbracket_b = \sum_{i=0}^k 2^i y_i & \llbracket t \rrbracket_u = \llbracket t \rrbracket_u = \sum_{i=1}^n y_i & \llbracket t \rrbracket_\mu = \sum_{i=0}^k w_i \times \Sigma \mathbf{y}_i \\
\text{(a) binary} & \text{(b) unary} & \text{(c) mixed-radix}
\end{array}$$

Fig. 1. Bit-blasting the integer variable t .

binary bit-blast: Let $k = \lfloor \log_2 n \rfloor$ and let $y_0, \dots, y_k$ be fresh Boolean variables corresponding to the binary representation of integer variable t . The equation in Figure 1(a) specifies the binary bit-blast of t and substituting in Equation 6 results in a system of pseudo-Boolean constraints introducing an order of $m \log n$ fresh Boolean variables with integer coefficients of value up to $\max(\rho) \cdot n$.

unary bit-blast: In the order encoding $[4, 3, 13]$, an integer variable $0 \leq t \leq n$ is represented in n bits, $[y_1, \dots, y_n]$ such that $y_1 \geq y_2 \geq \dots \geq y_n$. For example, if $n = 5$ and $t = 3$, then the representation is $[1, 1, 1, 0, 0]$. Let $y_1, \dots, y_n$ be fresh Boolean variables corresponding to the unary, order encoding, representation of the integer variable t . Note that with the unary encoding we add also binary clauses to keep the order between the bits ($y_1 \geq y_2 \geq \dots \geq y_n$). The equation in Figure 1(b) specifies the unary bit-blast of integer variable t . Substituting in Equation (6) results in a system of pseudo-Boolean constraints introducing an order of $m \cdot n$ fresh Boolean variables with integer coefficients of value up to $\max(\rho)$. It also introduces an order of $m \cdot n$ binary CNF clauses.

mixed-radix bit-blast: Let $\mu = \langle r_0, r_1, \dots, r_{k-1} \rangle$ be a mixed radix base, let $\langle w_0, \dots, w_k \rangle$ be the weights for μ , and let $\mathbf{y}_0, \dots, \mathbf{y}_k$ be a sequence of bit-vectors representing, in the order encoding, the values of the digits of the mixed radix bit-blast of t . For $0 \leq i < k$, the vector $\mathbf{y}_i$ consists of r_i fresh Boolean variables, and $\mathbf{y}_k$ consists of $\lfloor n \div w_k \rfloor$ fresh Boolean variables. Denoting by $\Sigma \mathbf{y}_i$ the sum of the Boolean variables in bit-vector $\mathbf{y}_i$, the equation in Figure 1(c) specifies the mixed radix bit-blast of integer variable t . Substituting in Equation 6 results in a system of pseudo-Boolean constraints introducing at most $m \cdot n$ fresh Boolean variables with integer coefficients. It also introduces at most $m \cdot n$ binary CNF clauses to encode the order between the bits in the corresponding bit-vectors $\mathbf{y}_i$. For a given pseudo-Boolean modulo-constraint the choice of a mixed radix base to bit-blast it to a pseudo-Boolean constraint is left to a brute-force search algorithm which aims to minimize the size of the representation of the resulting pseudo-Boolean constraint. This works well in practice as the number of potential bases is not large.

The following example demonstrates the transformation with unary bit-blasting. It also illustrates that in practice, the domain of the integer variables t_j of Equation 5 is tighter than the prescribed $0 \leq t_j \leq n$.

Example 6. Consider the bases, $\rho_1 = \langle 2, 3, 5, 7, 11, 13 \rangle$ and $\rho_2 = \langle 17, 3, 19, 23 \rangle$ and again Constraint (1). Recall from Example 3 the derived pseudo-Boolean equivalences with respect to the two bases. To detail the RNS decomposition of Constraint (1) to pseudo-Boolean constraints we illustrate the (tighter) domains of the integer variables t_j of Equation (5) where $0 \leq t_j \leq \left\lfloor \frac{(\sum a_{ij}) - c_j}{p_j} \right\rfloor$.

Substituting in Equation (5), we have the following bounds for t_j with respect to ρ_1 , on the left and with respect to ρ_2 , on the right.

$$\begin{aligned}
0 \leq t_1 &\leq \lfloor (1+1+1+1-0)/2 \rfloor = 2 & 0 \leq t_1 &\leq \lfloor (9-0)/17 \rfloor = 0 \\
0 \leq t_2 &\leq \lfloor (2+1-1)/3 \rfloor = 0 & 0 \leq t_2 &\leq \lfloor (2+1-1)/3 \rfloor = 0 \\
0 \leq t_3 &\leq \lfloor (1+4+2+4-3)/5 \rfloor = 0 & 0 \leq t_3 &\leq \lfloor (13+3-3)/19 \rfloor = 0 \\
0 \leq t_4 &\leq \lfloor (5+4+1+2-6)/7 \rfloor = 0 & 0 \leq t_4 &\leq \lfloor (22+1-22)/23 \rfloor = 0 \\
0 \leq t_5 &\leq \lfloor (5+8+4+4-1)/11 \rfloor = 1 \\
0 \leq t_6 &\leq \lfloor (10+4+11+6-10)/13 \rfloor = 1
\end{aligned}$$

Based on these bounds we obtain the following encodings to pseudo-Boolean constraints using unary bit blasting with respect to ρ_1 , on the left and with respect to ρ_2 , on the right. Here, $[y_{11}, y_{12}]$ represents t_1 , $[y_{31}]$ represents t_3 , $[y_{51}]$ represents t_5 , and $[y_{61}]$ represents t_6 , all in the order encoding.

$$\begin{aligned}
1x_1 + 1x_2 + 1x_3 + 1x_4 - 2y_{11} - 2y_{12} &= 0 & 9x_1 &= 0 \\
2x_3 + 1x_4 &= 1 & 2x_3 + 1x_4 &= 1 \\
1x_1 + 4x_2 + 3x_3 + 4x_4 - 5y_{31} &= 3 & 13x_1 + 3x_2 &= 3 \\
5x_1 + 4x_2 + 1x_3 + 2x_4 &= 6 & 22x_2 + 1x_3 &= 22 \\
5x_1 + 8x_2 + 4x_3 + 4x_4 - 11y_{51} &= 1 \\
10x_1 + 4x_2 + 11x_3 + 6x_4 - 13y_{61} &= 10
\end{aligned}$$

4 Choosing an RNS base

In [1], the authors consider two different options for an RNS base when encoding a given pseudo-Boolean constraint.

Definition 2 (primes and prime powers RNS bases [1]). *Let $S \in ms(\mathbb{N})$, and for any k let $p_1, p_2, \dots, p_k$ denote the sequence of the first k prime numbers. The **primes** RNS base is $\rho_p = \{p_1, p_2, \dots, p_k\}$ where k is the smallest integer such that $\pi_{\rho_p} > \Sigma S$. The **prime powers** RNS base is $\rho_{pp} = \{p_1^{n_1}, p_2^{n_2}, \dots, p_k^{n_k}\}$ where for each i , n_i is the smallest integer such that $\log_2(\Sigma S) \leq p_i^{n_i}$ and k is the smallest integer such that $\pi_{\rho_{pp}} > \Sigma S$.*

Definition 2 determines an RNS base focusing only on the maximal number that needs to be represented in the base (the sum of the coefficients). As already hinted at in Examples 3 and 4, we maintain that the choice of the RNS base should be geared towards the specific coefficients occurring in the constraint. We consider two additional options to select an RNS base. The first option focuses on the number of variables in the derived pseudo-Boolean modulo constraints, and the second on the number of clauses in their resulting CNF encoding.

For a given pseudo-Boolean constraint, the following definition specifies an RNS base which minimizes the variable occurrences in the pseudo-Boolean modulo constraints resulting from the RNS transformation specified in Equation (3). So we want as many as possible of the base elements to divide as many as possible of the coefficients in the constraint. Let $f_S(p)$ denote the number of elements from $S \in ms(\mathbb{N})$ that are divided by the (prime) number p and for $\rho = \langle p_1, \dots, p_k \rangle$ denote $f_S(\rho) = \langle f_S(p_1), \dots, f_S(p_k) \rangle$.

Definition 3 (optV RNS base). RNS base ρ for $S \in ms(\mathbb{N})$ is **optV**, if ρ is a non-redundant RNS base for S ; and the tuple $f_S(\rho)$ is maximal (in the lexicographic order) over all of the non-redundant RNS bases for S .

Example 7. Consider the multiset $S = \langle 621, 459, 323, 7429 \rangle$ of coefficients from Constraint (1). The following tables illustrate the representations of the elements of S in the three RNS bases defined above, highlighting in the bottom line(s) the number of elements divided by each base element.

ρ_P	2	3	5	7	11	13
s						
323	1	2	3	1	4	11
459	1	0	4	4	8	4
621	1	0	1	5	5	10
7429	1	1	4	2	4	6
#divided	0	2	0	0	0	0

ρ_{PP}	16	27	25
s			
323	3	26	23
459	11	0	9
621	13	0	21
7429	5	4	4
#divided	0	2	0

ρ_{optV}	17	3	19	23
s				
323	0	2	0	1
459	0	0	3	22
621	9	0	13	0
7429	0	1	0	0
#divided	3	2	2	2

The following theorem defines an algorithm to construct an **optV** RNS base.

Theorem 1 (finding an optV RNS base). Let $S \in ms(\mathbb{N})$ be such that it has a non-redundant RNS base and let $P = \langle p_1, \dots, p_n \rangle$ be the sequence of distinct primes smaller than $\max(S)$, sorted so that for all $0 < i < n$, $\langle f_S(p_i), p_i \rangle \succ_{lex} \langle f_S(p_{i+1}), p_{i+1} \rangle$. Let $\rho = \langle p_1, \dots, p_k \rangle$ ($k \leq n$) be the shortest prefix of P such that $\pi_\rho > \Sigma S$. Then ρ is an **optV** RNS base for S .

Example 8. Consider again the multiset S from Example 7. In the context of the statement of Theorem 1 we have $P = \langle 17, 3, 19, 23, 2, 5, 7, 11, 13, \dots, 7417 \rangle$ and $\rho = \langle 17, 3, 19, 23 \rangle$.

Proof. (of Theorem 1) First note that ρ as constructed in the statement of the theorem is well-defined: That S has a non-redundant RNS base implies that $\pi_P > \Sigma S$ and hence there exists $k \leq n$ such that $\rho = \langle p_1, \dots, p_k \rangle$ is a non-redundant RNS base for S . Assume falsely that there exists another non-redundant RNS base $\rho' = \langle q_1, \dots, q_l \rangle$ such that $f_S(\rho) \prec_{lex} f_S(\rho')$. It follows that neither ρ nor ρ' are a prefix of the other (otherwise contradicting that both are non-redundant) which implies that there exists an index $i \leq \min(k, l)$ such that $p_i \neq q_i$ and for $0 < j < i$, $p_j = q_j$. Namely, ρ and ρ' have a common prefix of length $i-1$. Assume, without loss of generality, that there is no choice of ρ' which has a longer common prefix with ρ . Otherwise, take for ρ' the choice with a longer common prefix (observe that the value of i is bound from above by k). From the construction of ρ we have that $f_S(p_i) \geq f_S(q_i)$ (otherwise we would have selected q_i in the construction of ρ). From the selection of ρ' such that $f(\rho) \prec_{lex} f(\rho')$ we have that $f_S(p_i) \leq f_S(q_i)$. It follows that $f_S(p_i) = f_S(q_i)$. Again from the construction of ρ (this time because $f_S(p_i) = f_S(q_i)$) we have that $p_i < q_i$. Now consider $\rho'' = \langle p_1, \dots, p_i, q_{i+1}, \dots, q_l \rangle$ and observe that $\pi_{\rho''} < \pi_{\rho'}$ because all elements are the same except at position i where $p_i < q_i$. If ρ'' is a base for S , then it is non-redundant and we have a contradiction because $f_S(\rho) \prec f_S(\rho') = f_S(\rho'')$ and ρ and ρ'' have a common prefix of length i . If ρ'' is not a base for S then it

follows that $\rho''' = \langle p_1, \dots, p_i, q_{i+1}, \dots, q_l, q_i \rangle$ is a non-redundant base for S , and we have a contradiction because $f_S(\rho) \prec f_S(\rho') \preceq_{lex} f_S(\rho''')$ and ρ and ρ''' have a common prefix of length i . $\square$

We now consider an RNS base which aims to minimize the number of CNF clauses resulting from the eventual encoding of the modulo-constraints specified in Equation (3). The size of the CNF encoding is determined not only by the RNS base but also by the bit-blasting technique and the subsequent encoding technique for the resulting pseudo-Boolean constraints of Equation (6).

Given a given pseudo-Boolean constraint C with coefficients $S \in ms(\mathbb{N})$, for each prime number $p < \max(S)$, let $h_C^\tau(p)$ denote the number of CNF clauses when encoding C modulo p given the encoding technique τ . Here, τ specifies the bit-blast alternative adopted in Equation (6) and the subsequent encoding to CNF technique applied to the resulting pseudo-Boolean constraints. For a specific choice of τ , an **optC** RNS base is one that minimizes the sum of $h_C^\tau(p)$ (over the elements of ρ).

Definition 4 (optC RNS base). *RNS base ρ for pseudo-Boolean constraint C with coefficients $S \in ms(\mathbb{N})$ using encoding technique τ is **optC**, if ρ is a non-redundant RNS base for S and the value of $\sum_{p \in \rho} h_C^\tau(p)$ is minimal.*

In our implementation the search for an **optC** RNS base is brute-force. Given a pseudo-Boolean constraint with coefficients S and an encoding technique τ , for each prime $p < \max(S)$ we encode the derived bit-blast of $C \bmod p$ to obtain the value of $h_C^\tau(p)$. We then seek a set of primes with sufficiently large multiplication and minimal sum of $h_C^\tau(p)$ values. Practice shows that the search time is negligible. However, finding an efficient way to perform this search is an interesting research problem on its own.

5 Experiments

Our objective is to illustrate the benefit of our proposed RNS based transformation comparing, for a variety of different “off-the-shelf” solvers, the time to solve an instance using or not using the proposed RNS decomposition. We emphasize that the intention is not to compare between the solvers, nor to present results with full range of other available solvers.

We have selected 4 basic solvers: the Sat4j solver based on resolution [6] (the Sat4j solver based on cutting planes was consistently slower for our benchmarks); and three solvers based on MINISAT⁺ [14], which apply encoding techniques based on: BDDs, binary adders, and sorting networks. For the encodings based on sorting networks, where MINISAT⁺ employs a mixed radix representation, we apply an improved version of the tool which we call MINISAT⁺⁺. Improvements include: the application of an optimal mixed radix base as described in [11] (the base selected by MINISAT⁺ is often not optimal); and fixing several bugs in MINISAT⁺ such as the one reported in [1] (in the section on experiments).

instance (s/u)	without RNS				optV RNS			ρ_P RNS	
	Adders	Sorters	BDDs	S4J.R	U $\times$ Sorters	B $\times$ BDDs	MR $\times$ Adders	PBM	B $\times$ BDDs
BI_cuww4 (s)	∞	∞	0.5	∞	0.1	14.9	98.3	∞	∞
BI_prob10 (s)	326.5	351.5	5.9	∞	412.3	26.6	996.5	∞	147.0
MI_cuww1 (s)	0.0	0.2	0.2	0.1	7.6	∞	1.1	∞	∞
MI_cuww2 (s)	∞	∞	0.8	∞	245.8	∞	∞	∞	∞
MI_cuww3 (s)	∞	106.9	0.5	0.5	3.9	1.87	0.1	∞	∞
MI_cuww5 (s)	27.8	∞	1.1	∞	∞	∞	∞	∞	∞
MI_prob1 (s)	4.5	0.4	4.3	303.0	13.9	100.5	154.3	∞	127.2
MI_prob2 (s)	1.8	∞	4.9	49.5	0.5	∞	∞	∞	∞
MI_prob3 (u)	∞	∞	0.2	∞	∞	∞	∞	∞	∞
MI_prob4 (s)	602.3	14.4	1.7	133.2	4.4	21.3	9.6	∞	∞
MI_prob5 (s)	44.8	17.14	6.8	∞	3.8	60.9	863.2	∞	388.5
MI_prob6 (s)	18.7	∞	3.0	508.1	100.2	∞	113.2	∞	∞
MI_prob7 (s)	∞	163.7	8.1	109.1	17.3	27.4	118.9	∞	∞
MI_prob8 (s)	∞	13.7	8.0	440.9	494.2	∞	751.5	∞	∞

Table 1. PB12 with various solvers — solving times (seconds) with 1800 sec. timeout.

In addition to the four basic solvers (without RNS), we consider the full range of RNS based solvers based on these which are parametrized by three choices: The selection of the RNS base, the selection of technique to bit-blast modulo arithmetic, and the selection of the underlying solver (from the four above mentioned) applied to the result of the RNS decomposition. We also consider the RNS based solver described in [1] which we call PBM. All solvers run on the same machine and apply the same underlying MINISAT solver (where relevant).

We consider two benchmark suites. The first suite, which we call PB12, consists of the 28 instances from the DEC-INT-LIN category of the Pseudo-Boolean Competition 2012 [15] (we show results for the 14 largest instances). Each instance consists of a single pseudo-Boolean constraint (with equality) and is categorized depending on the size of its coefficients: BI (big integers), MI (medium integers) or SI (small integers). The benchmark suite consists of both satisfiable and unsatisfiable instances. The largest constraint involves 106 coefficients, and the largest coefficient is 106,925,262. The instance name indicates to which of the three categories the instance belongs to (see for example Table 1).

For the second suite, which we call RNP, we follow [1] and consider a collection of randomly generated number partition problems. Here, an (n, L) -instance is a set of integer values $S = \{a_1, \dots, a_n\}$ such that each element is selected randomly in the range $0 \leq a_i < 2^L$. The objective is to determine a partition of S into two parts that sum to the same value. Each such instance is straightforward to express in terms of a single pseudo-Boolean constraint: $\sum a_i x_i = \sum a_i / 2$. We have selected instances randomly generated for $3 \leq n \leq 30$ and $3 \leq L \leq 2n$. The problem is known to have a phase transition when $L/n = 1$ [8], and so we focus on these instances. Here, the instance name indicates the values of the parameters n and L (see for example Table 2). For example, the instance 25-25-1 is instance number 1 with 25 coefficients, each involving a 25 bit random value.

All of the following tables report solving times, indicated in seconds, with a timeout of 1800 seconds. The symbol ∞ indicates that the timeout was encountered, and the symbol — indicates that memory was insufficient to construct the (BDD-based) encoding. The tables detail solving times for the 14 larger in-

instance (s/u)	without RNS				optC RNS			ρ_p RNS	
	Adders	Sorters	BDDs	S4J.R	U×BDDs	B×BDDs	MR×Sorters	PBM	B×BDDs
25-25-1 (s)	192.7	18.1	0.2	20.9	7.9	2.2	13.6	20.2	29.1
25-25-2 (u)	181.1	159.2	0.4	69.4	15.9	9.6	37.9	15.5	45.1
25-25-3 (u)	63.8	8.2	0.1	6.2	0.2	3.8	28.6	11.0	26.8
25-25-4 (u)	58.0	55.6	0.2	36.3	1.1	5.2	41.8	47.6	70.9
25-25-5 (u)	108.9	34.0	0.1	9.4	0.9	25.9	108.7	18.8	22.7
25-25-6 (u)	41.9	42.5	0.2	26.5	0.5	11.4	22.6	18.5	30.5
25-25-7 (u)	99.9	22.4	0.1	7.3	21.9	8.6	34.2	68.8	39.5
25-25-8 (u)	119.5	51.5	0.4	63.3	24.1	15.9	59.6	25.8	43.4
25-25-9 (s)	26.2	25.7	0.4	0.4	2.6	0.6	5.8	1.9	1.8
25-25-10 (s)	21.9	16.9	0.3	17.0	4.4	2.8	1.3	0.7	9.7
30-30-1 (u)	∞	∞	9.0	∞	14.3	∞	∞	1696.7	∞
30-30-2 (s)	∞	34.9	4.6	∞	867.8	994.7	∞	1006.2	653.9
30-30-3 (u)	∞	∞	8.7	∞	∞	1751.4	∞	∞	∞
30-30-4 (s)	844.8	418.6	-	∞	269.0	66.89	211.6	757.0	689.2
30-30-5 (u)	∞	∞	-	∞	52.8	∞	∞	∞	∞
30-30-6 (u)	∞	1762.3	7.8	∞	139.1	∞	∞	∞	∞
30-30-7 (u)	∞	∞	9.9	∞	1416.3	∞	∞	∞	∞
30-30-8 (u)	∞	1448.2	7.6	∞	397.3	497.9	1362.6	1346.6	∞
30-30-9 (u)	∞	∞	-	∞	982.3	721.3	1143.8	345.7	1129.1
30-30-10 (u)	∞	∞	9.8	∞	∞	1394.1	∞	∞	∞
35-35-1 (?)	∞	∞	-	∞	∞	∞	∞	∞	∞
35-35-2 (s)	∞	∞	-	∞	∞	∞	1602.9	∞	∞
35-35-3 (s)	∞	∞	-	∞	∞	∞	1152.9	∞	∞
35-35-4 (s)	∞	∞	-	∞	669.4	∞	∞	∞	∞
35-35-5 (s)	∞	333.2	-	∞	∞	∞	∞	∞	∞
35-35-6 (?)	∞	∞	-	∞	∞	∞	∞	∞	∞
35-35-7 (s)	∞	∞	-	∞	∞	∞	∞	∞	1788.1
35-35-8 (s)	∞	∞	-	∞	∞	807.5	∞	∞	∞
35-35-9 (s)	∞	∞	-	∞	427.6	∞	∞	∞	∞
35-35-10 (s)	∞	∞	-	104.0	48.9	1143.5	∞	∞	∞

Table 2. RNP with various solvers — solving times (seconds) with 1800 sec. timeout.

stances of the PB12 benchmarks and for 30 instances from the RNP benchmark. For each instance we indicate if it is satisfiable or not (s/u). Experiments run on a Quad-core Intel i5-2400 at 3.1 GHz with 4 GB RAM with at most 3GB allocated to the representation BDDs.

Tables 1 and 2 present results for the benchmark instances using a series of solvers. The first four are the basic solvers: “without RNS”. The next three are “with RNS”. In each of the two tables we portray results for one solver for each of the three types of bit-blasting combined with one of the four basic solvers. In Table 1 we consider an **optV** RNS base and in Table 2, an **optC** RNS base. Both tables indicate a comparison (in the two right columns) of our approach with the PBM solver which is also based on RNS decomposition. Here we apply their dynamic programming based transformation with the ρ_p RNS base (their best configuration for the given benchmarks) and our RNS decomposition with binary bit-blasting and an underlying BDD solver (with the same RNS base).

Tables 1 and 2 indicate the general utility of the RNS based approach. In Table 1 the BDD based encodings (without RNS) perform best. But, the RNS solvers come in second. Except for the case of BDDs, the combination of a base solver with RNS solves more instances than without. Table 2 illustrates the case where the BDD based approach (without RNS) suffers from the size of the encodings as it fails to encode most of the large instances with the allocated

instance (s/u)	with RNS:U×Sorters				with RNS:B×Sorters				with RNS:MR×Sorters			
	ρ_p	ρ_{pp}	ρ_{optV}	ρ_{optC}	ρ_p	ρ_{pp}	ρ_{optV}	ρ_{optC}	ρ_p	ρ_{pp}	ρ_{optV}	ρ_{optC}
BI_cuww4 (s)	∞	∞	0.1	∞	∞	∞	0.4	∞	∞	∞	0.5	∞
BI_prob10 (s)	∞	∞	412.3	278.9	∞	∞	6.1	55.7	949.2	∞	55.1	1523.3
MI_cuww1 (s)	1134.4	∞	7.6	3.6	∞	∞	0.6	0.2	∞	∞	6.9	3.1
MI_cuww2 (s)	∞	∞	245.9	∞	∞	∞	∞	∞	∞	∞	∞	∞
MI_cuww3 (s)	∞	∞	3.9	3.9	∞	∞	0.6	∞	∞	∞	1.2	522.4
MI_cuww5 (s)	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
MI_prob1 (s)	57.5	1423.8	13.9	11.0	241.7	∞	9.8	226.2	784.2	685.9	28.7	174.2
MI_prob2 (s)	∞	∞	0.5	11.2	∞	∞	1.5	284.1	∞	406.6	∞	41.5
MI_prob3 (u)	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
MI_prob4 (s)	∞	∞	4.4	1504.2	∞	∞	16.4	32.5	∞	∞	∞	394.7
MI_prob5 (s)	330.9	∞	3.4	3.3	1769.5	630.5	1.4	3.3	44.9	138.3	2.5	1.2
MI_prob6 (s)	137.0	26.9	100.2	∞	1772.9	∞	∞	∞	∞	∞	9.0	∞
MI_prob7 (s)	∞	593.7	17.2	∞	∞	∞	∞	∞	∞	∞	17.7	1019.8
MI_prob8 (s)	537.2	∞	494.2	18.0	∞	∞	816.6	∞	∞	∞	∞	∞

Table 3. PB12 benchmarks; Solvers based on RNS decomposition and Sorters; Impact of RNS base selection and bit-blast technique. Solving times (seconds) with 1800 sec. timeout.

3GB of RAM. For the second benchmark, overall, the RNS based solvers are superior. The comparison with the PBM solver indicates that our approach based on bit-blasting to pseudo-Boolean constraints is superior to encoding the pseudo-Boolean modulo constraints directly to CNF.

Tables 3 and 4 illustrate the impact of the choice of the RNS base and bit-blasting technique on our approach for the PB12 and RNP benchmarks respectively. In Table 3 we focus on solvers based on RNS decomposition using Sorters. In Table 4 we focus on solvers based on RNS decomposition using BDDs. For each instance we consider its RNS transformations using the four choices of RNS base: ρ_p (primes), ρ_{pp} (prime powers), ρ_{optV} (optimal with respect to number of variables), and ρ_{optC} (optimal with respect to number of clauses) and the three choices of bit-blasting techniques. We can clearly see that the optimal bases are superior. As for the bit-blasting no conclusion can be made.

Table 5 presents the “bottom-line” advantage of applying RNS decomposition. For each of the benchmark instances we indicate the best result “without” and “with” RNS decomposition. For each such result we also indicate the solver (configuration) with which this result is obtained. For RNS that means specifying the base solver, the choice of RNS base, and the choice of bit-blast for modulo arithmetic. Table 5(a), for the PB12 benchmark, indicates that (except for one instance) the RNS based techniques perform better, always with one of the optimal RNS bases introduced in Section 4. Table 5(b), for the RNP benchmark, indicates that when the BDD encoding without RNS applies, it is typically fastest. However, when the encoding with BDDs exhausts memory the RNS based approach is typically faster. Especially note that for all of the instances where the “No RNS” column does not apply BDDs (including those where there is a timeout because of the other base solvers), it is the case that the BDD technique exhausts memory. For most of these instances the fastest RNS solver is the one that applies BDDs and in this case there is no memory problem.

In a final experiment we observe that the average size of the conflict clauses reported by MINISAT⁺⁺ (using sorters) is smaller with RNS decomposition. We

instance (s/u)	with RNS: U×BDDs				with RNS: B×BDDs				with RNS: MR×BDDs			
	ρ_p	ρ_{pp}	ρ_{optV}	ρ_{optC}	ρ_p	ρ_{pp}	ρ_{optV}	ρ_{optC}	ρ_p	ρ_{pp}	ρ_{optV}	ρ_{optC}
25-25-1 (s)	34.0	62.0	0.7	7.9	29.1	50.2	1.2	1.6	26.8	68.7	0.4	14.0
25-25-2 (u)	36.9	52.0	10.5	15.9	45.1	53.1	18.5	6.6	42.4	56.4	17.0	12.7
25-25-3 (u)	33.5	45.1	7.7	0.2	26.8	66.9	10.5	2.5	28.6	48.6	15.0	4.9
25-25-4 (u)	63.5	43.9	4.2	1.1	70.9	46.3	6.9	2.3	59.7	49.2	7.1	6.5
25-25-5 (u)	26.9	81.3	39.7	0.9	22.7	77.0	41.0	13.8	38.6	74.2	41.6	19.7
25-25-6 (u)	36.5	70.4	12.1	0.5	30.5	70.2	12.7	8.9	35.5	60.3	12.9	18.5
25-25-7 (u)	38.1	53.6	9.28	21.9	39.5	45.7	8.8	6.4	38.6	45.4	9.4	12.9
25-25-8 (u)	44.5	73.3	28.36	24.1	43.4	81.9	34.0	11.4	49.3	73.6	19.9	21.7
25-25-9 (s)	6.6	27.8	1.5	2.6	1.8	15.8	0.4	0.5	2.6	7.7	0.3	2.7
25-25-10 (s)	3.3	12.4	4.6	4.5	9.7	5.0	4.2	2.1	10.1	17.9	1.5	2.2
30-30-1 (u)	∞	∞	∞	14.3	∞	∞	∞	∞	∞	∞	∞	∞
30-30-2 (s)	1543.0	∞	583.6	867.8	653.9	∞	192.1	1197.9	681.2	∞	1126.9	885.3
30-30-3 (u)	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
30-30-4 (s)	760.6	∞	112.6	269.0	689.2	∞	89.6	84.8	710.2	∞	98.0	423.7
30-30-5 (u)	∞	∞	∞	52.8	∞	∞	1751.7	∞	∞	∞	1362.8	∞
30-30-6 (u)	∞	∞	∞	139.1	∞	∞	∞	136.3	∞	∞	∞	107.9
30-30-7 (u)	∞	∞	1113.3	1416.3	∞	∞	1130.4	∞	∞	∞	1139.3	∞
30-30-8 (u)	∞	∞	1120.9	397.3	∞	∞	915.1	597.8	∞	∞	801.1	1191.1
30-30-9 (u)	1287.3	∞	604.3	982.3	1129.1	∞	623.4	843.3	1224.5	∞	580.77	476.9
30-30-10 (u)	∞	∞	∞	∞	∞	∞	1719.0	1603.2	∞	∞	1603.6	∞
35-35-1 (?)	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
35-35-2 (s)	∞	∞	1250.9	∞	∞	∞	∞	∞	∞	∞	∞	∞
35-35-3 (s)	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
35-35-4 (s)	∞	∞	∞	669.4	∞	∞	∞	∞	∞	∞	∞	∞
35-35-5 (s)	∞	∞	∞	∞	∞	∞	∞	∞	1397.1	∞	∞	∞
35-35-6 (?)	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
35-35-7 (s)	∞	∞	170.3	∞	1788.1	∞	630.7	∞	∞	∞	590.5	∞
35-35-8 (s)	∞	∞	∞	∞	∞	∞	∞	831.1	∞	∞	∞	∞
35-35-9 (s)	∞	∞	∞	427.6	∞	∞	∞	∞	∞	∞	∞	∞
35-35-10 (s)	∞	∞	∞	48.9	∞	∞	∞	1242.6	∞	∞	∞	∞

Table 4. RNP benchmarks; Solvers based on RNS decomposition and BDDs; Impact of RNS base selection and bit-blast technique. Solving times (seconds) with 1800 sec. timeout.

compute the average size of the conflict clauses considering the first k clauses generated with each technique (where k is as large as possible but required to be the same per instance for all techniques compared). We consider the average over all relevant combinations of RNS base and bit blasting options. For the PB12 benchmark, where the number of variables is much larger, the average learned clause size is 281 without decomposition, and 188 with. For the RNP benchmark the averages are 68 (without) and 62 (with). The RNS decomposition replaces an original constraint by a conjunction of “smaller” constraints, each one typically involving less of the original constraint problem variables. Conflicts derived from a single constraint will then generate smaller learned clauses and support better propagation during search.

6 Conclusion

We introduce a new encoding technique for pseudo-Boolean constraints based on RNS decomposition. The new approach, defined as a transformation from pseudo-Boolean constraints to conjunctions of pseudo-Boolean constraints, can be combined with any pseudo-Boolean constraint solver. We demonstrate that careful selection of the RNS base, fine-tuned for each specific pseudo-Boolean

(a) PB12 benchmarks

instance	No RNS		With RNS	
BI_cuww4	0.5	bdd	0.1	<code>sort.ρ_{optV}</code> (u)
BI_prob10	4.9	sort	0.5	<code>add.ρ_{optV}</code> (b)
MI_cuww1	0.1	add	0.1	<code>add.ρ_{optV}</code> (b)
MI_cuww2	0.8	bdd	0.6	<code>add.ρ_{optV}</code> (b)
MI_cuww3	0.5	bdd	0.1	<code>add.ρ_{optV}</code> (mr)
MI_cuww5	1.1	bdd	0.4	<code>sort.ρ_{optC}</code> (mr)
MI_prob1	0.4	sort	0.5	<code>sort.ρ_{optV}</code> (b)
MI_prob2	1.8	add	0.3	<code>sort.ρ_{optC}</code> (u)
MI_prob3	0.2	bdd	∞	
MI_prob4	1.7	bdd	0.4	<code>sort.ρ_{optV}</code> (u)
MI_prob5	5.9	sort	0.3	<code>S4J.ρ_{optV}</code> (b)
MI_prob6	3.0	bdd	0.6	<code>add.ρ_{optV}</code> (b)
MI_prob7	8.1	bdd	2.4	<code>S4J.ρ_{optV}</code> (u)
MI_prob8	8.0	bdd	1.6	<code>sort.ρ_{optC}</code> (mr)

(b) RNP benchmarks

instance	No RNS		With RNS	
25-25-1	0.2	bdd	0.3	<code>sort.ρ_{optC}</code> (b)
25-25-2	0.4	bdd	6.6	<code>bdd.ρ_{optC}</code> (b)
25-25-3	0.1	bdd	0.2	<code>bdd.ρ_{optC}</code> (u)
25-25-4	0.2	bdd	0.8	<code>bdd.ρ_{optC}</code> (u)
25-25-5	0.1	bdd	0.7	<code>bdd.ρ_{optC}</code> (u)
25-25-6	0.2	bdd	0.5	<code>bdd.ρ_{optC}</code> (u)
25-25-7	0.1	bdd	6.5	<code>bdd.ρ_{optC}</code> (b)
25-25-8	0.4	bdd	11.4	<code>bdd.ρ_{optC}</code> (b)
25-25-9	0.4	bdd	0.2	<code>sort.ρ_{optC}</code> (b)
25-25-10	0.4	bdd	0.3	<code>sort.ρ_{optV}</code> (mr)
30-30-1	9.0	bdd	10.0	<code>bdd.ρ_{optC}</code> (u)
30-30-2	4.6	bdd	151.1	<code>bdd.ρ_{optV}</code> (b)
30-30-3	8.7	bdd	1588.8	<code>sort.ρ_{optC}</code> (mr)
30-30-4	418.6	sort	66.9	<code>bdd.ρ_{optC}</code> (b)
30-30-5	∞		44.9	<code>bdd.ρ_{optC}</code> (u)
30-30-6	7.8	bdd	95.0	<code>bdd.ρ_{optC}</code> (mr)
30-30-7	9.9	bdd	1069.8	<code>bdd.ρ_{optV}</code> (b)
30-30-8	7.6	bdd	345.4	<code>bdd.ρ_{optC}</code> (u)
30-30-9	∞		449.5	<code>bdd.ρ_{optC}</code> (mr)
30-30-10	9.8	bdd	1201.9	<code>sort.ρ_{optC}</code> (b)
35-35-1	∞		∞	
35-35-2	598.7	sort	539.7	<code>sort.ρ_{optV}</code> (b)
35-35-3	∞		868.7	<code>sort.ρ_{optC}</code> (b)
35-35-4	∞		590.2	<code>bdd.ρ_{optC}</code> (u)
35-35-5	333.2	sort	362.7	<code>sort.ρ_{optC}</code> (mr)
35-35-6	∞		∞	
35-35-7	44.6	sort	43.7	<code>sort.ρ_{optC}</code> (b)
35-35-8	∞		807.5	<code>bdd.ρ_{optC}</code> (b)
35-35-9	∞		422.9	<code>bdd.ρ_{optC}</code> (u)
35-35-10	∞		42.1	<code>bdd.ρ_{optC}</code> (u)

Table 5. Best solving times (seconds) with and without RNS (1800 sec. timeout).

instance, is key to the success of the approach. We show that when applying our approach we can solve instances that we cannot solve without. In particular, in cases where the BDD encoding is out of scope of current solvers due to the size of the encoding, we can apply and solve the constraints using BDDs after RNS decomposition. We compare our approach with previous work which applies RNS decomposition but encodes pseudo-Boolean modulo constraints directly to SAT. The comparison indicates the advantage of our approach where pseudo-Boolean modulo constraints are bit-blasted to standard pseudo-Boolean constraints. A repository of benchmark instances (before and after applying our transformation) as well as a more extensive presentation of the experimental results, are available at <http://www.cs.bgu.ac.il/~mcodish/Benchmarks/SAT2014>.

Acknowledgment: We thank the authors of [1] for making their PBM solver available, as well as their generator for RNP instances.

References

1. Aavani, A., Mitchell, D.G., Ternovska, E.: New encoding for translating pseudo-Boolean constraints into SAT. In: Frisch, A.M., Gregory, P. (eds.) SARA. AAAI (2013)
2. 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. (JAIR)* 45, 443–480 (2012)
3. Ansótegui, C., Manyà, F.: Mapping problems with finite-domain variables into problems with Boolean variables. In: SAT (2004)
4. Bailleux, O., Boufkhad, Y.: Efficient CNF encoding of Boolean cardinality constraints. In: CP2003. pp. 108–122 (2003)
5. Barth, P.: Logic-based 0-1 constraint programming. Kluwer Academic Publishers, Norwell, MA, USA (1996)
6. Berre, D.L., Parrain, A.: The Sat4j library, rel. 2.2. *JSAT* 7(2-3), 59–6 (2010)
7. Bixby, R.E., Boyd, E.A., Indovina, R.R.: MIPLIB: A test set of mixed integer programming problems. *SIAM News* 25, 16 (1992)
8. Borgs, C., Chayes, J.T., Pittel, B.: Phase transition and finite-size scaling for the integer partitioning problem. *Random Struct. Algorithms* 19(3-4), 247–288 (2001)
9. Bryant, R.E., Lahiri, S.K., Seshia, S.A.: Deciding CLU logic formulas via Boolean and pseudo-Boolean encodings. In: Proc. Intl. Workshop on Constraints in Formal Verification (CFV '02) (2002)
10. Bryant, R.E., Kroening, D., Ouaknine, J., Seshia, S.A., Strichman, O., Brady, B.: Deciding bit-vector arithmetic with abstraction. In: IN PROC. TACAS 2007. pp. 358–372. Springer (2007)
11. Codish, M., Fekete, Y., Fuhs, C., Schneider-Kamp, P.: Optimal base encodings for pseudo-Boolean constraints. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS. Lecture Notes in Computer Science, vol. 6605, pp. 189–204. Springer (2011)
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Third Edition. The MIT Press, 3rd edn. (2009)
13. Crawford, J.M., Baker, A.B.: Experimental results on the application of satisfiability algorithms to scheduling problems. In: Hayes-Roth, B., Korf, R.E. (eds.) AAAI. vol. 2, pp. 1092–1097. AAAI Press / The MIT Press, Seattle, WA, USA (1994)
14. Eén, N., Sörensson, N.: Translating pseudo-Boolean constraints into SAT. *JSAT* 2(1-4), 1–26 (2006)
15. Manquinho, V., Roussel, O.: Pseudo-Boolean competition 2012, <http://www.cril.univ-artois.fr/PB12/>
16. Manquinho, V.M., Roussel, O.: The first evaluation of Pseudo-Boolean solvers (PB'05). *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* 2(1-4), 103–143 (2006)
17. Metodi, A., Codish, M., Stuckey, P.J.: Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *J. Artif. Intell. Res. (JAIR)* 46, 303–341 (2013)
18. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear CSP into SAT. *Constraints* 14(2), 254–272 (2009)