Noname manuscript No.
(will be inserted by the editor)

Bound-Consistent Spread Constraint

Application to load balancing in nurse-to-patient assignments

Pierre Schaus · Jean-Charles Régin

Received: date / Accepted: date

Abstract Given a vector of finite domain variables, the spread constraint aims at minimizing the sum of squares of these variables while constraining the sum of these to be equal to a given value. We improve the existing filtering for spread achieving a bound-consistent filtering without increasing the complexity. Previous versions of the algorithm considered a relaxed version of the bound-consistency assuming interval domains defined on rational numbers rather than integers. We apply our new algorithm to a real life problem: the daily assignment of newborn infant patients to nurses in a hospital. The objective is to balance the workload of the nurses, while satisfying a variety of side constraints. Prior work proposed a MIP model for this problem, which unfortunately did not scale to large instances and only approximated the objective function, since minimizing the variance cannot be expressed in a linear model. This paper presents a two-step approach, first assigning nurses to region of the hospital then assigning the patients to these. We show that our approach allows to tackle large instances with hundreds of patients and nurses in a few seconds using the OscaR optimization system.

Keywords Global Constraint · Nurse Rostering · Fair Assignment

1 Introduction

Balancing constraints arise in many real-world applications, most often to express the need of a fair distribution of items or work. Simonis [16] suggested a global constraint to balance the shift distribution among nurses and Pesant [7] proposed the use of balancing constraints for a fair allocation of individual schedules.

Two global constraints and their propagators are available in constraint programming for optimizing load balancing: `spread` [8, 12], which constrains the vari-

P. Schaus
Université catholique de Louvain, ICTEAM (Belgium)
E-mail: pierre.schaus@uclouvain.be

J-C. Régin
Université de Nice-Sophia Antipolis (France)
E-mail: jcregin@gmail.com

ance and the mean of a set of variables, and `deviation` [13,11], which constrains the mean absolute deviation and the mean of a set of variables. We also say that `spread` and `deviation` respectively constrain the L_2 and L_1 norms of a set of variables $X_1..X_n$ with respect to their mean ($s = \sum_{i\in[1..n]} X_i$), *i.e.*,

- L_1: $\sum_{i\in[1..n]} |X_i - s/n|$;
- L_2: $\sum_{i\in[1..n]} (X_i - s/n)^2$.

These criteria are not equivalent: Minimizing L_1 or L_2 does not lead to the same solutions and it is not always obvious which one to choose. In fact, this is an old and recurrent debate (see for instance [1]). The L_2 criteria is more sensitive to outliers.

Contributions This article improves the filtering algorithm for the `spread` constraint. The initial filtering introduced in [8] is for a very general version of `spread` including a variable mean and a variable representing the median. Unfortunately this algorithm has never been implemented and is very difficult to understand[1]. A simplified version with a fixed constant mean was introduced in [12]. As in [8], this algorithm achieves a relaxed version of the bound-consistency assuming continuous interval domains. We introduce a stronger filtering achieving the classical bound-consistency on integer domains. We give all the details (proofs and pseudocode) of the intermediate results introduced in [8,12] and reused in our new algorithm. Our algorithm is explained visually and illustrated on numeric examples for each step; making is possible for someone to re-implement it. Our implementation is freely available in OscaR solver [6].

As an application for `spread` we revisit a problem of nurse-to-patient assignment first introduced in [4] and solved with CP in [14]. This problem considers a fair distribution of the workload assigned to nurses. We use the two-step decomposition approach from [4] computing first the number of nurses allocated to each zone before solving the problem in each zone independently. We prove that the first step can be solved optimally by solving a resource allocation problem. This proof was missing in [14]. Finally we show that for some instances, we can prove the two-step approach to be optimal using a new optimality check procedure for this problem.

Organization Section 2 formally defines the consistency levels achieved by the filtering algorithms from [8,12] and our new algorithm. Section 3 reviews the existing filtering algorithm and explains in detail the new filtering algorithms achieving bound-consistency for `spread`. Section 4 introduces the two-step decomposition approach to solve the nurse-to-patient assignment problem and experimental results using the new filtering for `spread`.

2 Definition

Preliminaries: We use the following definitions and notations to describe the semantics of the `spread` constraint and propagators.

[1] the ideas of the paper are correct. However, it contains some errors in the resolution of second degree equations which complexifies its implementation

Definition 1 Let X be a finite-domain (discrete) *variable*. The *domain* of X is a set of ordered values that can be assigned to X and is denoted by $Dom(X)$. The minimum (resp. maximum) value of the domain is denoted by $X^{\min} = \min(Dom(X))$ (resp. $X^{\max} = \max(Dom(X))$. An integer interval with integer bounds a and b is denoted $[a..b] \subseteq \mathbb{Z}$, while a rational interval is denoted $[a, b] \subseteq \mathbb{Q}$. An assignment on the variables $\mathbf{X} = [X_1, X_2, ..., X_n]$ is denoted by the tuple $\mathbf{x}$ and the i-th entry of this tuple by $\mathbf{x}[i]$. The extended rational interval domain of X_i is $I_D^{\mathbb{Q}}(X_i) = [X_i^{\min}, X_i^{\max}]$ and its integer interval domain is $I_D^{\mathbb{Z}}(X_i) = [X_i^{\min} \,..\, X_i^{\max}]$. Since we always consider n variables, when not specified otherwise our summations always range over $[1..n]$ *i.e.* $\sum_i$ is equivalent to $\sum_{i\in[1..n]}$. Similarly when it is possible we simplify $\forall i \in [1..n]$ into $\forall i$.

Before defining formally the spread constraint observe that if $\sum_i X_i = s$ then:

$$n \cdot \sum_i |X_i - s/n|^2 = n \cdot \sum_i X_i^2 - s^2. \tag{1}$$

It means that adding a constraint on L_2 criteria with a fixed sum value is equivalent to adding a constraint on $\sum_i X_i^2$ since s and n are constants. But working with the latter is more convenient when dealing with finite domain integer variables. We now define the `spread` constraint with a fixed mean.

Definition 2 Given finite domain variables $\mathbf{X} = (X_1, X_2, ..., X_n)$, an integer value s and a finite domain variable Δ, $\texttt{spread}(\mathbf{X}, s, \Delta)$ holds if and only if

$$\sum_i X_i = s \quad \text{and} \quad \sum_i X_i^2 \leq \Delta.$$

Example 1 Tuple $\mathbf{x} = (2, 3, 3, 2) \in \texttt{spread}([X_1, X_2, X_3, X_4], s = 10, \Delta = 26)$ but $\mathbf{x} = (1, 4, 3, 2) \notin \texttt{spread}([X_1, X_2, X_3, X_4], s = 10, \Delta = 26)$ because $1^2 + 4^2 + 3^2 + 2^2 = 30 > 26$.

The definition of bound-consistency specialized for continuous and integer interval domains is given next:

Definition 3 ($\mathbb{Q}$-bound-consistency and $\mathbb{Z}$-bound-consistency) A constraint $C(X_1, \ldots, X_n)$ $(n > 1)$ is $\mathbb{Q}$-bound-consistent (resp. $\mathbb{Z}$-bound-consistent) with respect to domains $Dom(X_i)$ if for all $i \in \{1, \ldots, n\}$ and each value $v_i \in \{X_i^{\min}, X_i^{\max}\}$, there exist values $v_j \in I_D^{\mathbb{Q}}(X_j)$ (resp. $v_j \in I_D^{\mathbb{Z}}(X_j)$) for all $j \in \{1, \ldots, n\} - \{i\}$ such that $(v_1, \ldots, v_n) \in C$.

Note that for the L_1 criterion, the $\mathbb{Q}$-bound-consistency filtering was introduced in [13] and the $\mathbb{Z}$-bound-consistency filtering in [11]. The $\mathbb{Q}$-bound-consistency filtering for `spread` is described in [8,12]. Our contribution is a stronger $\mathbb{Z}$-bound-consistency filtering for integer domains obtained by adapting the algorithm from [12].

We illustrate in the next example that the $\mathbb{Z}$-bound-consistency for `spread` can be stronger than the $\mathbb{Q}$-bound-consistency:

Example 2 Consider 10 variables with domains all equal to [1..2] and a sum that must be equal to 15. An assignment (from the extended continuous domain) minimizing the sum of squares would assign all of them to 1.5. The lower bound on

the sum of square variables will thus be set to $10 \cdot 1.5^2 = 22.5$. An integer assignment minimizing the sum of squares would assign five variables to 2 and the other five to 1. The lower bound on the sum of square variables would thus be set to $5 \cdot 2^2 + 5 \cdot 1^2 = 25$.

To achieve $\mathbb{Z}$-bound-consistency, the propagators for `spread` compute $\underline{\Delta}^{\mathbb{Z}}$ to filter $\Delta^{\min}$, and $\overline{X}_i^{\mathbb{Z}}$ and $\underline{X}_i^{\mathbb{Z}}$ to filter $X_i^{\max}$ and $X_i^{\min}$:

$$\underline{\Delta}^{\mathbb{Z}} = \min_{\mathbf{x}}\{\sum_i \mathbf{x}[i]^2 \text{ s.t. } \sum_i \mathbf{x}[i] = s \text{ and } \forall i : \mathbf{x}[i] \in I_D^{\mathbb{Z}}(X_i)\} \tag{2}$$

$$\overline{X}_i^{\mathbb{Z}} = \max_{\mathbf{x}}\{\mathbf{x}[i] \text{ s.t. } \sum_j \mathbf{x}[j]^2 \leq \Delta^{\max} \text{ and } \sum_j \mathbf{x}[j] = s \text{ and } \forall j : \mathbf{x}[j] \in I_D^{\mathbb{Z}}(X_j)\}. \tag{3}$$

Note that $\underline{X}_i^{\mathbb{Z}}$ is defined similarly. The filtering of Δ is implemented in time complexity $\mathcal{O}(n \cdot \log(n))$ and that of $\mathbf{X}$ in $\mathcal{O}(n^2)$. The next section explains how to compute these quantities efficiently to filter Δ and X_i's.

3 Filtering for Spread

First we explain in detail the $\mathcal{O}(n \cdot \log(n))$ algorithm from [8] to compute $\underline{\Delta}^{\mathbb{Q}}$. Then we adapt this algorithm to compute $\underline{\Delta}^{\mathbb{Z}}$. The key point of the algorithm given in the next theorem (in the $\mathbb{Q}$ context) is a clear characterization of an optimal solution to the problem of minimization of the sum of squares with a given sum. This theorem says that in an assignment of sum s minimizing the sum of squares, there cannot exist two values that can be made closer by moving both of them inside their corresponding domains.

Theorem 1 $\mathbf{x} \in \text{argmin}_{\mathbf{y}}\{\sum_i \mathbf{y}[i]^2 \text{ s.t. } \sum_i \mathbf{y}[i] = s \text{ and } \forall i : \mathbf{y}[i] \in I_D^{\mathbb{Q}}(X_i)\} \Leftrightarrow \sum_i \mathbf{x}[i] = s$ *and* $\nexists\ (i,j)$ *such that:* $i \neq j$, $\mathbf{x}[i] < X_i^{\max}$, $\mathbf{x}[j] > X_j^{\min}$ *and* $\mathbf{x}[i] < \mathbf{x}[j]$.

Proof We denote by $opt(\mathbf{x})$ the left member of the bi-conditional symbol and by $\neg p(\mathbf{x})$ the right member.

($\Rightarrow$) $p(\mathbf{x}) \Rightarrow \neg opt(\mathbf{x})$: Assume it is possible to find a pair i, j such that $\mathbf{x}[i] < X_i^{\max}$, $\mathbf{x}[j] > X_j^{\min}$ and $\mathbf{x}[i] < \mathbf{x}[j]$, then it is possible to transform the assignment $\mathbf{x}$ into a assignment $\mathbf{x}'$ also of sum s with $\sum_i \mathbf{x}'[i]^2 < \sum_i \mathbf{x}[i]^2$. We define the positive value $\delta = \min\{((\mathbf{x}[j]-\mathbf{x}[i])/2), (X_i^{\max}-\mathbf{x}[i]), (\mathbf{x}[j]-X_j^{\min})\}$ and $\mathbf{x}'[k] = \mathbf{x}[k]\ \forall k \neq i, j$, $\mathbf{x}'[j] = \mathbf{x}[j] - \delta$, $\mathbf{x}'[i] = \mathbf{x}[i] + \delta$. If we define $\Delta(\mathbf{x}) = \sum_i \mathbf{x}[i]^2$, then $\Delta(\mathbf{x}) - \Delta(\mathbf{x}') = -2\delta^2 + 2\delta \cdot (\mathbf{x}[j] - \mathbf{x}[i]) \geq \delta^2 > 0$ because $(\mathbf{x}[j] - \mathbf{x}[i]) \geq 2\delta$.
($\Leftarrow$) $\neg opt(\mathbf{x}) \Rightarrow p(\mathbf{x})$: We assume that $\mathbf{x}$ is not optimal and we consider $\mathbf{x}'$ a modified version of $\mathbf{x}$ also of sum s. Without loss of generality, we consider that the first k entries were increased, the next l were decreased and the other were left unchanged: $(\mathbf{x}'[i] - \mathbf{x}[i]) = \delta_i > 0$ for $i \in [1..k]$, $(\mathbf{x}[i] - \mathbf{x}'[i]) = \delta_i > 0$ for

$i \in [k+1..k+l]$ and $\mathbf{x}[i] = \mathbf{x}'[i]$ for $i \in [(k+l+1)..n]$. The sum s is preserved from $\mathbf{x}$ to $\mathbf{x}'$ if $\sum_{i=1}^{k} \delta_i - \sum_{i=k+1}^{k+l} \delta_i = 0$. The difference between the sum of squares can be written as:

$$\Delta(\mathbf{x}) - \Delta(\mathbf{x}') = -\sum_{i=1}^{k+l} \delta_i^2 - 2 \cdot \sum_{i=1}^{k} \mathbf{x}[i]\delta_i + 2 \cdot \sum_{i=k+1}^{k+l} \mathbf{x}[i]\delta_i.$$

We show that this difference is strictly negative (*i.e.* $\mathbf{x}'$ does not improve $\mathbf{x}$) if every increasing entry is larger than every decreasing entry, $\forall i \in [1..k], \forall j \in [k+1..k+l] : \mathbf{x}[i] > \mathbf{x}[j]$. A lower bound on $(\sum_{i=1}^{k} \mathbf{x}[i]\delta_i)$ is $((\min_{i\in[1..k]} \mathbf{x}[i]) \cdot \sum_{i=1}^{k} \delta_i)$ and a strict upper bound on $(\sum_{i=k+1}^{k+l} \mathbf{x}[i]\delta_i)$ is $((\min_{i\in[1..k]} \mathbf{x}[i]) \cdot \sum_{i=k+1}^{k+l} \delta_i)$. Hence we have:

$$\Delta(\mathbf{x}) - \Delta(\mathbf{x}') < (-\sum_{i=1}^{k+l} \delta_i^2) + 2 \cdot (\min_{i\in[1..k]} \mathbf{x}[i]) \cdot (-\sum_{i=1}^{k} \delta_i + \sum_{i=k+1}^{k+l} \delta_i) = -\sum_{i=1}^{k+l} \delta_i^2 < 0$$

because $\sum_{i=1}^{k} \delta_i - \sum_{i=k+1}^{k+l} \delta_i = 0$. □

An assignment $\mathbf{x}$ such that $\nexists\, i, j$ with $\mathbf{x}[i] < X_i^{\max}$, $\mathbf{x}[j] > X_j^{\min}$ and $\mathbf{x}[i] < \mathbf{x}[j]$ is called a ν-$\mathbb{Q}$centered assignment in [8]:

Definition 4 (ν-$\mathbb{Q}$centered assignment [8]) A ν-$\mathbb{Q}$centered assignment $\mathbf{x} \in \mathbb{Q}^n$ on $\mathbf{X}$ with $\nu \in \mathbb{Q}$ is such that $\forall i$:

$$\mathbf{x}[i] = \begin{cases} X_i^{\max} & \text{if } X_i^{\max} < \nu \\ X_i^{\min} & \text{if } X_i^{\min} > \nu \\ \nu & \text{otherwise.} \end{cases}$$

Corollary 1 ([8]) $\mathbf{x} \in \operatorname{argmin}_{\mathbf{y}} \{\sum_i \mathbf{y}[i]^2 \text{ s.t. } \sum_i \mathbf{y}[i] = s \text{ and } \forall i : \mathbf{y}[i] \in I_D^{\mathbb{Q}}(X_i)\}$ *if and only if* $\mathbf{x}$ *is a* ν-$\mathbb{Q}$*centered assignment of sum* s*.*

The optimal value $\underline{\Delta}^{\mathbb{Q}}$ can be obtained from a tuple which is ν-$\mathbb{Q}$centered and exhibiting a sum of s. Note that by definition of a ν-$\mathbb{Q}$centered assignment, this tuple is completely defined by the value ν. The algorithm from [8] searches for the value ν such that the ν-$\mathbb{Q}$centered assignment exhibits a sum of s.

For a given value ν, the sum of the corresponding ν-$\mathbb{Q}$centered assignment is

$$\sum_{X_i^{\min} > \nu} X_i^{\min} + \sum_{X_i^{\max} < \nu} X_i^{\max} + \sum_{\nu \in I_D^{\mathbb{Q}}(X_i)} \nu. \tag{4}$$

In order to simplify the notations, we denote by:

- $R(\nu)$, the (Right) variables $\{X_i \text{ s.t. } X_i^{\min} > \nu\}$,
- $L(\nu)$, the (Left) variables $\{X_i \text{ s.t. } X_i^{\max} < \nu\}$ and by
- $M(\nu)$, the (Medium) variables $\{X_i \text{ s.t. } X_i \notin (R(\nu) \cup L(\nu))\}$.

The cardinality of the sets $R(\nu)$, $L(\nu)$ and $M(\nu)$ are denoted respectively $r(\nu)$, $l(\nu)$ and $m(\nu)$. Let us furthermore define

- $es(\nu) = \sum_{X_i \in R(\nu)} X_i^{\min} + \sum_{X_i \in L(\nu)} X_i^{\max}$ (Extrema-Sum), and
- $es_{(2)}(\nu) = \sum_{X_i \in R(\nu)} (X_i^{\min})^2 + \sum_{X_i \in L(\nu)} (X_i^{\max})^2$.

When no ambiguity is possible, the ν argument will be dropped.

With the introduced notations, the sum of the ν-$\mathbb{Q}$centered assignment (4) can be simply written as $es + m \cdot \nu$. Our objective to find the lower bound $\underline{\Delta}^{\mathbb{Q}}$ is achieved if we can find a value ν such that $es + m \cdot \nu = s$.

Nonempty intervals such that every domain either completely subsumes it or does not overlap it, have the property that for every value ν inside it, the sets $R(\nu)$, $L(\nu)$, $M(\nu)$ and the value $es(\nu)$ are constant. This is more formally stated in Property 1.

Property 1 All intervals $I = [a, b] \subset \mathbb{Q}$ with $a < b$ such that $\forall i : (I \subseteq I_D^{\mathbb{Q}}(X_i)) \vee (I \geq I_D^{\mathbb{Q}}(X_i)) \vee (I \leq I_D^{\mathbb{Q}}(X_i))$ have the property that for every $a < (\nu^1, \nu^2) < b$ the following equalities hold: $R(\nu^1) = R(\nu^2)$, $L(\nu^1) = L(\nu^2)$, $M(\nu^1) = M(\nu^2)$ and $es(\nu^1) = es(\nu^2)$.

Proof Direct consequences of the definitions of R, L, M and es.

Property 1 leads naturally to an extension of the definitions of R, L, M and es to intervals I such as the ones considered in Property 1:

- $R(I)$, the variables $\{X_i$ s.t. $X_i^{\min} \geq \max(I)\}$,
- $L(I)$, the variables $\{X_i$ s.t. $X_i^{\max} \leq \min(I)\}$,
- $M(I)$, the variables $\{X_i$ s.t. $X_i^{\min} \leq \min(I)$ and $X_i^{\max} \geq \max(I)\}$ and
- $es(I) = \sum_{X_i \in L(I)} X_i^{\max} + \sum_{X_i \in R(I)} X_i^{\min}$.

Note that $M(I) = \mathbf{X} - (R(I) \cup L(I))$.

When ν varies inside such an interval I, the sum ranges in the interval $si(I) = [es(I) + m(I) \cdot \min(I), es(I) + m(I) \cdot \max(I)]$ (si stands for Sum Interval). If s does not fall in $si(I)$ then the value ν we are looking for does not lie inside I neither. On the contrary if s belongs to this interval, ν is the solution of the equation $s = es + m \cdot \nu$ that is $\nu = (s - es)/m$.

There are at most $2 \cdot n - 1$ intervals I to consider (when all the bounds are different). These are obtained by sorting the set of upper and lower bounds of every variable X_i into increasing order. Any two consecutive values of this sorted sequence form an interval I satisfying Property 1.

Definition 5 Let $B(\mathbf{X})$ be the sorted sequence in non-decreasing order of the set of bounds[2] $\bigcup_i \{X_i^{\min}, X_i^{\max}\}$. Let $\mathcal{I}(\mathbf{X})$ be the set of intervals defined by a pair of two consecutive elements of $B(\mathbf{X})$. The k^{th} interval of $\mathcal{I}(\mathbf{X})$ is denoted by I_k. For an interval $I = I_k$ we define the operator $\text{prev}(I) = I_{k-1}, (k > 1)$ and $\text{succ}(I) = I_{k+1}, (k < |\mathcal{I}(\mathbf{X})|)$.

Example 3 (Building $I(\boldsymbol{X})$) Let $\mathbf{X} = \{X_1, X_2, X_3\}$ with $Dom(X_1) = [1..3]$, $Dom(X_2) = [2..6]$ and $Dom(X_3) = [3..9]$. Then $\mathcal{I}(X) = \{I_1, I_2, I_3, I_4\}$ with $I_1 = [1, 2]$, $I_2 = [2, 3]$, $I_3 = [3, 6]$, $I_4 = [6, 9]$. We have $\text{prev}(I_3) = I_2$ and $\text{succ}(I_3) = I_4$.

Observe that for two consecutive intervals I_k and I_{k+1} taken from $\mathcal{I}(\mathbf{X})$, the sum intervals are also contiguous: $\max(si(I_k)) = \min(si(I_{k+1}))$. It is then possible to make an algorithm to filter the lower bound of Δ. Algorithm 1 computes $\underline{\Delta}^{\mathbb{Q}}$ by iterating over the contiguous intervals $si(I_k)$ until the sum lies inside it.

[2] without duplicates.

Algorithm 1: Filtering of Δ

Data: $\mathcal{I}(\mathbf{X})$ and for all $I \in \mathcal{I}(\mathbf{X})$: $m(I)$ and $es(I)$.
Result: $\Delta^{\min} \leftarrow \max\{\underline{\Delta}^{\mathbb{Q}}, \Delta^{\min}\}$.
1 **forall the** $I \in \mathcal{I}(\boldsymbol{X})$ **do**
2 **if** $s \in si(I)$ **then**
3 $\nu \leftarrow (s - es(I))/m(I)$;
4 $\underline{\Delta}^{\mathbb{Q}} \leftarrow \left(es_{(2)}(I) + m(I) \cdot \nu^2\right)$;
5 $\Delta^{\min} \leftarrow \max\{\underline{\Delta}^{\mathbb{Q}}, \Delta^{\min}\}$;
6 break ;

Algorithm 1 executes in $\mathcal{O}(n)$ once $\mathcal{I}(\mathbf{X})$ is computed and that $es(I)$ and $m(I)$ are available for every $I \in \mathcal{I}(\mathbf{X})$. Unfortunately, computing $\mathcal{I}(\mathbf{X})$ requires to sort the bounds, hence $\mathcal{O}(n \log(n))$ time. Algorithm 1 also needs the values $es(I)$ and $m(I)$. For a given I, these can be obtained in $\Theta(n)$ by scanning every variable once. This would raise the overall complexity of Algorithm 1 to $\mathcal{O}(n^2)$ since in the worst case, $es(I)$ and $m(I)$ must be computed for every $I \in \mathcal{I}(\mathbf{X})$. A smarter procedure is possible to compute $es(I)$ and $m(I)$ in linear time for every $I \in \mathcal{I}(\mathbf{X})$. Lemma 1 explains how the values $es(I)$ can be computed in linear time for all $I \in \mathcal{I}(\mathbf{X})$ once $l(I), r(I)$ are given for all $I \in \mathcal{I}(\mathbf{X})$. Finally Algorithm 2 computes $l(I)$, $r(I)$ and $m(I)$ for all $I \in \mathcal{I}(\mathbf{X})$ in linear time too. This last algorithm can be easily understood with the invariant given in the pseudocode.

Lemma 1 ([8]) *$es(I_{k+1}) = es(I_k) + (p_{k+1} - q_{k+1}) \cdot \max(I_k)$ where $p_{k+1} = l(I_{k+1}) - l(I_k)$ and $q_{k+1} = r(I_k) - r(I_{k+1})$.*

Note that lc is initialized at line 1 with b_1^+ and not 0 because some variables might already be bound to value b_1.

Algorithm 2: Compute $\mathcal{I}(\mathbf{X})$ and $l(I), r(I), m(I)$ for all $I \in \mathcal{I}(\mathbf{X})$

Data: The sorted sequence of the set of bounds $B(\mathbf{X}) = \langle b_1, b_2, ..., b_k \rangle$ and for each bound b_i the information $b_i^+ = |\{X_i | X_i^{\max} = b_i\}|$ and $b_i^- = |\{X_i | X_i^{\min} = b_i\}|$.
Result: $\mathcal{I}(\mathbf{X})$ and for all $I \in \mathcal{I}(\mathbf{X})$: $m(I)$, $l(I)$ and $r(I)$.
1 $lc \leftarrow b_1^+$
2 $rc \leftarrow n - b_1^-$
3 $\mathcal{I} \leftarrow list()$
4 **for** $i \leftarrow 2$ **to** k **do**
  `/* invariant:` $lc = |\{X_j | X_j^{\max} \leq b_{i-1}\}|$ and $rc = |\{X_j | X_j^{\min} \geq b_i\}|$ `*/`
5 $I \leftarrow [b_{i-1}, b_i]$
6 $\mathcal{I}.add(I)$
7 $l(I) = lc$
8 $r(I) = rc$
9 $m(I) = n - l(I) - r(I)$
10 $lc \leftarrow lc + b_i^+$
11 $rc \leftarrow rc - b_i^-$

The complete filtering algorithm dominated by a complexity of $\mathcal{O}(n \log(n))$ is:

- sort the bounds ($\mathcal{O}(n \log(n))$),

i	I_i	$R(I_i)$	$L(I_i)$	$M(I_i)$	$es(I_i)$	$es_{(2)}(I_i)$	$si(I_i)$
1	$[1,2]$	x_2, x_3	ϕ	x_1	5	13	$[6,7]$
2	$[2,3]$	x_3	ϕ	x_1, x_2	3	9	$[7,9]$
3	$[3,6]$	ϕ	x_1	x_2, x_3	3	9	$[9,15]$
4	$[6,9]$	ϕ	x_1, x_2	x_3	9	45	$[15,18]$

Table 1 Values I_i, $R(I_i)$, $L(I_i)$, $M(I_i)$, $es(I_i)$, $es_{(2)}(I_i)$, $si(I_i)$ relative to Example 4.

- compute $\mathcal{I}(\mathbf{X})$ and for all $I \in \mathcal{I}(\mathbf{X})$: $r(I)$, $l(I)$ and $m(I)$ with Algorithm 2 ($\Theta(n)$),
- compute $es(I)$ for all $I \in \mathcal{I}(\mathbf{X})$ with Lemma 1 ($\mathcal{O}(n)$),
- filter Δ with Algorithm 1 ($\mathcal{O}(n)$).

Example 4 (Computing $\underline{\Delta}^{\mathbb{Q}}$) Variables and domains are from Example 3 and the sum considered is $s = 10$. Relevant values necessary to compute $\underline{\Delta}^{\mathbb{Q}}$ are given in Table 1. Since $s \in si(I_3)$: $\nu = (s - es)/m = (10 - 3)/2 = 3.5$ and $\underline{\Delta}^{\mathbb{Q}} = es_{(2)}(I_3) + m(I_3) \cdot 3.5^2 = 33.5$. For $s = 9$, $s \in si(I_2)$ and $s \in si(I_3)$. Whichever interval is chosen between I_2 and I_3, the value ν is the same. Consequently the value for $\underline{\Delta}^{\mathbb{Q}}$ is also the same.

The remainder of the section explains how to compute $\underline{\Delta}^{\mathbb{Z}} \geq \lceil \underline{\Delta}^{\mathbb{Q}} \rceil$ that will allow us to achieve $\mathbb{Z}$-bound-consistency rather than $\mathbb{Q}$-bound-consistency on `spread`.

As shown in the next theorem, the optimal solution obtained with integer assignments is very similar to the one obtained on rational domains in Theorem 1.

Theorem 2 $\mathbf{x} \in \text{argmin}_{\mathbf{y}}\{\sum_i \mathbf{y}[i]^2$ s.t. $\sum_i \mathbf{y}[i] = s$ and $\forall i : \mathbf{y}[i] \in I_D^{\mathbb{Z}}(X_i)\}$ $\Leftrightarrow \sum_i \mathbf{x}[i] = s$ *and* $\nexists\ (i,j)$ *such that:* $i \neq j$, $\mathbf{x}[i] < X_i^{\max}$, $\mathbf{x}[j] > X_j^{\min}$ *and* $\mathbf{x}[i] + 1 < \mathbf{x}[j]$.

Proof Similar to the proof of Theorem 1:

($\Rightarrow$) Take $\delta = 1$.
($\Leftarrow$) All the δ_i's are integer and greater or equal to 1. We show that the difference $(\Delta(\mathbf{x}) - \Delta(\mathbf{x}'))$ is nonpositive if every increasing entry is at most one smaller than every decreasing entry, $\forall i \in [1..k], \forall j \in [k+1..k+l] : \mathbf{x}[i] + 1 \geq \mathbf{x}[j]$. A lower bound on $(\sum_{i=1}^{k} \mathbf{x}[i]\delta_i)$ is $((\min_{i\in[1..k]} \mathbf{x}[i]) \cdot \sum_{i=1}^{k} \delta_i)$ and an upper bound on $(\sum_{i=k+1}^{k+l} \mathbf{x}[i]\delta_i)$ is $((\min_{i\in[1..k]} \mathbf{x}[i] + 1) \cdot \sum_{i=k+1}^{k+l} \delta_i)$. Hence we have:

$$\begin{aligned}
\Delta(\mathbf{x}) - \Delta(\mathbf{x}') &\leq (-\sum_{i=1}^{k+l} \delta_i^2) + (\min_{i\in[1..k]} \mathbf{x}[i]) \cdot (-2\sum_{i=1}^{k} \delta_i + 2\sum_{i=k+1}^{k+l} \delta_i) + 2\sum_{i=k+1}^{k+l} \delta_i \\
&\leq (-\sum_{i=1}^{k+l} \delta_i^2) + 2\sum_{i=k+1}^{k+l} \delta_i \\
&\leq (-\sum_{i=1}^{k} \delta_i^2) + \sum_{i=k+1}^{k+l} \delta_i(2 - \delta_i).
\end{aligned}$$

Since $\forall i \in [1..k+l] : \delta_i \geq 1$ we have $\sum_{i=1}^{k} \delta_i^2 \geq \sum_{i=1}^{k} \delta_i$ and $\sum_{i=k+1}^{k+l} \delta_i(2-\delta_i) \leq \sum_{i=k+1}^{k+l} \delta_i$. Thus

$$-\sum_{i=1}^{k} \delta_i^2 + \sum_{i=k+1}^{k+l} \delta_i(2-\delta_i) \leq -\sum_{i=1}^{k} \delta_i + \sum_{i=k+1}^{k+l} \delta_i = 0. \quad \square$$

Definition 6 A $\mathbb{Z}$centered assignment $\mathbf{x}$ is such that $\forall i : \mathbf{x}[i] \in I_D^{\mathbb{Z}}(X_i)$ and $\nexists\, i, j$ such that $\mathbf{x}[i] < X_i^{\max}$, $\mathbf{x}[j] > X_j^{\min}$ and $\mathbf{x}[i] + 1 < \mathbf{x}[j]$.

Algorithm 3 computes a $\mathbb{Z}$centered assignment of sum s. This algorithm can

Algorithm 3: An inefficient algorithm to compute a $\mathbb{Z}$centered assignment of sum s.

Result: A $\mathbb{Z}$centered assignment $\mathbf{x}$ of sum s.
1 $\mathbf{x} \leftarrow$ a valid assignment of sum s
2 **while** $\exists\, i, j$ *such that* $\mathbf{x}[i] < X_i^{\max}$, $\mathbf{x}[j] > X_j^{\min}$ *and* $\mathbf{x}[i] + 1 < \mathbf{x}[j]$ **do**
3 $\quad \mathbf{x}[i] \leftarrow \mathbf{x}[i] + 1$
4 $\quad \mathbf{x}[j] \leftarrow \mathbf{x}[j] - 1$

be easily implemented but is very inefficient. A smarter method is possible. The idea is that it is always possible to distribute the m entries assigned to ν on the integer values $\lfloor \nu \rfloor$ and $\lceil \nu \rceil$ while conserving a sum of s. Clearly, such an assignment is $\mathbb{Z}$centered since the values assigned to ν differ by at most one after the transformation.

The distribution of the m entries on $\lfloor \nu \rfloor$ and $\lceil \nu \rceil$ while conserving the sum of s is given as follows. The value ν always takes the form of $(s - es)/m$ and there are exactly m entries assigned to ν. Hence it is always possible to distribute these m entries on $\lfloor \nu \rfloor$ and $\lceil \nu \rceil$ without modifying the sum. The only question to answer is how many of them must be assigned to $\lfloor \nu \rfloor$ and to $\lceil \nu \rceil$? Let us denote by ν^+ the number of entries that must be assigned to $\lceil \nu \rceil$ and by ν^- the number of entries that must be assigned to $\lfloor \nu \rfloor$. Of course $\nu^+ + \nu^- = m$. Looking at the Figure 1, the sum is preserved if

$$\nu^- \cdot ((s - es) \bmod m) = (m - \nu^-) \cdot (m - (s - es) \bmod m).$$

Hence the distribution on $\lfloor \nu \rfloor$ and $\lceil \nu \rceil$ is:

$\nu^- = m - (s - es) \bmod m$
$\nu^+ = (s - es) \bmod m.$

Algorithm 1 can be modified into Algorithm 4 to achieve a bound-consistent filtering of Δ with the computation of $\underline{\Delta}^{\mathbb{Z}}$.

Example 5 The variables are the same as in Example 4 and the sum is $s = 10$. From Example 4, $s \in si(I_3)$ and $\nu = (s - es)/m = (10 - 3)/2 = 3.5$. The distribution of overlapping variables between $\lfloor \nu \rfloor$ and $\lceil \nu \rceil$ is given by $\nu^+ = (s - es(I)) \bmod m(I) = (10 - 3) \bmod 3 = 1$ and $\nu^- = 1$. Consequently $\underline{\Delta}^{\mathbb{Z}} = es_{(2)}(I_3) + 4^2 + 3^2 = 34$.

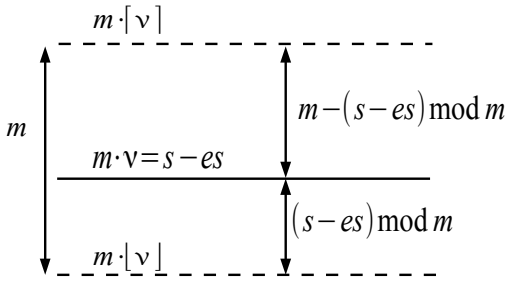

Fig. 1 Distances between $\nu = (s - es)/m$, $\lfloor \nu \rfloor$ and $\lceil \nu \rceil$.

Algorithm 4: Bound-consistent filtering of Δ

Data: $\mathcal{I}(\mathbf{X})$ and for all $I \in \mathcal{I}(\mathbf{X})$: $m(I)$ and $es(I)$.
Result: $\Delta^{\min} \leftarrow \max\{\underline{\Delta}^{\mathbb{Z}}, \Delta^{\min}\}$.
forall the $I \in \mathcal{I}(\boldsymbol{X})$ **do**
  if $s \in si(I)$ **then**
    $\nu \leftarrow (s - es(I))/m(I)$
    $\nu^{+} \leftarrow (s - es(I)) \bmod m(I)$
    $\nu^{-} \leftarrow m - (s - es(I)) \bmod m(I)$
    $\underline{\Delta}^{\mathbb{Z}} \leftarrow es_{(2)}(I) + \nu^{+} \cdot \lceil \nu \rceil^2 + \nu^{-} \cdot \lfloor \nu \rfloor^2$
    $\Delta^{\min} \leftarrow \max\{\underline{\Delta}^{\mathbb{Z}}, \Delta^{\min}\}$
    break

3.1 Filtering of **X**

Only the propagation of the upper bound of X_i is considered here since the propagation of its lower bound is a symmetrical problem. The propagation of X_i is achieved in [12] by computing the largest consistent value assuming the domains of other variables are rational intervals:

$$\overline{X_i}^{\mathbb{Q}} = \max_{\mathbf{x}}\{\mathbf{x}[i] \text{ s.t. } \sum_i \mathbf{x}[i]^2 \leq \Delta^{\max} \text{ and } \sum_i \mathbf{x}[i] = s \text{ and } \forall j : \mathbf{x}[j] \in I_D^{\mathbb{Q}}(X_j)\}. \quad (5)$$

3.1.1 Algorithm to compute $\overline{X_i}^{\mathbb{Q}}$

The idea of the algorithm from [8] is to start from the a ν-$\mathbb{Q}$centered assignment $\mathbf{x}$ of sum s found with Algorithm 1. This assignment is the one of sum s with minimal sum of squares $\underline{\Delta}^{\mathbb{Q}}$. If $\mathbf{x}[i] = X_i^{\max}$, the upper bound of the variable X_i is consistent. Otherwise the optimal value $\overline{X_i}^{\mathbb{Q}}$ is obtained by successively assigning X_i to increasing values until either the minimal sum of squares reaches $\Delta^{\max}$ or it is proved that $X_i^{\max}$ is consistent. At that point, $\overline{X_i}^{\mathbb{Q}}$ is equal to the current value considered for X_i. This procedure is valid since the increasing values assigned to X_i range from $\mathbf{x}[i]$ to $X_i^{\max}$ and the minimal sum of squares $\underline{\Delta}^{\mathbb{Q}}$ increases quadratically when the value assigned to X_i increases.

A detailed description of Algorithm 5 follows. Let us denote by x_i the current value assigned to X_i and by $d = x_i - \mathbf{x}[i]$ the distance of the current value of X_i to the ith entry in the starting ν-$\mathbb{Q}$centered assignment $\mathbf{x}$ of sum s. Let us furthermore denote by $\underline{\Delta}^{\mathbb{Q}'}, si', es', m', \nu'$ the modified values if X_i were assigned to $x_i = \mathbf{x}[i] + d$.

The interval I from $\mathcal{I}(\mathbf{X})$ is such that $s \in si(I)$. Let us first assume $X_i \in R(I)$, which implies $\mathbf{x}[i] = X_i^{\min}$ (by definition of a ν-$\mathbb{Q}$centered assignment). Recall that if $X_i \in M(I)$ then $\mathbf{x}[i] = \nu = (s - es(I))/m$ and if $X_i \in L(I)$ then $\mathbf{x}[i] = X_i^{\max}$ and no filtering of the upper bound is possible.

The following lemma describes the quadratic evolution of $\underline{\Delta}^{\mathbb{Q}}$ with d, *i.e.* when x_i increases. An illustration of Lemma 2 is given in Figure 2.

Lemma 2 *if* $d \leq s - \min(si(I))$ *then*

1. $es'(I) = es(I) + d$,
2. $es'_{(2)}(I) = es_{(2)} + d^2 + 2d \cdot X_i^{\min}$,
3. $m' = m$,
4. $\nu' = \nu - d/m$ *and*
5. $\underline{\Delta}^{\mathbb{Q}'} = es'_{(2)}(I) + m \cdot (\nu')^2$
   $\underline{\Delta}^{\mathbb{Q}'} = es_{(2)}(I) + d^2 + 2dX_i^{\min} + m \cdot (\nu - \frac{d}{m})^2$
   $\underline{\Delta}^{\mathbb{Q}'} = \underline{\Delta}^{\mathbb{Q}} + (d^2 + 2dX_i^{\min} + d^2/m - 2d\nu)$.

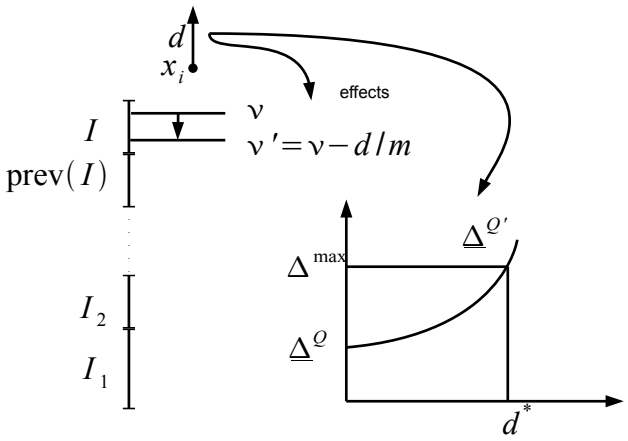

Fig. 2 Consequences of the increasing of x_i by d : $\underline{\Delta}^{\mathbb{Q}'}$ increases quadratically and ν' decreases by d/m.

From Lemma 2, the minimal sum of squares increases quadratically with d. The maximum consistent value for X_i is the nonnegative solution of the following second degree equation:

$$\underline{\Delta}^{\mathbb{Q}} + d^2 + 2dX_i^{\min} + d^2/m - 2d\nu = \Delta^{\max}.$$

The nonnegative solution of this equation is $d^* = \frac{-b+\sqrt{b^2-ac}}{a}$ where $a = 1 + 1/m$, $b = X_i^{\min} - \nu$ and $c = \underline{\Delta}^{\mathbb{Q}} - \Delta^{\max}$. This reasoning is valid as long as $d \leq s - \min(si(I))$. Otherwise, ν' moves outside the interval I and the modified values in Lemma 2 are no longer valid. If $d^* \leq s - \min(si(I))$ then $\overline{X}_i^{\mathbb{Q}} = \mathbf{x}[i] + d^*$. Otherwise, $\mathbf{x}[i] + s - \min(si(I))$ is a consistent value for X_i and larger values for x_i must be considered. The procedure is repeated with the interval $\text{prev}(I)$ instead of I. Indeed when $X_i \leftarrow x_i = \mathbf{x}[i] + s - \min(si(I))$, the value ν' is equal to $\min(I) = \max(\text{prev}(I))$ and $s = \min(si'(I)) = \max(si'(\text{prev}(I)))$.

The process is repeated until one valid solution of the second degree equation $d^* \leq s - \min(si(I))$ is found or the current interval considered is equal to I_1.

Until now we assumed that $X_i \in R(I)$. The case $X_i \in M(I)$ can also lead to the filtering of $X_i^{\max}$. This case can be reduced to the same procedure as for the case $X_i \in R(I)$:

- The x_i starts to increase from ν rather than from $X_i^{\min}$ as previously. Hence, the interval I is conceptually split in two at the value ν: $I' = [\min(I), \nu]$ and $[\nu, \max(I)]$.
- The value $\overline{X}_i^{\mathbb{Q}}$ is computed on this modified configuration by considering that the domain of X_i is now $[\nu, X_i^{\max}]$.
- The same procedure as described previously can be applied since with this modified domain we have that $X_i \in R(I')$.

We say that the interval I is conceptually split because actually we only need to adapt the computation of $m'(I)$, $es'(I)$ and $es'_{(2)}$ from Lemma 2. For $x_i = X_i^{\min} + d$ and an interval $I \in \mathcal{I}(\mathbf{X})$ such that $x_i > \min(I)$, if $X_i \in M(I)$ then $m'(I') = m(I) - 1$ where $I' = [\min(I), \min\{\max(I), x_i\}]$. The reason is that $I_D^{\mathbb{Q}}(X_i)$ subsumes I' hence if $X_i \leftarrow x_i$, then X_i belongs to $R(I')$ and not to $M(I')$ anymore. The unified procedure to adapt the values m', es' and $es'_{(2)}$ for an interval I when $X_i \leftarrow x_i$ with $x_i \geq \min(I)$ is given in the Procedure getUpdatedValues.

Procedure getUpdatedValues(x_i, I)

Data: $\mathbf{x}_i \in I_D^{\mathbb{Q}}(X_i)$ and $x_i \geq \min(I)$, $I \in \mathcal{I}(\mathbf{X})$
Result: Let $I' = [\min(I), \min\{\max(I), x_i\}]$ and $\mathbf{X}' = \{X_1, ..., X_{i-1}, X_i \leftarrow x_i, X_{i+1}, ..., X_n\}$. Return the values $m(I'), es(I')$ and $es_{(2)}(I')$ computed with respect to $\mathbf{X}'$.

1 $d \leftarrow x_i - X_i^{\min}$
2 $m' \leftarrow m(I)$
3 $es' \leftarrow es(I) + d$
4 $es'_{(2)} \leftarrow es_{(2)}(I) + d^2 + 2dX_i^{\min}$
5 **if** $X_i \in M(I)$ **then**
6 $\quad m' \leftarrow m' - 1$
7 $\quad es' \leftarrow es' + X_i^{\min}$
8 $\quad es'_{(2)} \leftarrow es'_{(2)} + (X_i^{\min})^2$
9 **return** $m', es', es'_{(2)}$

The algorithm to compute $X_i^{\max}$ is given in Algorithm 5. In lines 1-5, the current value for x_i is initialized to $X_i^{\min}$ if $X_i \in R(I)$ and to ν if $X_i \in M(I)$. In the main loop in lines 6–23, the algorithm tries to discover whether there is a

value for x_i such that the value of the sum of squares reaches the maximum value $\Delta^{\max}$ while keeping the value ν in the current interval I. If this is not possible (line 14) and the upper bound $X_i^{\max}$ is not yet proved to be consistent (line 21), the procedure is repeated on the previous interval $\text{prev}(I)$ with x_i increased (line 20) such that the current value ν is equal to $\max(I)$ in the next iteration of the loop.

Algorithm 5: Filtering of X_i

```
   Data: I s.t. s ∈ si(I), X_i ∈ X
   Result: X_i^max ← min{⌊X̄_i^ℚ⌋, X_i^max}
 1 if X_i ∈ L(I) then return                    /* X_i^max is consistent */
 2 ν ← (s − es(I))/m(I)
 3 x_i ← X_i^min
 4 if X_i ∈ M(I) then
 5     x_i ← ν
 6 repeat
 7     m', es', es'_(2) ← getUpdatedValues(x_i, I)
 8     si' ← es' + m' · min(I)                  /* si' = min(si'(I)) */
 9     if m' > 0 then
10         ν' ← (s − es')/m'
11         Δ̲^ℚ ← es'_(2) + m' · (ν')^2
12         d_1 ← s − si'
13         d_2 ← (−b + √(b^2 − a · c))/a        /* a, b, c def. in text */
14         if d_2 ≤ d_1 then
15             x_i ← x_i + d_2                  /* x_i = X̄_i^ℚ */
16             X̄_i^ℤ ← getX̄_i^ℤ(x_i, I)         /* b-c upper bound */
17             X_i^max ← min{X_i^max, X̄_i^ℤ}
18             return
19         else
20             x_i ← x_i + d_1
21     if x_i ≥ X_i^max then return            /* X_i^max is consistent */
22     if I = I_1 then break
23     I ← prev(I)
24 until I = I_1
25 X_i^max ← min{X_i^max, x_i}
```

The complexity of Algorithm 5 is linear in the number of intervals in $\mathcal{I}$, which is smaller than $2n-1$. It is applied for each variable X_i to make the filtering $X_i^{\max} \leftarrow \min(X_i^{\max}, \lfloor \overline{X}_i^{\mathbb{Q}} \rfloor)$ and a similar procedure is used to filter the lower bounds $X_i^{\min}$. Hence the overall complexity of filtering $\mathbf{X}$ is $\mathcal{O}(n^2)$. This complexity can actually be improved to $\mathcal{O}(n \cdot \log(n))$ by doing a binary search on the intervals I rather than a linear search. The binary search is valid since the sum of squares increases piece-wise quadratically and continuously with the current value of x_i.

Once again, the filtering with the value $\overline{X}_i^{\mathbb{Q}}$ is not $\mathbb{Z}$-bound-consistent because it corresponds to an assignment with some variables assigned to ν which might be not integer. Line 16 of Algorithm 5 computes the $\mathbb{Z}$-bound-consistent upper bound for X_i given $\overline{X}_i^{\mathbb{Q}}$ and the current interval I. We explain in the following $\text{get}\overline{X}_i^{\mathbb{Z}}(x_i,\, I)$ detailed in Algorithm 6 running in $\mathcal{O}(m)$. The main steps of the

procedure are illustrated in Figure 3. Line 1, rounds down the value $\overline{X}_i^{\mathbb{Q}}$. This corresponds to the arrow A in Figure 3. Lines 2–6 compute the sum of squares corresponding to a $\mathbb{Z}$centered assignment of sum s with X_i assigned to $\overline{X}_i^{\mathbb{Z}}$. We can use the *getUpdatedValues* procedure to do this in constant time. Since the $\mathbb{Q}$centered assignment of sum s has been transformed into a $\mathbb{Z}$centered assignment of sum s, the sum of squares might have increased ($\underline{\Delta}^{\mathbb{Z}} > \underline{\Delta}^{\mathbb{Q}} = \Delta^{\max}$). In this case, the current value $\overline{X}_i^{\mathbb{Z}}$ at line 1 is not consistent for X_i and there is some opportunity to decrease it even more by steps of 1 until a consistent value for $X_i^{\max}$ is found (lines 7–9).

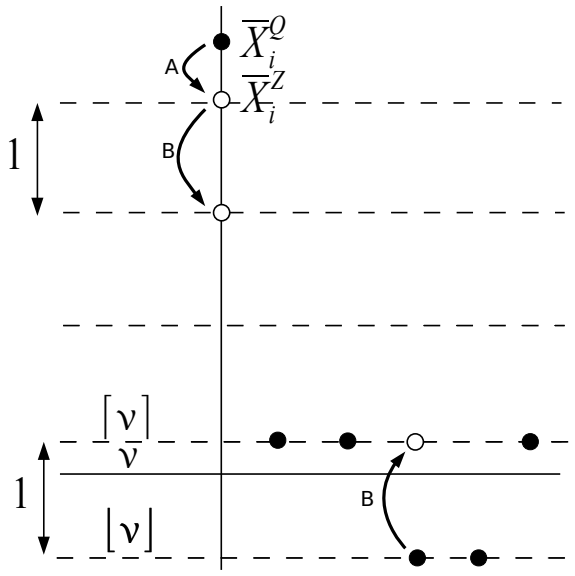

Fig. 3 Illustration of the main steps of the Algorithm 6

Algorithm 6: get$\overline{X}_i^{\mathbb{Z}}(x_i, I)$

```
1  X̄_i^ℤ ← ⌊x_i⌋
2  m', es', es'_(2) ← getUpdatedValues(X̄_i^ℤ, I)
3  ν ← (s − es')/m'
4  ν⁺ ← (s − es') mod m'
5  ν⁻ ← m' − ν⁺
6  Δ̲^ℤ ← es'_(2) + ν⁺ · ⌈ν⌉² + ν⁻ · ⌊ν⌋²
7  while Δ̲^ℤ > Δ^max do
8      Δ̲^ℤ ← Δ̲^ℤ + 2 · (⌈ν⌉ − X̄_i^ℤ)
9      X̄_i^ℤ ← X̄_i^ℤ − 1
10 return X̄_i^ℤ
```

Note that no more than m iterations are needed because as shown in Figure 3 with arrows B, each time $\overline{X}_i^{\mathbb{Z}}$ is decreased by 1, one variable assigned to $\lfloor \nu \rfloor$ must be increased in the $\mathbb{Z}$centered assignment (to maintain the correct sum s). We are guaranteed than within at most m steps the $\mathbb{Z}$centered assignment will correspond to a $\mathbb{Q}$centered assignment which was proved to exhibit a sum of squares $\leq \Delta^{\max}$ in Algorithm 5. It remains to explain how to update in constant time the value $\underline{\Delta}^{\mathbb{Z}}$ in line 8 of Algorithm 6, each time $\overline{X}_i^{\mathbb{Z}}$ is decreased by one. We refer to arrows B of Figure 3 for the visual explanation and to the next transformations to obtain the formula.

$$\begin{aligned}
\underline{\Delta}^{\mathbb{Z}'} &\leftarrow es'_{(2)} + 1 - 2\overline{X}_i^{\mathbb{Z}} + (\nu^+ + 1) \cdot \lceil \nu \rceil^2 + (\nu^- - 1) \cdot \lfloor \nu \rfloor^2 \\
&= \underline{\Delta}^{\mathbb{Z}} + 1 - 2\overline{X}_i^{\mathbb{Z}} + \lceil \nu \rceil^2 - \lfloor \nu \rfloor^2 \\
&= \underline{\Delta}^{\mathbb{Z}} + 1 - 2\overline{X}_i^{\mathbb{Z}} + (\lceil \nu \rceil - \lfloor \nu \rfloor) \cdot (\lceil \nu \rceil + \lfloor \nu \rfloor) \\
&= \underline{\Delta}^{\mathbb{Z}} + 1 - 2\overline{X}_i^{\mathbb{Z}} + 1 \cdot (2\lceil \nu \rceil - 1) \\
&= \underline{\Delta}^{\mathbb{Z}} + 2 \cdot (\lceil \nu \rceil - \overline{X}_i^{\mathbb{Z}})
\end{aligned}$$

The term $1 - 2\overline{X}_i^{\mathbb{Z}}$ is the resulting modification to $es'_{(2)}$ caused by the decreasing by one of $\overline{X}_i^{\mathbb{Z}}$, and the $+1$ and -1 applied respectively on ν^+ and ν^- are because one variable assigned to $\lfloor \nu \rfloor$ is increased by one. The other lines are simple algebraic manipulations to simplify the formula.

4 Nurse-to-patient assignment problems

As an application for `spread` we consider the daily assignment of newborn infant patients to nurses in a hospital described in [4] and solved with CP in [14]. In this problem, some infants require little attention, while others need significant care. The amount of work required by the infant during one shift is called the *acuity*. A nurse is in charge of a group of infants and the total amount of acuity is the workload of the nurse during that shift. For ensuring an optimal care quality and perceived fairness for the nurses, it is essential to balance the workload. In addition, the problem features various side constraints:

- A nurse can work in only one zone, but the patients are located in p different zones.
- A nurse cannot be responsible for more than $children^{\max}$ infants.
- The total amount of acuity of a nurse cannot exceed $acuity^{\max}$.

The balance objective and the various constraints make it very difficult to find a good solution in a reasonable time. Since nurses only work in one zone, the number of nurses assigned to each zone has already a huge impact on the quality of the balancing. In [4], the problem was tackled using a MIP model recalled in Section 4.1, but the results were not satisfactory. In this paper, we reuse our best two-step CP approach introduced in [14] in order to reach the required solution quality and scalability. This approach is recalled in Section 4.2. We improve this approach in two ways:

1. We recognize that the first step is a *discrete resource allocation* problem that can be solved optimally with well-known algorithms.

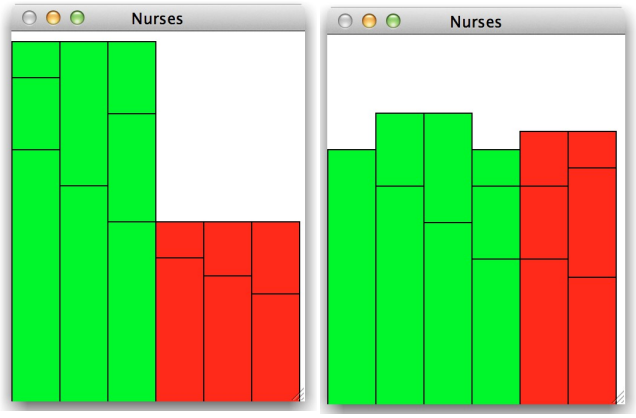

Fig. 4 Comparison of two solutions on a 6 nurses, 14 infants, and 2 zones problem. Solution on the left is obtained by minimizing the range-sum criterion. Solution on the right is obtained by minimizing the variance.

2. We show that the final result can for some instances be proved optimal using a check procedure based on the second best solution of the resource allocation problem.

4.1 The MIP Model

We now review the main variables of the MIP model from [4]. We also describe the limitations of the MIP model and suggest why a CP approach may address them. Due to space reasons, we do not reproduce the entire MIP model but readers can consult [4] for more details. The technical details presented here are sufficient for our purposes. The MIP model contains four families of variables:

1. $X_{ij} = 1$ if infant i is assigned to nurse j and 0 otherwise;
2. $Z_{jk} = 1$ if nurse j is assigned zone k and 0 otherwise;
3. $Y_{k,\max}$ is the maximum acuity of a nurse in zone k;
4. $Y_{k,\min}$ is the minimum acuity of a nurse in zone k.

All these variables are linked with linear constraints to enforce the constraints of the problem. The objective function implements what we call the *range-sum* criterion and consists of minimizing the sum of the acuity ranges of the p zones, *i.e.*,

$$\sum_{k=1}^{p} (Y_{k,\max} - Y_{k,\min}).$$

The MIP model has a fundamental limitation: The objective function may produce poorly balanced workloads. It tends to equalize the workload inside the zones but may produce huge differences among the workload of different zones. This is illustrated in Figure 4. The left solution is obtained by minimizing the range-sum criterion and the right solution by minimizing the variance (L_2 norm in the next section). The range-sum objective is minimal on the left because the workloads

inside each of the two zones are identical. Unfortunately, nurses in the first zone work twice as much as those in the second zone. The right solution is obtained by minimizing the variance and is significantly more appealing. *This illustrates clearly that "the high level objective that all nurses should be assigned an equal amount of patient acuity" [4] is not properly captured with the range-sum criterion.*

It is not immediately obvious how to remedy these problems. The variance is non-linear and is not easily modelled in a MIP approach. In addition, a CP approach may exploit the combinatorial structure in the bin-packing and the side-constraints, while the MIP relaxation is generally bad for bin-packing like problems. Finally, there are important symmetries that are not removed in their model: For a given solution, the nurses are completely interchangeable.

4.2 A Two-Step CP Model

This approach introduced in [14] first pre-computes the number of nurses assigned to each zone and then assigns the patients to nurses. This simplifies the resolution by

1. removing one degree of flexibility which is the number of nurses in each zone.
2. removing the necessity of expressing disjointedness constraint between the patients assigned to different nurses since the set of nurses that can be assigned to each patient can be pre-computed.

Furthermore pre-assigning the number of nurses in each zone, decomposes the problem. Given the pre-computed number of nurses in each zone, it is equivalent to minimize L_2 among all the nurses at once or to minimize L_2 separately inside each zone (see equation 1). So if a given number of nurses is assigned to regions, the total acuity for that region is fixed and the best we can do is minimizing the L_2 criterion inside this zone independently of other zones.

4.2.1 Step I: Finding the number of nurses in each zone

The problem of discovering the right number of nurses assigned to each nurses is crucial because the decomposition may be significantly sub-optimal if these numbers are not properly chosen. Indeed, the number of nurses assigned to each zone has a crucial impact on the quality of the balancing. However, after visualizing some optimal solutions, we observed that the workloads of the nurses are extremely well balanced (almost the same) inside the zones. This suggested solving a relaxation of the problem to discover a good distribution of the nurses to the zones. The relaxation allows the acuity of a child in a zone to be distributed among the nurses of that zone (continuous relaxation of the acuity). Since the acuity of a child can be split, the relaxed problem will have an optimal solution where the nurses of a zone have exactly the same workload $\frac{A_k}{x_k}$, *i.e.*, the total acuity $A_k = \sum_{i \in \mathcal{P}_k} a_i$ of zone k divided by the number of nurses x_k in zone k. This is schematically illustrated on Figure 5 for a two-zone relaxation problem and stated in Proposition 1.

Proposition 1 *An optimal solution of the relaxed problem must have the same workload for all the nurses in a given zone.*

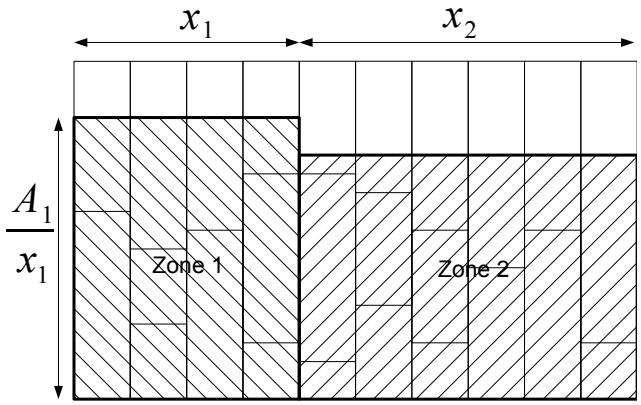

Fig. 5 Illustration of a solution of the relaxation solved to find the number of nurses in each zone.

Proof If this is not the case, the sum of squares inside that zone can be reduced by making closer the workload given to two nurses (same argument as the one used in proof of Theorem 1) .

The mathematical formulation of the relaxed problem is

$$\min \sum_{k=1}^{p} x_k \cdot \left(\frac{A_k}{x_k}\right)^2 \tag{6}$$

$$s.t. \sum_{k=1}^{p} x_k = m \tag{7}$$

$$x_k \in \mathbb{Z}_0^+ \tag{8}$$

with m integer and $m \geq p$. The workload of all the nurses of zone k is $\frac{A_k}{x_k}$.

Solving the Relaxation This problem is a discrete resource allocation problem with separable objective convex function [5]. It can be solved optimally with the greedy `INCREMENT` algorithm described in [3]. The algorithm starts with one nurse for each zone, and consecutively increments the number of nurses from the zone that will increase the least its L_2. When the number of nurses m is reached, the allocation is optimal [3]. Algorithm 7 is an instance of INCREMENT method applied to our problem running in $\mathcal{O}(p + m \cdot \log p)$:

The complexity is obtained using a heap data structure to select in $\mathcal{O}(\log p)$ the $k = \text{argmax}_i\{\frac{A_i^2}{x_i} - \frac{A_i^2}{x_i+1}\}$ at each iteration and it takes a linear time $\mathcal{O}(p)$ to initialize it.

[3] A complete proof of correction of the algorithm can be found in [10].

Algorithm 7: Solve the relaxation problem

$x_i \leftarrow 1$ for $i \in [1..p]$
while $\sum_{i \in [1..p]} x_i < m$ **do**
  $k \leftarrow \text{argmax}_i \{ \frac{A_i^2}{x_i} - \frac{A_i^2}{x_i+1} \}$
  $x_k \leftarrow x_k + 1$
return x

4.2.2 Step II: CP Model

Listing 1 gives the OscaR CP [6] model for the second step. In particular it considers the allocation of patients of zone i given the number of nurses (nbNursesInZone(i)) assigned to this zone in Step I. This model minimizes the sum of squared acuities (`sum2` variable). The constraint `binpacking` is the one introduced in [15] making the link between the acuity of each patient and the total acuity of each nurse. The `gcc` constraint is the global cardinality constraint introduced in [9] forcing each nurse to have between 1 and 3 patients. The search is a nondeterministic search [2] that breaks the symmetries dynamically, trying at each node to assign the chosen patient to the nurses that already have some patients plus at most one nurse without any patient yet.

Listing 1 Scala/OscaR CP Model solving the zone i

```
val cp = CPSolver()
// for each patient, the nurse assigned to this patient (decision variables)
val nurseOfPatient = Array.fill(nbPatientsInZone(i))(CPVarInt(cp,0 until nbNursesInZone(i)))
// for each nurse, his/her acuity
val acuityOfNurse = Array.fill(nbNursesInZone(i))(CPVarInt(cp,1 to 105))
// sum of squares of the acuities
val sum2 = CPVarInt(cp, 0 to Int.MaxValue)
cp.minimize(sum2) subjectTo {
  // acuityByZone(i)(j) is the acuity required by infant j (located in zone i)
  cp.add(spread(acuityOfNurse,acuityByZone(i).sum,sum2))
  // each nurse must have between 1 and 3 patients
  cp.add(gcc(nurseOfPatient,0 until nbNursesInZone(i),1,3))
  // binpacking constraint linking acuity of patients and nurses
  cp.add(binpacking(nurseOfPatient,acuityByZone(i),acuityOfNurse))
} exploration {
  // dynamic symmetry breaking search
  while (!allBounds(nurseOfPatient)) {
    val maxUsed = nurseOfPatient.maxBoundOrElse(-1)
    val x = nurseOfPatient.minDomNotBound
    cp.branchAll(0 to maxUsed+1)(v => cp.post(x == v))
  } run()
}
```

4.2.3 Proving optimality of our solutions

We define the *candidate optimal solution* to be the solution obtained after the optimization in each zone independently, based on the nurse allocation obtained from the continuous relaxation. This candidate optimal solution obtained with our two-step approach is not optimal if another allocation of the number of nurses to the different zones would have led to a better sum of squares objective at the end of the second step. There are a finite number of ways of allocating the nurses to the

different zones. Optimality could be proven by solving the second step for every possible allocation. We can avoid that effort if the objective of the second best continuous relaxation of the first step is actually larger than the sum of squares of our candidate optimal solution. The question to answer now is how to compute the second best continuous relaxation. This second best solution is obtained by taking the second best decision when selecting k during the last iteration of Algorithm 7. More precisely when $\sum_i x_i = m - 1$, the second best should be selected instead of the best one with argmax_i. The correctness of this procedure is guaranteed by the separable convex nature of the function to minimize. Alternatively to find the second best solution, one can start from the optimal solution with $\sum_i x_i = m$ then update this solution by applying the change $x_i \leftarrow x_i - 1$, $x_j \leftarrow x_j + 1$ that minimizes the increase: $\text{argmin}_{i \neq j | x_j > 1}\{(\frac{A_i^2}{x_i+1} - \frac{A_i^2}{x_i}) + (\frac{A_i^2}{x_j-1} - \frac{A_j^2}{x_j})\}$.

4.3 Experimental Results

Problem Instances Reference [4] specifies a statistical model to generate instances very similar to their real instances. This statistical model was also used to measure the robustness of their solution technique with respect to the number of nurses, the number of infants, and the number of zones. The model contains a single parameter: the number of zones. The maximum acuity per nurse is fixed to $acuity^{\max} = 105$ and the maximum number of infants per nurse is fixed to $children^{\max} = 3$. The instance generator fixes the number of nurses, the number of infants, the acuity, and the zone of each infant. The different steps to generate an instance are as follows:

- The number of patients in a zone is specified by a Poisson random variable with mean 3.8 and offset by 10.
- The acuity Y of a patient is obtained by first generating a number $X \sim Binomial(n = 8, p = 0.23)$ and then choosing the number $Y \sim Unif(10 \cdot (X + 1), 10 \cdot (X + 1) + 9)$.
- The total number of nurses is obtained by solving a First Fit Decreasing (FFD) procedure in each zone. More precisely, the total number is the number of nurses found in each zone by the FFD procedure. The FFD procedure starts by ranking the patients in decreasing acuity. Then, the patient with the highest acuity is assigned to the first nurse. The next patients are assigned successively to the first nurse that can accommodate them without violating the maximum acuity and the number of patient constraints.

We generated 10 instances of 3 a zones problem, and three larger instances with 6, 15 and 20 zones. The results are given on Table 2. The columns respectively represent the number of nurses, the number of patients, the total number backtracks, the overall time, the average workload per nurse, the lower bound on the standard deviation coming from the continuous relaxation (Step I), the standard deviation of the best solution, and the lower bound computed from the second best continuous relaxation. It can be seen that the computation times are very short (at most one second) and except for one 3 zones instance and the 20 zones instance (in bold), all the results are proven optimal since the lower bound of the second best continuous relaxation is always larger than the standard deviation of

the optimal solution. Note that the MIP formulation from [4] could hardly solve problems with 2 zones in less than 30 minutes.

Table 2 Results on 3, 6, 15 and 20 zones problem instances

zones	m	n	#fails	time(ms)	avg load	lb1 std	std optimal	lb2 std
3	15	42	11973	503	84.20	2.90	3.04	11.74
3	18	43	4473	173	79.78	5.48	5.84	5.87
3	17	43	17155	484	81.41	3.42	4.46	8.95
3	17	42	14089	489	83.82	5.58	5.65	6.74
3	18	43	43421	1188	81.00	4.93	5.77	7.36
3	14	38	3335	110	85.36	2.13	3.08	13.23
3	19	48	23041	573	87.42	2.26	3.07	9.18
3	16	44	19817	598	84.88	6.38	**6.70**	**6.68**
3	19	49	20370	454	86.00	1.91	2.49	9.64
3	17	41	7606	192	82.18	3.04	3.40	9.41
6	31	78	12019	565	84.58	3.57	4.20	6.85
15	71	198	38651	2272	81.95	5.17	5.33	5.34
20	102	258	1176852	25169	82.71	4.87	**5.54**	**5.27**

The last three lines of table 2 are obtained with instances from a 6,15 and 20 zones. The solution for the 15 zones instance is depicted in Figure 6. This solution is also proven optimal since the second best lower bound is larger than the standard deviation computed for our best solution.

Fig. 6 Solution of a 15 zones instance obtained with the two-step approach.

5 Conclusion

We have introduced the bound-consistent filtering algorithms for the spread constraint on finite integer domains. We used spread to solve a real-life nurse-to-patient assignment problem using a two-step decomposition approach. We show that in practice this decomposition is optimal on typical instances. Our implementation of the spread constraint is open source and available on the repository of OscaR [6]. All the instances used in our experiments are available on `http://becool.info.ucl.ac.be/resources/nurse-patient-assignment`.

References

1. Gorard, S.: Revisiting a 90-year-old debate: The advantages of the mean deviation. British Journal of Educational Studies pp. 417–439 (2005)
2. Hentenryck, P., Michel, L.: Nondeterministic control for hybrid search. Constraints **11**(4), 353–373 (2006)
3. Ibaraki, T., Katoh, N.: Resource Allocation Problems: Algorithmic Approaches. The MIT Press, Cambridge, Mass. (1988)
4. Mullinax, C., Lawley, M.: Assigning patients to nurses in neonatal intensive care. Journal of the Operational Research Society **53**, 25–35 (2002)
5. N. Katoh, T.I., Mine, H.: A polynomial time algorithm for the resource allocation problem with a convex objective function. The Journal of the Operational Research Society pp. 449–455 (1979)
6. OscaR Team: OscaR: Scala in OR (2012). Available from `https://bitbucket.org/oscarlib/oscar`
7. Pesant, G.: Constraint-based rostering. The 7th International Conference on the Practice and Theory of Automated Timetabling PATAT 2008 (2008)
8. Pesant, G., Régin, J.: Spread: A balancing constraint based on statistics. Lecture Notes in Computer Science **3709**, 460–474 (2005)
9. Régin, J.C.: Generalized arc consistency for global cardinality constraint. AAAI-96 pp. 209–215 (1996)
10. Schaus, P.: Balancing and bin-packing constraints in constraint programming. PhD thesis, Université catholique de Louvain, INGI, (2009)
11. Schaus, P., Deville, Y., Dupont, P.: Bound-consistent deviation constraint. 13th International Conference on Principles and Practice of Constraint Programming (CP 2007) **4741** (2007)
12. Schaus, P., Deville, Y., Dupont, P., Régin, J.: Simplification and extension of spread. 3d Workshop on Constraint Propagation And Implementation (2006)
13. Schaus, P., Deville, Y., Dupont, P., Régin, J.: The deviation constraint. Proceedings of CP-AI-OR **4510**, 269–284 (2007)
14. Schaus, P., Van Hentenryck, P., Régin, J.C.: Scalable load balancing in nurse to patient assignment problems. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 248–262. Springer (2009)
15. Shaw, P.: A constraint for bin packing. In: Principles and Practice of Constraint Programming CP 2004, pp. 648–662 (2004)
16. Simonis, H.: Models for global constraint applications. Constraints **12**, 63–92 (2007)