Supplemental Material to A Propagator Design Framework for Constraints over Sequences

Jean-Noël Monette and **Pierre Flener** and **Justin Pearson**
Uppsala University, Dept of Information Technology
751 05 Uppsala, Sweden
FirstName.LastName@it.uu.se

Abstract

This supplemental material gives the extended definitions of the tuple operations, lists the definition of some transformation operators, details the examples about DEVIATION, SEQBIN, and LONGESTPLATEAU, and describes the experimental protocol for the experiments.

1 Extended Definitions of Tuple Operations

The *projection* $\langle v_{i_1}, \ldots, v_{i_k}\rangle$ of a tuple $\langle v_1, \ldots, v_n\rangle$ onto a subset of its components $i_1, \ldots, i_k$ is written $\pi_{i_1,\ldots,i_k}(\langle v_1, \ldots, v_n\rangle)$. The projection $\pi_{i_1,\ldots,i_k}(S)$ of a tuple set S is the set of the projections of the tuples in S ($\pi_{i_1,\ldots,i_k}(S) = \{\langle v_{i_1}, \ldots, v_{i_k}\rangle \mid \langle v_1, \ldots, v_n\rangle \in S\}$).

The *concatenation* of two tuples is defined as $\langle v_1, \ldots, v_n\rangle \cdot \langle w_1, \ldots, w_m\rangle = \langle v_1, \ldots, v_n, w_1, \ldots, w_m\rangle$. The *Cartesian product* of two tuple sets is the pointwise lifting of the concatenation: $S \times T = \{s \cdot t \mid s \in S \wedge t \in T\}$.

The *restriction* (written σ) of a tuple set S is the subset containing the tuples whose projection on some components $i_1, \ldots, i_k$ is contained in another set T: $\sigma_{i_1,\ldots,i_k;T}(S) = \{s \mid s \in S \wedge \pi_{i_1,\ldots,i_k}(s) \in T\}$.

The *selection* operator in the article can be defined in terms of the projection and restriction operators: $\rho_v(S) = \pi_2(\sigma_{1;\{v\}}(S))$. In this appendix, we make use of a slightly more general selection operator: $\rho_v^{i_i,\ldots,i_k}(S) = \pi_{i_1,\ldots,i_k}(\sigma_{1;\{v\}}(S))$.

2 Transformation Operators

We list here a few transformation operators used in the refinement of pruning rules (beside the ones described in the main text).

$\text{smap\_o\_filter}(\lambda\langle\rangle\,.\,e, \lambda t\,.\,\text{true}, \langle\rangle) \rightsquigarrow e$ (Simplification)

$\text{smap\_o\_filter}(\lambda\langle\rangle\,.\,e, \lambda\langle\rangle\,.\,b, \langle\rangle) \rightsquigarrow$ if b then e else $\emptyset$ (Simplification)

$\langle k, z\rangle \in \text{dom}(\mathbf{XY}) \rightsquigarrow z \in \rho_k^2(\text{dom}(\mathbf{XY}))$ (Isolation of a variable)

$\max(A, B) \rightsquigarrow \text{bmax}(\langle A, B\rangle)$ (Grouping)

$C_1 \wedge C_2 \rightsquigarrow C_1$ (Weakening)

$S_1 \cap S_2 \rightsquigarrow S_1$ (Weakening)

$\text{lift}[v_1 = S_2] \rightsquigarrow v_1 \in S_2$ (Lifting of equality from values to sets)

$\text{lift}[S_1 = S_2] \rightsquigarrow S_1 \cap S_2 \neq \emptyset$ (Lifting of equality from values to sets)

3 Simplified DEVIATION Example

This is DEVIATION using the following DP formulation as in Examples 1, 2, and 3 of the main text:

$$S_0 = 0 \wedge D_0 = 0 \qquad (C_\text{F})$$

$$\left(\begin{array}{c} S_i = S_{i-1} + X_i \wedge \\ D_i = D_{i-1} + |X_i - m| \end{array}\right) \quad i \in 1..n \qquad (C_i)$$

$$S_n = m \cdot n \wedge D_n = D \qquad (C_\text{L})$$

After introduction of the $\mathbf{SD}_i$ tuple variables, it becomes:

$$\mathbf{SD}_0 = \langle 0, 0\rangle \qquad (C_\text{F})$$

$$\mathbf{SD}_i = \mathbf{SD}_{i-1} + \langle X_i, |X_i - m|\rangle \quad i \in 1..n \qquad (C_i)$$

$$\mathbf{SD}_n = \langle m \cdot n, D\rangle \qquad (C_\text{L})$$

$$\mathbf{SD}_i = \langle S_i, D_i\rangle \quad i \in 0..n \qquad (C_{\mathbf{A}_i})$$

Constraint $C_{\mathbf{A}_i}$ and variables S_i and D_i can be removed, assuming S_i and D_i are not used in any other constraint. We now describe a complete non-incremental and stateless propagator. To reproduce an incremental version is beyond the scope of this appendix. The propagator makes use of a preprocessing step. This preprocessing is used to reduce the domain of the variables using a weaker domain representation and weaker pruning rules. In practice, we introduce a $\mathbf{SDP}_i$ tuple variable with domain representation $\mathcal{I}^2$, while the original $\mathbf{SD}_i$ variable has a domain representation $\mathcal{E} \rightarrow \mathcal{I}$. The preprocessing corresponds to the cutoff presented in (Pesant 2011), which is useful to achieve the given time complexity. The propagator is described as follows. The individual pruning rules are described below.

filter $\mathbf{SDP}_n$ based on C_L with rule r_{p1}
for $i = n$ **down to** 1
  filter $\mathbf{SDP}_{i-1}$ based on C_i with rule r_{p2}
filter $\mathbf{SD}_0$ based on C_F with rule r_1
for $i = 1$ **to** n
  filter $\mathbf{SD}_i$ based on C_i and $\mathbf{SDP}_i$ with rule r_2
filter $\mathbf{SD}_n$ based on C_L with rule r_3
filter D based on C_L with rule r_4
for $i = n$ **down to** 1
  filter $\mathbf{SD}_{i-1}$ based on C_i with rule r_5
  filter X_i based on C_i with rule r_6

The link between the two tuple variable domain representations is performed in pruning rule r_2 by enforcing that $\pi_1(\mathbf{SD}_i)$ is a subset of $\pi_1(\mathbf{SD^P}_i)$. We present all pruning rules in the form "Y in e", meaning that the domain of variable Y must be restricted to become a (non-strict) subset of expression e. The preprocessing pruning rules are the following ones:

$$\mathbf{SD^P}_n \text{ in } \{m \cdot n\} \times \text{bnd}(D) \qquad (r_{p1})$$
$$\mathbf{SD^P}_{i-1} \text{ in } \mathbf{SD^P}_i - (\text{bnd}(X_i) \times |\text{bnd}(X_i) - m|) \qquad (r_{p2})$$

where $\text{bnd}(X)$ is the smallest interval enclosing the domain of an integer variable X (i.e., it is a shortcut for $\min(\text{dom}(X))..\max(\text{dom}(X))$). The other pruning rules are:

$$\mathbf{SD}_0 \text{ in } \{\langle 0, 0\rangle\} \qquad (r_1)$$
$$\mathbf{SD}_i \text{ in } \text{smap\_o\_filter}(\qquad (r_2)$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, \{s_{i-1} + x_i\} \times (\rho^2_{s_{i-1}}(\text{dom}(\mathbf{SD}_{i-1})) + \{|x_i - m|\}),$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, (s_{i-1} + x_i) \in \pi_1(\mathbf{SD^P}_i),$$
$$\quad \text{dom}(X_i) \times \pi_1(\text{dom}(\mathbf{SD}_{i-1})))$$
$$\mathbf{SD}_n \text{ in } \{m \cdot n\} \times \text{dom}(D) \qquad (r_3)$$
$$D \text{ in } \pi_2(\text{dom}(\mathbf{SD}_n)) \qquad (r_4)$$
$$\mathbf{SD}_{i-1} \text{ in } \text{smap\_o\_filter}(\lambda \langle x_i, s_i\rangle \,.\, \{s_i - x_i\} \times \left(\rho^2_{s_i}(\text{dom}(\mathbf{SD}_i)) - \{|x_i - m|\}\right), \qquad (r_5)$$
$$\quad \lambda t \,.\, \text{true}, \text{dom}(X_i) \times \pi_1(\text{dom}(\mathbf{SD}_i)))$$
$$X_i \text{ in } \text{smap\_o\_filter}(\qquad (r_6)$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, \{x_i\},$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, \rho^2_{s_{i-1}}(\text{dom}(\mathbf{SD}_{i-1})) + \{|x_i - m|\} \cap \rho^2_{(s_{i-1}+x_i)}(\text{dom}(\mathbf{SD}_i)) \neq \emptyset,$$
$$\quad \text{dom}(X_i) \times \pi_1(\text{dom}(\mathbf{SD}_{i-1})))$$

4 Complete DEVIATION Example

The DP formulation of DEVIATION used in Examples 1, 2, and 3 of the main text as been simplified for ease of presentation. (Pesant 2011) actually implicitly uses another DP formulation than the one of Example 1, namely:

$$S_0 = 0 \wedge D^f_0 = 0 \wedge D^b_0 = D \wedge D^t_0 = D \qquad (C_\text{F})$$
$$\left(\begin{array}{c} S_i = S_{i-1} + X_i \wedge \\ D^f_i = D^f_{i-1} + |X_i - m| \wedge \\ D^b_i = D^b_{i-1} - |X_i - m| \wedge \\ D^f_i + D^b_i = D^t_i \wedge \\ D^t_i = D^t_{i-1} \end{array}\right) \quad i \in 1..n \qquad (C_i)$$
$$S_n = m \cdot n \wedge D^f_n = D \wedge D^b_n = 0 \wedge D^t_n = D \qquad (C_\text{L})$$

where D^f_i is the partial deviation for the variables from 1 to i, D^b_i is the partial deviation for the variables from $i+1$ to n, and D^t_i is the total deviation. All D^t_i are equal to each other and to D. The 4-tuple link variable $\mathbf{SD}_i$ represents S_i, D^f_i, D^b_i, and D^t_i, and we use domain representation $(\mathcal{E} \to \mathcal{I}^2) \times \mathcal{I}$. As we use a Cartesian product to separate D^t_i from the rest of the tuple, an implementation can ensure that the domains of all D^t_i are implemented by a unique object (as they are all equal). To simplify presentation, we will leave the D^t_i out of the formulation and use D in their place.

Hence from now on, $\mathbf{SD}_i$ represents S_i, D^f_i, and D^b_i, with domain representation $\mathcal{E} \to \mathcal{I}^2$. The DP formulation after introduction of the tuple variables is:

$$\mathbf{SD}_0 = \langle 0, 0, D\rangle \qquad (C_\text{F})$$
$$\left(\begin{array}{c} \mathbf{SD}_i = \mathbf{SD}_{i-1} + \langle X_i, |X_i - m|, -|X_i - m|\rangle \\ \wedge\ \pi_2(\mathbf{SD}_i) + \pi_3(\mathbf{SD}_i) = D \end{array}\right) \quad i \in 1..n \qquad (C_i)$$
$$\mathbf{SD}_n = \langle m \cdot n, D, 0\rangle \qquad (C_\text{L})$$

The control is the same as in the simplified version. We use the symbol $\mathcal{U}$ (representing the universe set, i.e., $\mathbb{Z}$ or a subset thereof) when some component of a tuple variable is not restricted by a pruning rule. Here are the pruning rules:

$$\mathbf{SD^P}_n \text{ in } \{m \cdot n\} \times \mathcal{U}^2 \qquad (r_{p1})$$
$$\mathbf{SD^P}_{i-1} \text{ in } (\pi_1(\mathbf{SD^P}_i) - \text{bnd}(X_i)) \times \mathcal{U}^2 \qquad (r_{p2})$$
$$\mathbf{SD}_0 \text{ in } \{\langle 0, 0\rangle\} \times \text{dom}(D) \qquad (r_1)$$
$$\mathbf{SD}_i \text{ in } \text{smap\_o\_filter}(\qquad (r_2)$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, \{s_{i-1} + x_i\} \times (\rho^2_{s_{i-1}}(\text{dom}(\mathbf{SD}_{i-1})) + \{|x_i - m|\}) \times \mathcal{U},$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, (s_{i-1} + x_i) \in \pi_1(\mathbf{SD^P}_i),$$
$$\quad \text{dom}(X_i) \times \pi_1(\text{dom}(\mathbf{SD}_{i-1})))$$
$$\mathbf{SD}_n \text{ in } \{m \cdot n\} \times \text{dom}(D) \times \{0\} \qquad (r_3)$$
$$D \text{ in } \pi_2(\text{dom}(\mathbf{SD}_n)) \qquad (r_4)$$
$$\mathbf{SD}_{i-1} \text{ in } \text{smap\_o\_filter}(\lambda \langle x_i, s_i\rangle \,.\, \{s_i - x_i\} \times \mathcal{U} \times (\rho^3_{s_i}(\text{dom}(\mathbf{SD}_i)) + \{|x_i - m|\}), \qquad (r_5)$$
$$\quad \lambda t \,.\, \text{true}, \text{dom}(X_i) \times \pi_1(\text{dom}(\mathbf{SD}_i)))$$
$$X_i \text{ in } \text{smap\_o\_filter}(\qquad (r_6)$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, \{x_i\},$$
$$\quad \lambda \langle x_i, s_{i-1}\rangle \,.\, \rho^2_{s_{i-1}}(\text{dom}(\mathbf{SD}_{i-1})) + \{|x_i - m|\} + \rho^3_{(s_{i-1}+x_i)}(\text{dom}(\mathbf{SD}_i)) \cap \text{dom}(D) \neq \emptyset,$$
$$\quad \text{dom}(X_i) \times \pi_1(\text{dom}(\mathbf{SD}_{i-1})))$$

5 Simplified SEQBIN Example

We now present a propagator for SEQBIN based on the formulation given in the main text (Example 4). The DP formulation, after introduction of the $\mathbf{XS}_i$ tuple variables, is:

$$\pi_2(\mathbf{XS}_0) = 0 \qquad (C_\text{F})$$
$$\left(\begin{array}{c} \pi_2(\mathbf{XS}_i) = \pi_2(\mathbf{XS}_{i-1}) + \\ [B(\pi_1(\mathbf{XS}_{i-1}), \pi_1(\mathbf{XS}_i))] \\ \wedge\ D(\pi_1(\mathbf{XS}_{i-1}), \pi_1(\mathbf{XS}_i)) \end{array}\right) \quad i \in 1..n \qquad (C_i)$$
$$\pi_2(\mathbf{XS}_n) = S \qquad (C_\text{L})$$
$$\pi_1(\mathbf{XS}_i) = X_i \quad i \in 0..n \qquad (C_{\mathbf{A}_i})$$

Note that we simplified the $C_{\mathbf{A}_i}$ constraints to eliminate the S_i variables introduced by the DP formulation. The control of the set of pruning rules is the following one:

filter $\mathbf{XS}_0$ based on C_F and $C_{\mathbf{A}_0}$ with rule r_1
for $i = 1$ **to** n
  filter $\mathbf{XS}_i$ based on C_i and $C_{\mathbf{A}_i}$ with rule r_2
filter S based on C_L with rule r_3
filter $\mathbf{XS}_n$ based on C_L with rule r_4
for $i = n$ **down to** 1
  filter $\mathbf{XS}_{i-1}$ based on C_i with rule r_5
for $i = n$ **down to** 0
  filter X_i based on $C_{\mathbf{A}_i}$ with rule r_6

Pruning rules r_1 and r_2 are used to propagate two constraints at the same time. Those pruning rules can also be derived from the DP formulation: their instantiation is based on the conjunction of the two constraints. Here are the refined pruning rules:

$$
\begin{array}{ll}
\mathbf{XS}_0 \text{ in } \mathrm{dom}(X_0) \times \{0\} & (r_1)\\
\mathbf{XS}_i \text{ in } \mathrm{smap\_o\_filter}(& (r_2)\\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, \{x_i\} \times & \\
\qquad (\rho^2_{x_{i-1}}(\mathrm{dom}(\mathbf{XS}_{i-1})) + \{[B(x_{i-1}, x_i)]\}), & \\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, D(x_{i-1}, x_i), & \\
\quad \mathrm{dom}(X_i) \times \pi_1(\mathrm{dom}(\mathbf{XS}_{i-1}))) & \\
S \text{ in } \pi_2(\mathrm{dom}(\mathbf{XS}_n)) & (r_3)\\
\mathbf{XS}_n \text{ in } \pi_1(\mathrm{dom}(\mathbf{XS}_n)) \times \mathrm{dom}(S) & (r_4)\\
\mathbf{XS}_{i-1} \text{ in } \mathrm{smap\_o\_filter}(& (r_5)\\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, \{x_{i-1}\} \times & \\
\qquad (\rho^2_{x_i}(\mathrm{dom}(\mathbf{XS}_i)) - \{[B(x_{i-1}, x_i)]\}), & \\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, D(x_{i-1}, x_i), & \\
\quad \pi_1(\mathrm{dom}(\mathbf{XS}_i)) \times \pi_1(\mathrm{dom}(\mathbf{XS}_{i-1}))) & \\
X_i \text{ in } \pi_1(\mathrm{dom}(\mathbf{XS}_i)) & (r_6)
\end{array}
$$

6 Complete SEQBIN Example

We now present the description of SEQBIN for the propagator presented in (Petit, Beldiceanu, and Lorca 2011; Katsirelos, Narodytska, and Walsh 2012). Here is the DP formulation:

$$S^f_0 = 0 \wedge S^b_0 = S_0 \qquad (C_{\mathrm{F}})$$

$$\left(\begin{array}{c} S^f_i = S^f_{i-1} + [B(X_{i-1}, X_i)] \ \wedge \\ S^b_{i-1} = S^b_i + [B(X_{i-1}, X_i)] \ \wedge \\ S^f_{i-1} + S^b_{i-1} = S_{i-1} \wedge \\ S_i = S_{i-1} \wedge \\ D(X_{i-1}, X_i) \end{array}\right) \quad i \in 1..n \qquad (C_i)$$

$$S^f_n = S_n \wedge S^b_n = 0 \wedge S_n = S \qquad (C_{\mathrm{L}})$$

The link variables are X_i (current variable), S^f_i (forward count), S^b_i (backward count), and S_i (total count). The S_i are all equal and introduced to fit in the framework. They can all be replaced by S (as for D in the DEVIATION example) but we keep them to show another possible way.

We introduce a tuple variable $\mathbf{XS}_i$ for each link with a $(\mathcal{E} \to \mathcal{Z}^2) \times \mathcal{E}$ representation. The last component is the same for all i and can be implemented by a shared object.

The control is the same as for the simplified version. Here are the refined pruning rules:

$$
\begin{array}{ll}
\mathbf{XS}_0 \text{ in } \mathrm{dom}(X_0) \times \{0\} \times \mathrm{dom}(S) \times \mathrm{dom}(S) & (r_1)\\
\mathbf{XS}_i \text{ in } \mathrm{smap\_o\_filter}(& (r_2)\\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, \{x_i\} \times & \\
\qquad (\rho^2_{x_{i-1}}(\mathrm{dom}(\mathbf{XS}_{i-1})) + \{[B(x_{i-1}, x_i)]\}) \times & \\
\qquad\qquad \mathcal{U} \times \mathrm{dom}(S), & \\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, D(x_{i-1}, x_i), & \\
\quad \mathrm{dom}(X_i) \times \pi_1(\mathrm{dom}(\mathbf{XS}_{i-1}))) & \\
S \text{ in } \pi_2(\mathrm{dom}(\mathbf{XS}_n)) & (r_3)\\
\mathbf{XS}_n \text{ in } \mathrm{smap\_o\_filter}(& (r_4)\\
\quad \lambda\,\langle x_n\rangle \,.\, \{x_n\} \times \mathcal{U} \times \{0\} \times \mathrm{dom}(S), & \\
\quad \lambda\,\langle x_n\rangle \,.\, \rho^2_{x_n}(\mathrm{dom}(\mathbf{XS}_n)) \cap \pi_4(\mathrm{dom}(\mathbf{XS}_i)) \neq \emptyset, & \\
\qquad \pi_1(\mathrm{dom}(\mathbf{XS}_n))) & \\
\mathbf{XS}_{i-1} \text{ in } \mathrm{smap\_o\_filter}(& (r_5)\\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, \{x_{i-1}\} \times \mathcal{U} \times & \\
\qquad (\rho^3_{x_i}(\mathrm{dom}(\mathbf{XS}_i)) + \{[B(x_{i-1}, x_i)]\}) \times \mathrm{dom}(S), & \\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, D(x_{i-1}, x_i) \wedge & \\
\qquad (\rho^2_{x_{i-1}}(\mathrm{dom}(\mathbf{XS}_{i-1})) + \rho^3_{x_i}(\mathrm{dom}(\mathbf{XS}_i)) + & \\
\qquad\quad \{[B(x_{i-1}, x_i)]\}) \cap \pi_4(\mathrm{dom}(\mathbf{XS}_i)) \neq \emptyset, & \\
\quad \pi_1(\mathrm{dom}(\mathbf{XS}_i)) \times \pi_1(\mathrm{dom}(\mathbf{XS}_{i-1}))) & \\
X_i \text{ in } \pi_1(\mathrm{dom}(\mathbf{XS}_i)) & (r_6)
\end{array}
$$

7 Complete LONGESTPLATEAU Example

We use the following DP formulation of Section 7 in the main text:

$$K_0 = 1 \wedge M_0 = 1 \qquad (C_{\mathrm{F}})$$

$$\begin{array}{ll}
\textbf{if } X_i = X_{i-1} & \\
\textbf{then } K_i = K_{i-1} + 1 \wedge M_i = M_{i-1} & i \in 1..n \quad (C_i)\\
\textbf{else } K_i = 1 \wedge M_i = \max(M_{i-1}, K_{i-1}) &
\end{array}$$

$$L = \max(M_n, K_n) \qquad (C_{\mathrm{L}})$$

After introduction of the $\mathbf{XKM}_i$ tuple variables, it becomes:

$$\pi_{2,3}(\mathbf{XKM}_0) = \langle 1, 1\rangle \qquad (C_{\mathrm{F}})$$

$$\begin{array}{ll}
\textbf{if } \pi_1(\mathbf{XKM}_i) = \pi_1(\mathbf{XKM}_{i-1}) & \\
\textbf{then } \pi_{2,3}(\mathbf{XKM}_i) = \pi_{2,3}(\mathbf{XKM}_{i-1}) + \langle 1, 0\rangle & i \in 1..n\\
\textbf{else } \pi_{2,3}(\mathbf{XKM}_i) = \langle 1, \mathrm{bmax}(\pi_{2,3}(\mathbf{XKM}_{i-1}))\rangle & \\
& (C_i)
\end{array}$$

$$L = \mathrm{bmax}(\pi_{2,3}(\mathbf{XKM}_n)) \qquad (C_{\mathrm{L}})$$

$$\pi_1(\mathbf{XKM}_i) = X_i \quad i \in 0..n \qquad (C_{\mathbf{A}_i})$$

where $\mathrm{bmax}(\langle t_1, t_2\rangle) = \max(t_1, t_2)$. The control is identical to the one used for SEQBIN (modulo variable names). The refined pruning rules are:

$$
\begin{array}{ll}
\mathbf{XKM}_0 \text{ in } \mathrm{dom}(X_0) \times \{\langle 1, 1\rangle\} & (r_1)\\
\mathbf{XKM}_i \text{ in } \mathrm{smap\_o\_filter}(& (r_2)\\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, \{x_i\} \times (\textbf{if } x_i = x_{i-1} & \\
\qquad \textbf{then } \rho^{2,3}_{x_{i-1}}(\mathrm{dom}(\mathbf{XKM}_{i-1})) + \langle 1, 0\rangle & \\
\qquad \textbf{else } \{1\} \times \mathrm{bmax}(\rho^{2,3}_{x_{i-1}}(\mathrm{dom}(\mathbf{XKM}_{i-1})))), & \\
\quad \lambda t \,.\, \mathrm{true}, \mathrm{dom}(X_i) \times \pi_1(\mathrm{dom}(\mathbf{XKM}_{i-1}))) & \\
L \text{ in } \mathrm{smap\_o\_filter}(& (r_3)\\
\quad \lambda\,\langle x_n\rangle \,.\, \mathrm{bmax}(\rho^{2,3}_{x_n}(\mathrm{dom}(\mathbf{XKM}_n))), & \\
\quad \lambda t \,.\, \mathrm{true}, \pi_1(\mathrm{dom}(\mathbf{XKM}_n))) & \\
\mathbf{XKM}_n \text{ in } \mathrm{smap\_o\_filter}(& (r_4)\\
\quad \lambda\,\langle x_n\rangle \,.\, \{x_n\} \times \mathrm{bmax}^{-1}(\mathrm{dom}(L)) & \\
\quad \lambda t \,.\, \mathrm{true}, \pi_1(\mathrm{dom}(\mathbf{XKM}_n))) & \\
\mathbf{XKM}_{i-1} \text{ in } \mathrm{smap\_o\_filter}(& (r_5)\\
\quad \lambda\,\langle x_i, x_{i-1}\rangle \,.\, \{x_{i-1}\} \times (\textbf{if } x_i = x_{i-1} & \\
\qquad \textbf{then } \rho^{2,3}_{x_i}(\mathrm{dom}(\mathbf{XKM}_i)) - \langle 1, 0\rangle & \\
\qquad \textbf{else } \mathrm{bmax}^{-1}(\rho^2_1(\rho^{2,3}_{x_i}(\mathrm{dom}(\mathbf{XKM}_{i-1}))))), & \\
\quad \lambda t \,.\, \mathrm{true}, \pi_1(\mathrm{dom}(\mathbf{XKM}_i)) \times \pi_1(\mathrm{dom}(\mathbf{XKM}_{i-1}))) & \\
X_i \text{ in } \pi_1(\mathrm{dom}(\mathbf{XKM}_i)) & (r_6)
\end{array}
$$

The bmax^{-1} operation may return an infinite set, and care must be taken when implementing the r_4 and r_5 pruning rules.

8 Experimental Protocol for the LONGESTPLATEAU Experiment

The random sampling of the domains of the X_i variables has been done in two different ways (5,000 instances of each). Let d be the maximum domain size. In the first way, a coin is thrown for each value between 1 and d to decide if it is part of the domain. In the second way, a domain size s is drawn at random between 1 and d, then random values between 1 and d are picked until there are s different values.

The random generation of the L variable draws a lower and an upper bound at random between 1 and $\max(5, n/2)$. Drawing up to n generates too many trivially unfeasible instances when n gets larger.

The reduction of the size of the search space reported in Table 1 in the main text is computed as follows. Let D^{init} be the list of initial domains (of the X_i and L), D^{best} be the list of domains that are obtained from D^{init} by global domain consistency (of the DP formulation of the constraint), and D^p be the list of domains obtained from D^{init} by application (until fix-point) of propagator p. The reduction $\text{red}(p)$ achieved by propagator p is computed as:

$$\text{red}(p) = \frac{\text{card}(D^{\text{init}}) - \text{card}(D^p)}{\text{card}(D^{\text{init}}) - \text{card}(D^{\text{best}})}$$

where $\text{card}(D)$ is the product of the size of the domains in list D. A value of 1 for $\text{red}(p)$ means that propagator p achieves all possible pruning (i.e., domain consistency), and a value of 0 that it achieves no pruning at all. To avoid division by 0, we discard instances where no pruning is possible (i.e., $\text{card}(D^{\text{init}}) = \text{card}(D^{\text{best}})$). The reported result for each propagator p is computed as the arithmetic average of $\text{red}(p)$ over 10,000 prunable instances.

References

[Katsirelos, Narodytska, and Walsh 2012] Katsirelos, G.; Narodytska, N.; and Walsh, T. 2012. The SeqBin constraint revisited. In Milano, M., ed., *CP 2012*, volume 7514 of *Lecture Notes in Computer Science*, 332–347. Springer.

[Pesant 2011] Pesant, G. 2011. Filtering and counting for the spread and deviation constraints. In *ModRef 2011*. `http://www.crt.umontreal.ca/~{}quosseca/\detail_publication.php?id=64`.

[Petit, Beldiceanu, and Lorca 2011] Petit, T.; Beldiceanu, N.; and Lorca, X. 2011. A generalized arc-consistency algorithm for a class of counting constraints. In Walsh, T., ed., *IJCAI 2011*, 643–648. IJCAI/AAAI. Revised edition available at `http://arxiv.org/abs/1110.4719`.