# Acquiring Maps of Interrelated Conjectures on Sharp Bounds

**Nicolas Beldiceanu**
IMT Atlantique, LS2N (TASC), Nantes, France

**Jovial Cheukam-Ngouonou**
IMT Atlantique, LS2N (TASC), Nantes, France, and Université Laval, Québec, Canada

**Rémi Douence**
IMT Atlantique, LS2N, Inria, (Gallinette), Nantes, France

**Ramiz Gindullin**
IMT Atlantique, LS2N (TASC), Nantes, France

**Claude-Guy Quimper**
Université Laval, Québec, Canada

──── **Abstract** ────

To automate the discovery of conjectures on combinatorial objects, we introduce the concept of a *map of sharp bounds* on characteristics of combinatorial objects, that provides a set of interrelated sharp bounds for these combinatorial objects. We then describe a Bound Seeker, a CP-based system, that gradually acquires maps of conjectures. The system was tested for searching conjectures on bounds on characteristics of digraphs: it constructs sixteen maps involving 431 conjectures on sharp lower and upper-bounds on eight digraph characteristics.

## 1 Introduction

Research on conjectures making systems in the context of discrete mathematics is a topic that goes back to the late 1950s and the 1980s [8, 14, 32] and got renewed interest [20, 21, 29, 31]. Within CP, some initial research on the generation of implied constraints was done by Charley *et al.* [11] and the most recent work focuses on model and constraint acquisition [4, 7, 10, 19, 27, 28] rather than on conjecture making. Within OR, Hansen's AutoGraphiX system [1, 17] focuses on finding unrelated bounds using Variable Neighbourhood Search.

Four reasons motivate our work: (i) to highlight that CP can contribute to the automatic discovery of conjectures, (ii) to systematically search sharp bounds on characteristics of objects that show up in combinatorial problems, (iii) to stress the need to develop conjecture discovery programs that build up a body of strongly interrelated knowledge rather than unrelated conjectures as it has been the case so far, (iv) by the fact that bounds are an essential feature of branch-and-bound methods in optimisation but also a weakness of CP [16, 22]: the development of sharp bounds that consider several interrelated characteristics is still a

manual process [3, 6]. Our approach is unique among all works for conjectures generation, as the result is not a set, but rather a *graph of conjectures*, linked by projection (i.e. variable elimination) operators. Our contributions are:

- We introduce the concept of *map of sharp bounds* as a set of interrelated conjectures providing sharp lower and upper-bounds wrt the characteristic of a combinatorial object.
- For each conjecture on a sharp bound, the map gives some *extremal characteristics* i.e., the characteristic values common to all combinatorial objects achieving the bound.
- By introducing secondary characteristics and by permitting the use of common sub-expressions in a polynomial, as well as simple Boolean and conditional formulae, we tend to produce *explainable conjectures*. This also reveals *unified conjectures* across different subsets of characteristics.
- We demonstrate the usefulness of CP for acquiring such maps: using digraphs as combinatorial objects, the system produces 431 conjectures distributed in 16 maps obtained from 8 characteristics combined with lower and upper bounds. It retrieves a set of known results, enhances some known bounds, and comes up with new conjectures, some of which we proved to be true.

The significance of maps is twofold. Beyond sharp bounds, a map brings together the relations between several sharp bounds and the structure of combinatorial objects reaching each bound under the same edifice. A map can be used to test the mutual consistency of independently acquired bounds by verifying that one bound can be derived from another bound.

In Sect. 2, we introduce the concept of a *map* that presents a set of conjectures for sharp bounds and their logical relations. In Sect. 3.1, we provide the workflow of our acquisition system. We introduce, in Sect. 3.2, a parameterised CP conjecture generator. We evaluate the produced conjectures in Sect. 4, discuss related work in Sect. 5, and conclude in Sect. 6.

## 2   Conjectures map as a symbolic piece of knowledge

After providing an informal overview of maps of conjectures, and a first example of a map, we motivate, define and illustrate the map concept. Then we show how the use of secondary characteristics permits both acquiring formulae sharing common sub-expressions, and sometimes come up with the same bound for different subsets of input characteristics.

**Informal overview of maps**   Consider digraphs as an example of combinatorial objects. It is well known that any digraph $\mathcal{G}$ satisfies the following invariant: the number of arcs $a$ of $\mathcal{G}$ is less than or equal to the square of the number of vertices $v^2$ of $\mathcal{G}$, and the maximum value $v^2$ is only reached when the number of vertices of the smallest connected component of $\mathcal{G}$ is equal to $v$, i.e. $\mathcal{G}$ consists of a single connected component of $v$ vertices.

We are interested in systematically generating such candidate invariants, a.k.a. conjectures, for a richer set of characteristics, e.g. the number of connected components $c$ of $\mathcal{G}$, the number $\underline{c}$ of vertices of the smallest connected component of $\mathcal{G}$.

Our conjectures have one of the following forms: ($i$) sharp bounds of a digraph characteristic wrt other digraph characteristics, e.g. $a \leq v^2$, or ($ii$) implication showing that, when a sharp bound is reached, some characteristics are fixed or functionally determined by some other characteristics, e.g. $a = v^2 \Rightarrow c = 1$, and $a = v^2 \Rightarrow \underline{c} = v$.

Finally, we are interested in connecting sharp bounds, revealing that the right-hand side of an implication of type ($ii$) can be used to eliminate a characteristic of a sharp bound and retrieve a sharp bound with one less characteristic. For instance, replacing $\underline{c}$ by $v$ in the

sharp bound $a \leq \underline{c}^2 + (v - \underline{c})^2$, we retrieve the sharp bound $a \leq v^2$. We call these different conjectures and the links connecting sharp bounds "map".

**A first example of map**    As an example of combinatorial objects, we use in this paper digraphs with these characteristics: the number $v$ of vertices, the number $a$ of arcs, the number $c$ (resp. $s$) of connected components (resp. strongly connected components), the number $\underline{c}$ (resp. $\bar{c}$) of vertices of the smallest (resp. largest) connected component, the number $\underline{s}$ (resp. $\bar{s}$) of vertices of the smallest (resp. largest) strongly connected component. To compare the bounds obtained by the Bound Seeker with the database of invariants of the global constraint catalogue, see Sect.4.3 of [2], we assume that each vertex of a digraph has at least one incoming or outgoing arc.

▶ **Example 1.** Fig. 1 illustrates the map concept with a map containing three conjectures labelled as ❶, ❷, and ③:

- Two conjectures about the sharp bounds ❶ $a \leq (v - (c-1))^2 + (c-1)$, and ❷ $a \leq v^2$ on the maximum number of arcs $a$ in a digraph $\mathcal{G}$ wrt the number of vertices $v$, and the number of connected components $c$ of $\mathcal{G}$.
- The conjecture ③ of node (B) indicates that the bound $v^2$ is reached only when $c = 1$.

The arrow going from node (A) to node (B) is labelled by ③ as the bound $v^2$ is obtained by replacing $c$ by 1 in the bound $(v - (c-1))^2 + (c-1)$. The leftmost and rightmost parts of Fig. 1 show, in brown, two digraphs achieving these bounds.
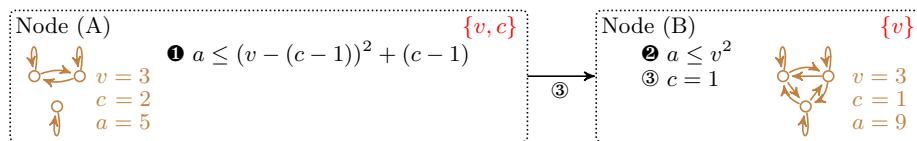


**Figure 1** Map of two sharp bounds on the maximum number of arcs of a digraph

In this paper, all maps of conjectures are presented in the same way as the map in Fig. 1: ($i$) the upper left corner of a node gives a *node label* in black, ($ii$) the upper right corner provides the *parameters* used in the sharp bound of this node in red, ($iii$) a dark label of the form ❶ refers to the *sharp bound* itself, ($iv$) a light label of the form ① designates an *equation* which must hold to reach the sharp bound given in ($iii$), ($v$) a brown illustration shows a *witness to the sharpness of the bound.* Finally, an arrow from a first node to a second node indicates which equation(s) in the second node should be used to substitute some parameters used in the first node's bound to retrieve the bound given in the second node. For space reasons, some large maps, e.g. Fig 4, may omit the elements ($i$) and ($v$).

**Motivating and defining the concept of map**    We introduce the concept of a *map of conjectures* as a way to reveal the links between a set of conjectures related to sharp bounds for a characteristic of a combinatorial object. Our goal is to describe conjectures on sharp bounds of characteristics of a combinatorial object, e.g. a digraph, a tree, and to organise these conjectures into a single structure, a *map of sharp bounds*, which (i) systematically interconnects these conjectures, and which (ii) describes the structure of the combinatorial objects for which the bounds are reached. In the map in Fig. 1, we consider for digraphs three characteristics, $a$, $v$ and $c$ for the number of arcs, of vertices, and of connected components.

▶ **Definition 2.** *Given a finite set of input characteristics $\mathcal{P}$ and an output characteristic $o \notin \mathcal{P}$, a* map of sharp upper bounds $\mathcal{M}_{\mathcal{P}}^{o \leq}$ *is defined as a digraph where:*

- *Each node of the map is associated with a subset $P \subseteq \mathcal{P}$ of input characteristics and corresponds to a* maximum conjecture *of the form $o \leq f(P)$. This inequality is tight, i.e. there exist values that can be given to the parameters $P$ in order to reach the equality. In addition, a node contains* maximality conjectures, *at most one per characteristic $q$ in the complement of $P$ wrt $\mathcal{P}$, represented by the symbolic equality $q = g_q(P)$, where $g_q$ is a function defined over realisable parameters values of $P$ and called a* maximum characterisation, *and expressing the following property: for any combination of parameters $P$ reaching the maximum $f(P)$, the characteristic $q$ is equal to $g_q(P)$.*
- *Each arc from conjecture $o \leq f_i(P_i)$ to conjecture $o \leq f_j(P_j)$ corresponds to a projection from a subset $P_i$ of input characteristics to a subset $P_j$ of input characteristics, by eliminating a characteristic $q_{i,j}$, i.e. $P_j = P_i \setminus \{q_{i,j}\}$. The arc is labelled with an equality $q_{i,j} = g_{q_{i,j}}(P_j)$ where $g_{q_{i,j}}(P_j)$ is the value given to $q_{i,j}$ to reach the equality in the conjecture $o \leq f_j(P_j)$. The equality $q_{i,j} = g_{q_{i,j}}(P_j)$ is called a* maximality conjecture.

In a map, there is a single output characteristic that we bound using the other characteristics called input characteristics. The output characteristic is the *bounded characteristic*, while the input characteristics are the *bounding characteristics*. While the maximum conjecture provides a bound on the output characteristic wrt the characteristics in $P$, the maximality conjectures indicate the values taken by the characteristics not in $P$ when the bound is reached. Similarly to $\mathcal{M}_{\mathcal{P}}^{o \leq}$, a map $\mathcal{M}_{\mathcal{P}}^{o \geq}$ provides a collection of sharp lower bounds as a set of minimum conjectures of the form $o \geq f_j(P_j)$, and a set of minimality conjectures.
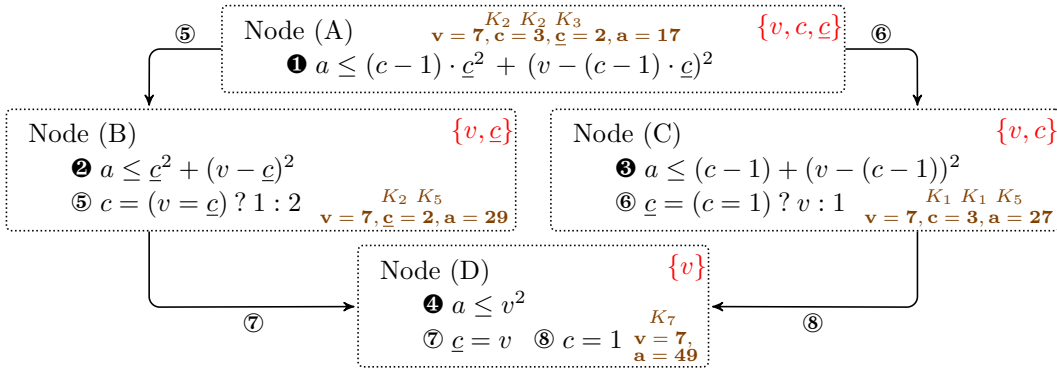


**Figure 2** Map $\mathcal{M}_{\{v,c,\underline{c}\}}^{a \leq}$ with the sharp upper-bounds ❶, ❷, ❸, ❹ for the number of arcs in a digraph; each node presents an example in brown: given a value for the characteristics attached to the node, a graph reaching the maximum is described, as a union of cliques $K_i$, with $i$ vertices, e.g. in node (B), given the assignments $v = 7$ and $\underline{c} = 2$, the digraph with 2 cliques $K_2$, $K_5$ reaches the maximum 29 for the number $a$ of arcs; $cond\,?\,x : y$ denotes $x$ if condition *cond* holds, $y$ otherwise.

▶ **Example 3** (Extending Ex. 1 to a map of four nodes). Fig. 2 presents Map $\mathcal{M}_{\{v,c,\underline{c}\}}^{a \leq}$, where we consider the following characteristics of digraphs: as input characteristics, the number $v$ of vertices, the number $c$ of connected components, and the number $\underline{c}$ of vertices of the smallest connected component; as output characteristic, the number $a$ of arcs. In Map $\mathcal{M}_{\{v,c,\underline{c}\}}^{a \leq}$, there are four nodes, corresponding to the subsets $\{v, c, \underline{c}\}$, $\{v, \underline{c}\}$, $\{v, c\}$ and $\{v\}$, shown in red, whereas the power set of $\{v, c, \underline{c}\}$ contains eight subsets. For the four other subsets, namely $\{c, \underline{c}\}$, $\{\underline{c}\}$, $\{c\}$ and $\emptyset$, no conjecture can be found, as the number of arcs is not upper

bounded wrt these characteristics. In the nodes (A), (B), (C) and (D), the items labelled with ❶, ❷, ❸ and ❹ indicate a maximum conjecture wrt the number $a$ of arcs, while the elements marked with ⑤, ⑥, ⑦ and ⑧ show maximality conjectures wrt $c$ and $\underline{c}$. For instance, in Node (B), the maximum conjecture ❷ $a \le \underline{c}^2 + (v - \underline{c})^2$ really means: among all digraphs with $v$ nodes and whose smallest component contains $\underline{c}$ nodes, the digraph with most arcs has exactly $\underline{c}^2 + (v - \underline{c})^2$ arcs. Each arc is labelled with a maximality conjecture giving the value of the characteristic that is eliminated. For instance, from Node (A) to Node (B), the characteristic $c$ that is eliminated from ❶ satisfies this maximality conjecture ⑤: when the maximum of number of arcs is reached, the value of $c$ is 1 if $v = \underline{c}$, 2 otherwise.

**Capturing more bounds with secondary characteristics**    As the number of input characteristics grows, the bound formulae can get rather complicated. Consequently, we introduce a set $\mathcal{A}$ of auxiliary characteristics to obtain simpler formulae. Examples of such auxiliary characteristics are, for instance, $(i)\, c_{>1}$, $(ii)\, s_{>1}$, and $(iii)\, c_{\in\{2,3\}}$ which correspond to $(i)$ the number of connected components with more than one vertex, $(ii)$ to the number of strongly connected components with more than one vertex, and $(iii)$ to the number of connected components with two or three vertices and for which all strongly connected components have only one vertex. Also initially introduced when searching for lower bounds on the number of arcs, such characteristics have proved useful for many other bounds. We introduce the notion of secondary characteristics of the node of a map, which will be illustrated in Ex. 5 and 6.

▶ **Definition 4.** *Given a node of a map that is associated to a subset $P \subseteq \mathcal{P}$ of input characteristics, to an output characteristics* o, *to a maximum conjecture of the form* o $\le f(P)$, *and a set of auxiliary characteristics $\mathcal{A}$, the* set of secondary characteristics *of the node is defined as the characteristics of the set $\mathcal{A} \cup (\mathcal{P} - P - \{o\})$ which are functionally determined by the set $P$ when* o $= f(P)$.

To test that a secondary characteristic is functionally determined by $P$, we check for each generated combination of values for $P$ that the value of the secondary characteristic is unique. This test is performed while generating our dataset used for acquiring conjectures.

To find bounds that exploit these secondary characteristics, we use a multi-level approach: $(i)$ first, we look for a formula for each secondary characteristic; $(ii)$ then we try to catch a sharp bound also considering the secondary characteristics for which we could find a formula. Both in $(i)$ and $(ii)$ a formula can either use input characteristics and secondary characteristics for which we already found a formula. As a result, we obtain formulae that are easier to interpret, as we can associate a straightforward meaning to the sub-terms that appear in a bound. Ex. 5 illustrates this point.

▶ **Example 5** (Bound expressed wrt several secondary characteristics). This example shows the only lower bound found by the Bound Seeker on the number of arcs $a$ of a digraph $\mathcal{G}$ wrt the size $\overline{c}$ of its largest connected component and the size $\underline{s}$ of its smallest strongly connected component. We have $\mathcal{P} = \{v, a, c, \underline{c}, \overline{c}, s, \underline{s}, \overline{s}\}$, the bound parameters $P = \{\overline{c}, \underline{s}\}$, the output characteristic o $= a$, and the auxiliary characteristics $\mathcal{A} = \{c_{>1}, s_{>1}\}$. All potential secondary characteristics $\mathcal{A} \cup (\mathcal{P} - P - \{o\}) = \{v, c, \underline{c}, s, \overline{s}, c_{>1}, s_{>1}\}$ are functionally determined by $\overline{c}$ and $\underline{s}$. The lower bound found by the Bound Seeker is $a \ge s_{>1} - c_{>1} + v$ with:

- $s_{>1} = \min(-\underline{s} + \overline{c} + 1, 2 \cdot (\underline{s} \ge 2))$,
- $c_{>1} = (\overline{c} = c\,?\,0 : c)$,           where  $c = 1 + (((\overline{c} - 2 \cdot \underline{s}) \le 0) \wedge ((\overline{c} \mod \underline{s}) \ge 1))$,
- $v\;\; = ((\overline{c} - \underline{c}) = 0\,?\,\overline{c} : \overline{c} + \underline{c})$,      where  $\underline{c} = ((2 \cdot \underline{s} - \overline{c}) \le 0\,?\,\overline{c} : \underline{s})$,

where a Boolean expression such as $(\underline{s} \ge 2)$ is used as an integer, i.e. either 0 for false or 1 for true. While the main formula $s_{>1} - c_{>1} + v$ is simple, it uses a secondary characteristic

$s_{>1}$ which is expressed directly wrt $\bar{c}$ and $\underline{s}$, and two other secondary characteristics $c_{>1}$ and $v$ which mention the two extra secondary characteristics $c$ and $\underline{c}$ for which two formulae involving only $\bar{c}$ and $\underline{s}$ could be found. The occurrence of Boolean expressions reflects slight variations in the structure of *witness digraphs*, i.e. digraphs reaching a sharp bound, as shown in Table 1.

| $\bar{c}$ | $\underline{s}$ | $a$ | $v$ | $c$ | $\underline{c}$ | $c_{>1}$ | $s_{>1}$ | witness digraph | $s_{>1} - c_{>1} + v$ |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 1 | 5 | 6 | 1 | 6 | 1 | 0 |  | $0 - 1 + 6$ |
| 6 | 3 | 7 | 6 | 1 | 6 | 1 | 2 |  | $2 - 1 + 6$ |
| 6 | 4 | 10 | 10 | 2 | 4 | 2 | 2 |  | $2 - 2 + 10$ |
| 6 | 6 | 6 | 6 | 1 | 6 | 1 | 1 |  | $1 - 1 + 6$ |

**Table 1** Digraphs minimising the number of arcs for four values of the bound parameters $\bar{c}$ and $\underline{s}$

Within a same map, expressing bounds in terms of secondary characteristics may reveal a same bound formula for several subsets of input characteristics. We observed this phenomenon in the majority of the acquired maps. Ex. 6 illustrates this for the acquired map giving the upper bound on the number of vertices of the largest connected component of a digraph.

▶ **Example 6** (Map example illustrating how bounds can be unified by using secondary characteristics). In the appendix, Fig. 4 depicts the maximum and maximality conjectures of the map $\mathcal{M}^{\bar{c} \leq}_{\{v,c,\underline{c},s,\underline{s},\overline{s}\}}$ found by the Bound Seeker for the upper-bound on the size of the largest connected component $\bar{c}$ with the related links. Note that $v$ needs to be an input characteristic, as otherwise the upper-bound of $\bar{c}$ is unbounded. Part (A) shows the 16 bounds found when using only the input characteristics: these bounds are defined by 5 maximum conjectures ❶,...,❺ and 4 maximality conjectures ⑥,...,⑨. Each link illustrates how a maximum conjecture is projected onto an other maximum conjecture via a maximality conjecture: e.g., the link ❺ $\xrightarrow{⑦}$ ❶ shows how the bound ❺ $\bar{c} \leq \underline{s} - c \cdot \underline{s} + v$ is rewritten as ❶ $\bar{c} \leq v$ as we have ⑦ $c = 1$. Part (B) shows the bounds found when also using the secondary characteristics $r$ and $\underline{c}$, where $r$ is a secondary characteristic corresponding to $v - c \cdot \underline{c}$. We only have 2 maximum conjectures ❶ $\bar{c} \leq v$ and ❷ $\bar{c} \leq r + \underline{c}$, where $r$ and $\underline{c}$ are defined by the 5 maximality conjectures ③,...,⑦ shown on Part (B). The natural upper-bound of $\bar{c}$ is the number of vertices of the digraph (see ❶), unless $c$ or $\underline{c}$ are part of the input characteristics (see ❷), which requires to consider the feasibility conditions induced by the use of such inputs.

Missing arcs are due to the lack of functional dependencies. For instance, in Part (A), we have no arc from $\{v,s\}$ to $\{v\}$, as the number of strongly connected components $s$ is not functionally determined by the number of vertices $v$ when the sharp bound ❶ is reached, i.e. when $\bar{c} = v$: e.g., for $\bar{c} = v = 2$ we both have $s = 2$ and $s = 1$ as shown by ●→● and ⟲.

## 3    A Bound Seeker

### 3.1    Overview of the map acquisition system

Parts (A) and (B) of Fig. 3 gives the different phases for generating a map: software components are shown in cyan and labelled with capital letters, while data is displayed in orange. We now detail the phases (A1), (A2), (A3), (B1), (B2), and (B3). To illustrate each phase, we use the bound table $\mathcal{T}^{a \leq}_{\{v,c\},3}$ provided in Part (C1) of Fig. 3.

**(A) Data acquisition for a map $\mathcal{M}^{o\leq}_{\mathcal{P}}$**

(A1) Data Generation (domain dependent)

Bound Tables ($\mathcal{T}^{o\leq}_{P,i}$, $P\subseteq\mathcal{P}$, $i\in[2,n]$): each column is an input, a bound, or a secondary characteristic for digraphs whose number of vertices does not exceed a given limit.

(A2) Metadata Generation

Metadata ($\mathcal{D}^{o\leq}_{P,i}$, $P\subseteq\mathcal{P}$, $i\in[2,n]$): columns information, row constraints, functional dependencies for each bound table.

(A3) Meta Metadata Generation

Selected Tables ($\mathcal{T}^{o\leq}_{P,k}$, $P\subseteq\mathcal{P}$): selected size for each subset of bound tables from which acquire conjectures.

**(B) Conjectures acquisition for each table $\mathcal{T}^{o\leq}_{P,k}$**

Metadata    Bound Tables

(B1) Candidate Generation

Candidate Formulae: select formula type, used functions, and used functional dependency to generate a maximum conjecture of the form $o\leq f(P)$ and maximality conjectures for the bound table $\mathcal{T}^{o\leq}_{P,k}$.

selected size $k$ / largest size $n$

(B2) CP Model Generation

Candidate Conjectures

(B3) Test Conjectures

Selected Conjectures

**(C) Bound table example**

(C1) bound table $\mathcal{T}^{a\leq}_{\{v,c\},3}$

| $v$ | $c$ | $a$ | $\overline{c}$ |
|-----|-----|-----|----------------|
| 1 | 1 | 1 | 1 |
| 2 | 1 | 4 | 2 |
| 2 | 2 | 2 | 1 |
| 3 | 1 | 9 | 3 |
| 3 | 2 | 5 | 2 |
| 3 | 3 | 3 | 1 |

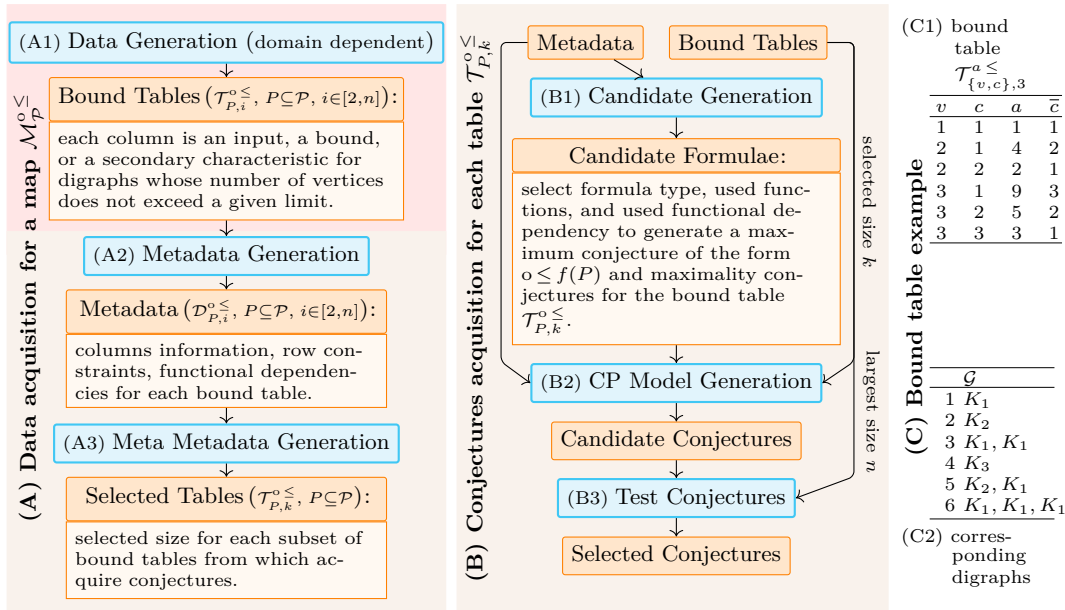| | $\mathcal{G}$ |
|---|---|
| 1 | $K_1$ |
| 2 | $K_2$ |
| 3 | $K_1, K_1$ |
| 4 | $K_3$ |
| 5 | $K_2, K_1$ |
| 6 | $K_1, K_1, K_1$ |

(C2) corresponding digraphs

■ **Figure 3** Workflow in the Bound Seeker: (A) data and (B) conjecture acquisition phases; Phase (A1) with a red background depends on the combinatorial objects we consider (digraphs in our case), while Phases (A2), (A3), . . . , (B3) are domain independent; (C1) example of an upper-bound table for digraphs of at most 3 vertices with the *input characteristic $v$, $c$*, the *output characteristics $a$*, and the *secondary characteristic $\overline{c}$* corresponding to the number of vertices of the largest connected component; (C2) digraphs corresponding to each entry of the bound table shown in (C1).

**(A1) Generating data** To learn valid conjectures for any digraph of at most $k$ vertices, we produce all parameter combinations of interest for digraphs up to a maximum number $n$ of vertices. An exhaustive generation of such data is not a problem, as a program is used for this purpose. However, the issue is to select the appropriate value of $k$, neither too small to create invalid conjectures for digraphs with more than $k$ vertices, nor too large to limit the number of generated constraints to acquire the conjectures in Phase (B2). To this end, Phase (A1) produces a table $\mathcal{T}$ with the characteristics values for digraphs of at most $n$ vertices in such a way that the size of the table $\mathcal{T}$ does not exceed a given memory limit. With this table $\mathcal{T}$, Phase (A1) extracts for each $i$ between 2 and $n$, for each subset of input characteristics $P$ of $\mathcal{P}$, and each output characteristic o, a bound table $\mathcal{T}^{o\leq}_{P,i}$ based only on the entries of $\mathcal{T}$ corresponding to digraphs with at most $i$ vertices. Each row of a bound table represents a feasible combination of values for $P$, with the corresponding bound value for o, and the values of the secondary characteristics.

Unlike all the next steps, Phase (A1) depends on the type of combinatorial objects for which we generate conjectures. For digraphs, our data generation phase uses a CP model to produce a set of bound tables that is used by the acquisition process. As illustrated in Part (C1) of Fig. 3, the bound table $\mathcal{T}^{a\leq}_{\{v,c\},3}$ provides a sharp upper bound of the output characteristic $a$ wrt the input characteristics $v$ and $c$. A bound table may also mention secondary characteristics, e.g. $\overline{c}$ in $\mathcal{T}^{a\leq}_{\{v,c\},3}$, which are functionally determined by the input characteristics. Each column of the table $\mathcal{T}^{a\leq}_{\{v,c\},3}$ refers to a characteristic, i.e. $v$, $c$, $a$, $\overline{c}$, while each row corresponds to a combination of parameter values for $v$, $c$ with the associated maximum number of arcs $a$ and the value of the secondary characteristic $\overline{c}$.

**(A2) Generating metadata** For each bound table $\mathcal{T}^{o\leq}_{P,i}$ (with $P \subseteq \mathcal{P}$ and $i \in [2,n]$),

with *nrows* rows, where $\mathcal{T}_{P,i}^{o\leq}[r,j]$ denotes the value of the $r$-th row and the $j$-th column, Phase (A2) calculates the aggregated information $\mathcal{D}_{P,i}^{o\leq}$ (with $P \subseteq \mathcal{P}$ and $i \in [2,n]$) used to select the size $k$ employed when searching for the conjectures of the subset $P$ and the output characteristics o, such as:

- The minimum/maximum values of each column and the number of distinct values.
- The minimal functional dependencies [24] that determine in the table $\mathcal{T}_{P,k}^{o\leq}$ the output characteristic and the secondary characteristics. Each functional dependency gives a subset of characteristics that functionally determine another characteristic. For instance, in the bound table $\mathcal{T}_{\{v,c\},3}^{a\leq}$, columns $a$ and $\bar{c}$ are functionally determined by columns $v$ and $c$. But column $a$ is also functionally determined by columns $v$ and $\bar{c}$.
- Binary constraints between two distinct columns $i$ and $j$ of the table $\mathcal{T}_{P,k}^{o\leq}$, i.e. constraints of the form $\forall r \in [1, nrows], \mathcal{T}_{P,k}^{o\leq}[r,i] \; op \; \mathcal{T}_{P,k}^{o\leq}[r,j]$ (with $op \in \{\leq, <, >, \geq\}$). In $\mathcal{T}_{\{v,c\},3}^{a\leq}$ we have for each row that the number of vertices is greater than or equal to the number of connected components, i.e. $v \geq c$, and similarly $v \geq \bar{c}$, $a \geq v$, $a \geq c$, $a \geq \bar{c}$.

Such knowledge is used to focus the search for conjectures: first by selecting promising subsets of input parameters for a formula, and second by providing information that avoids producing meaningless formulae. For instance, we do not generate a formula with a term $\min(v,c)$ as $v \geq c$ is true. The generated metadata is also the input of the next phase.

**(A3) Generating meta metadata to find the relevant size of the training dataset** Based on the information computed by Phase (A2), Phase (A3) determines for the subset $P$ and the output characteristic o, the size $k$ used when searching for conjectures. To select the size $k$ in the datasets $\mathcal{T}_{P,i}^{o\leq}$ (with $i \in [2,n]$) from which we acquire the conjectures, we operate as follows. As a functional dependency or a binary constraint of a table $\mathcal{T}_{P,i}^{o\leq}$ may become invalid for a table $\mathcal{T}_{P,j}^{o\leq}$ with $j > i$, we identify the smallest size $k$ from which the set of minimal functional dependencies and the set of binary constraints of the tables $\mathcal{T}_{P,k}^{o\leq}, \ldots, \mathcal{T}_{P,n}^{o\leq}$ remain identical. In practice, for space reason, we generated digraphs with up to $n = 26$ vertices. To avoid overfitting when the number of rows of table $\mathcal{T}_{P,k}^{o\leq}$ is too small, we select the smallest size corresponding to the table with at least 200 rows: on average, conjectures were produced using digraphs with up to 18 vertices.

**(B1) Generating candidate formulae** This phase generates for a bound table $\mathcal{T}_{P,k}^{o\leq}$, partially instantiated candidate formulae to acquire the corresponding maximal and maximality conjectures. Given the parameters $P$, the output characteristic o, the set of secondary characteristics of the selected bound table $\mathcal{T}_{P,k}^{o\leq}$, Phase (B1) produces on request the next candidate formula to find a conjecture. The set of potential characteristics that the formula may mention, and the formula itself, are restricted by the functional dependencies and the binary constraints that were identified by the metadata generation phase. Table 2 shows some candidates formulae that are successively produced for table $\mathcal{T}_{\{v,c\},3}^{a\leq}$.

**(B2) Generating a CP model linking a parameterised formula with the data** This phase uses a candidate formula generated by Phase (B1) to post an equational constraint for each entry in a bound table $\mathcal{T}_{P,k}^{o\leq}$ to obtain a formula where all input parameters and coefficients are fixed and thus produce a conjecture. Phase (B2) queries Phase (B1) for the next candidate parameterised formula, tries to instantiate it, and asks again for a next candidate formula. To find a value for each coefficient of a candidate formula, we use a constraint model to link a candidate formula to ($i$) the functional dependencies and binary constraints identified by the metadata generation phase, and ($ii$) all the bound table entries of the selected size. Many constraints break different symmetry types and force all sub-terms

| Candidate formulae generated by Phase (B1) | Formulae found by Phase (B2) |
| --- | --- |
| polynomial of degree 1 parameterised by $v$ and $c$ to determine $\bar{c}$ | $\bar{c} = v - c + 1$ |
| polynomial of degree 1 parameterised by $v$ and $c$ to determine $a$ | none |
| polynomial of degree 2 parameterised by $v$ and $c$ to determine $a$ | $a = c^2 - 2 \cdot v \cdot c + v^2 - c + 2 \cdot v$ |
| polynomial of degree 1 parameterised by $v$ and $\bar{c}$ to determine $a$ | none |
| polynomial of degree 2 parameterised by $v$ and $\bar{c}$ to determine $a$ | $a = \bar{c}^2 - \bar{c} + v$ |

**Table 2** Examples of candidates formulae and corresponding generated formulae for the bound table $\mathcal{T}^{a\,\leq}_{\{v,c\},3}$ in Part (C1) of Fig 3

of a formula to be meaningful. The second column of Table 2 shows for each candidate formula the corresponding concrete formula found by the CP model.

**(B3) Testing the candidate conjectures** This last phase tests the validity of the conjectures against the largest bound table $\mathcal{T}^{o\,\leq}_{P,n}$, i.e. against the largest available generated dataset.

## 3.2    A constraint approach for acquiring symbolic equations

The search for sharp bounds leads to the identification of equations in which the left-hand side is an output or a secondary characteristic, and the right-hand side is a formula involving input and secondary characteristics. As already noted in the introduction of [9] and in the conclusion of [18], the space of candidate formulae constitutes a major challenge for equation discovery methods. Rather than applying a bottom-up approach that generates formulae of increasing complexity, we adopt the following strategy. As we aim at finding simple formulae, we use three complementary classes of formulae that turned out to appear concomitantly in a map: (1) Boolean formulae involving $k$ arithmetic conditions linked by a single commutative logical operator or by a sum, (2) simple conditional formulae, and (3) formulae over polynomials that can share common sub-expressions. A first attempt to use only polynomials without common sub-expressions missed some formulae, e.g. see Ex. 5, and quite often provided too complicated formulae, as illustrated in Ex. 8. Based on the metadata introduced in Sect. 3.1, we will present a CP approach for restricting the space of formulae: for space reasons, we focus on polynomials sharing common sub-expressions.

### 3.2.1    A parameterised candidate formulae generator for Phase (B1)

**Formula syntax**    All conjectures we generate have the form *characteristic op formula*, where *op* is one of the comparison operators $\leq$, $=$, $\geq$, and *formula* is a formula involving a set of characteristics. Consequently, formulae are described by the following set of simplified grammar rules, where "Small Capitals" indicates a non-terminal symbol, "Roman" denotes a function or a known constant, "*Italic*" highlights a (digraph) characteristics, "**Bold**" denotes an unknown integer constant. Within these rules, polynomial(Params, degree) denotes a polynomial whose maximum degree is fixed (with degree $> 0$) on a non-empty subset of parameters of its potential parameters Params, and the functions geq0($x$), geq($x,y$), sum_consec($x$), cmod($x,y$), dmod($x,y$) resp. stand for 1 if $x \geq 0$ otherwise 0, 1 if $x \geq y$ otherwise 0, $\frac{x \cdot (x+1)}{2}$, $x - (y \bmod x)$, $x - (x \bmod y)$.

Formula    ::= **cst** | Bool | **cst** + Bool | Cond | Pol | PolBinary | PolUnary
Bool        ::= BoolOp(BoolConds)     BoolOp ::= $\wedge$ | $\vee$ | = | +
BoolConds ::= BoolCond, BoolConds | BoolCond
BoolCond  ::= Param Cmp **cst**     Cmp ::= $\leq$ | = | $\geq$ | $\neq$

$$
\begin{array}{ll}
\text{COND} & ::= (\text{BOOLCOND ? PARAMCST : PARAMCST}) \quad \text{PARAMCST} ::= \text{PARAM}|\textbf{cst} \\
\text{POL} & ::= \text{polynomial}(\text{PARAMS}, \text{degree}) \\
\text{POLBINARY} & ::= \text{BF}(\text{POL}, \text{POL}) \quad \text{BF} ::= \min|\max|\text{floor}|\text{mod}|\text{cmod}|\text{dmod}|\text{prod} \\
\text{POLUNARY} & ::= \text{UF1}(\text{POL}) \mid \text{UF2}(\text{POL}, \textbf{cst}) \\
\text{UF1} & ::= \text{geq0}|\text{sum\_consec} \quad \text{UF2} ::= \min|\max|\text{floor}|\text{mod}|\text{power} \\
\text{PARAMS} & ::= \text{PARAM}^* \quad \text{PARAM} ::= \text{CHAR}|\text{BTERM}|\text{UTERM} \quad \text{CHAR} ::= v|c|\underline{c}|\overline{c}|s|\underline{s}|\overline{s} \\
\text{BTERM} & ::= \text{BT}(\text{CHAR}, \text{CHAR}) \\
\text{UTERM} & ::= \text{sum\_consec}(\text{CHAR}) \mid \text{UT}(\text{CHAR}, \textbf{cst}) \mid \text{CHAR} \in [\textbf{cst}, \textbf{cst}] \\
\text{BT} & ::= \min|\max|\text{floor}|\text{ceil}|\text{mod}|\text{cmod}|\text{dmod}|\text{prod} \\
\text{UT} & ::= \min|\max|\text{floor}|\text{ceil}|\text{mod}|\text{geq}|\text{power}
\end{array}
$$

▶ **Example 7** (Examples of generated Boolean, conditional, polynomial formulae)**.**

- $(\overline{s} = 1) \wedge (\overline{c} \in [2,3])$ and $(v = \underline{c}) = (c = 1)$, where the 2nd formula denotes a condition that is satisfied only if both conditions $(v = \underline{c})$ and $(c = 1)$ are true, or both false.
- $(\underline{s} = 1 \; ? \; \lceil \frac{v}{2} \rceil : v)$ and $((\overline{c} - \underline{c}) = 0 \; ? \; \underline{c} : \underline{c} + \overline{c})$, where $(cond \, ? \, x : y)$ denotes $x$ if the condition $cond$ holds, $y$ otherwise.
- $(v \bmod \overline{c})^2 - \overline{c} \cdot (v \bmod \overline{c}) + v \cdot \overline{c}$ where $v \bmod \overline{c}$ is a shared binary term BTERM, $\lfloor \frac{(\overline{s} \geq 2) + \overline{s} + v}{2} \rfloor$ where $(\overline{s} \geq 2)$ is a unary term UTERM of the form geq$(\overline{s}, 2)$.

▶ **Example 8** (Finding simpler bounds using Boolean and conditional formulae)**.** We illustrate with an example generated by the system on the lower bound of the number of arcs $a$ wrt the size of the smallest and largest connected components $\underline{c}$ and $\overline{c}$, and the size $\overline{s}$ of the largest strongly connected component, how using Boolean and conditional formulae often leads to simpler conjectures. Without using Boolean and conditionals, we get $a \geq s_{>1} - c_{>1} + v$ with $s_{>1} = \min(\overline{s} - 1, 1)$, $c_{>1} = \min(\min(\underline{c}, 2), \min(\underline{c}, 2) + \overline{c} - \underline{c} - 1)$, and $v = \min(\overline{c} + \underline{c}, \underline{c} \cdot \overline{c} - \underline{c}^2 + \overline{c})$; enabling Boolean and conditional formulae, we get the simpler bound: $a \geq s_{>1} - c_{>1} + v$ with $s_{>1} = (\overline{s} \geq 2)$, $c_{>1} = (\underline{c} \geq 2) + ((\overline{c} - \underline{c}) \geq 1)$, and $v = ((\overline{c} - \underline{c}) = 0 \; ? \; \underline{c} : \underline{c} + \overline{c})$.

**Candidate formulae generator** Since we want to try out a variety of formulae, we create a parameterised candidate formulae generator, which, upon backtracking, proposes a new candidate formula with non-fixed coefficients; these are variables for the constants and for the input characteristics that will be used in a candidate formula. In this generator we specify:

- The structure of the formula, that is whether we use (1) a Boolean formula, (2) a simple conditional formula, or (3) a formula over polynomials; in this later case we also specify how many unary and binary terms occur in each polynomial.
- The arithmetic functions we may use in the terms.
- The complexity of a polynomial, that is its potential maximum degree, its maximum number of non-zero coefficients, the ranges of its coefficients.
- The list of possible combinations of characteristics that the candidate formula can use in its parameters. Such combinations correspond to functional dependencies identified by the metadata generation phase, i.e. Phase (A2).

We use more than one generator to design a formula generation policy where the simplest candidate formulae are tried first.

### 3.2.2 Constraint model for acquiring a conjecture for formulae over polynomials for Phase (B2)

Given a candidate formula $\mathcal{F}$, (corresponding either to POL, to POLBINARY, or to POLUNARY as described in the set of grammar rules in Sect. 3.2.1), for which the set of used parameters

is partially determined, and for which the coefficients are not yet fixed, we create a constraint model that relates these unknowns to all rows in a bound table. Our model includes four types of constraints, namely (i) structural constraints on the input and secondary characteristics that will be used in $\mathcal{F}$, (ii) symmetry-breaking constraints, (iii) constraints preventing the generation of formulae in which a term could be simplified, and (iv) equational constraints on each row of a bound table. We describe the model variables, the constraints on the characteristics used in $\mathcal{F}$, the constraints on the unary/binary terms and binary function of $\mathcal{F}$, and the equational constraints on the table entries. The number of variables and constraints of the model is linear wrt the number of table entries as it is dominated by the equational constraints. For reasons of space, concerning the constraints of the type (ii) and (iii), we will only detail the constraints related to the min function.

**Variables used in the model**   Table 3 introduces the variables used to represent a non-constant formula $\mathcal{F}$ involving at most $n_c$ characteristics (i.e. input and secondary characteristics), $n_u$ unary terms, $n_b$ binary terms, and $n_p$ polynomials, wrt a bound table $\mathcal{T}$ of *nrows* rows. We use $n$ as a shortcut for $n_c + n_u + n_b$. For the binary term $\mathcal{B}_i$, the variables $B\_IND1_i$, $B\_IND2_i$, $B\_O_i$ designate a term with the arguments $(C_{B\_IND1_i}, C_{B\_IND2_i})$ when $B\_O_i = 0$, and $(C_{B\_IND2_i}, C_{B\_IND1_i})$ otherwise. When the binary term is commutative, e.g. min, the order of the arguments is irrelevant and $B\_O_i$ will be set to 0 (see constraint (4.c) in Table 4), but otherwise, e.g. mod, the order matters.

| Objects | Variables | Comments |
|---|---|---|
| Characteristics $\mathcal{C}_j$ $(j \in [1, n_c])$ | $C_j \in \{0,1\}$ | $C_j = 1$ iff $\mathcal{C}_j$ used by formula $\mathcal{F}$ |
| Unary term $\mathcal{U}_i$ $(i \in [1, n_u])$ | $U_{i,j} \in \{0,1\}$ $(j \in [1, n_c])$ | $U_{i,j} = 1$ iff $\mathcal{C}_j$ used by $\mathcal{U}_i$ |
| | $U\_IND_i \in [1, n_c]$ | index of the used characteristics |
| | $U\_MIN_i$ | minimum value of the used characteristics |
| | $U\_MAX_i$ | maximum value of the used characteristics |
| | $U\_CST_i$ | constant used in $\mathcal{U}_i$ |
| $(r \in [1, nrows])$ | $U\_VAL_{i,r}$ | value of term $\mathcal{U}_i$ wrt $r$-th row and the $j$-th column (with $U_{i,j} = 1$ ) of table $\mathcal{T}$ |
| Binary term $\mathcal{B}_i$ $(i \in [1, n_b])$ | $B_{i,j} \in \{0,1\}$ $(j \in [1, n_c])$ | $B_{i,j} = 1$ iff $\mathcal{C}_j$ used by $\mathcal{B}_i$ |
| | $B\_IND1_i \in [1, n_c]$ | index of first used characteristic |
| | $B\_IND2_i \in [1, n_c]$ | index of second used characteristic |
| | $B\_O_i \in \{0,1\}$ | order of used characteristics in arguments |
| $(r \in [1, nrows])$ | $B\_VAL_{i,r}$ | value of term $\mathcal{B}_i$ wrt $r$-th row, the $B\_IND1_i$-th, and the $B\_IND2_i$-th columns of table $\mathcal{T}$ |
| Polynomial $\mathcal{P}_i$ of degree $d_i$ | $P_{i,j} \in \{0,1\}$ with $j \in [1, n]$ and $n = n_c + n_u + n_b$ | $\begin{cases} P_{i,j} = 1, j \in [1, n_c] & \Rightarrow \mathcal{C}_j & \text{used by } \mathcal{P}_i \\ P_{i,j} = 1, j \in [n_c + 1, n_{cu}] \Rightarrow \mathcal{U}_{j-n_c} & \text{used by } \mathcal{P}_i \\ P_{i,j} = 1, j \in [n_{cu} + 1, n] \Rightarrow \mathcal{B}_{j-n_c-n_u} \text{used by } \mathcal{P}_i \end{cases}$ |
| $(i \in [1, n_p])$ | $M_{i,k}$ $(k \in [1, \binom{n+d_i}{d_i}])$ | $M_{i,k}$ is the $k$-th coefficient of $\mathcal{P}_i$, the coefficient with the largest $k$ is the constant |
| $(r \in [1, nrows])$ | $P\_VAL_{i,r}$ | value of polynomial $\mathcal{P}_i$ wrt $r$-th row of table $\mathcal{T}$ |

**Table 3** Variables of the model, where $n_{cu}$ is an abbreviation of the term $n_c + n_u$.

**Constraints on the structure of the formula**   The upper part of Table 4 lists the con-

straints, (i) specifying which characteristics the formula $\mathcal{F}$ uses, i.e. see (1a), (ii) forcing a unary term, a binary term, and a polynomial to use the appropriate number of characteristics, i.e. see (2a), (3a) and (4a), (iii) connecting the characteristics used by the unary and binary terms with the characteristics used in the polynomials and the formula, i.e. see (5a), (6a), (iv) restricting non-zero coefficients of polynomials, i.e. see (7a), (8a).

**Constraints on unary/binary terms and on a binary function**   Within Table 4, constraint (1b) (resp. (1c), (2c)), links the 0-1 variables $U_{i,j}$ (resp. $B_{i,j}$) to the index of the characteristic involved in the term. To avoid generating unary terms of the form $\min(Characteristic, Cst)$ which could just be rewritten as $Characteristic$ or as $Cst$, constraint (4b) restricts the minimum and maximum values of the constant. When using the min function in a binary term, constraint (4c) avoids generating equivalent binary terms whose arguments are permuted. Constraint (5c) prevents generating a binary term when the min could be simplified, e.g. avoids generating $\min(\underline{c}, \bar{c})$ as the metadata information found in Phase (A2) indicates that $\underline{c}$ is always smaller than or equal to $\bar{c}$. Finally, when the candidate formula $\mathcal{F}$ is a binary function corresponding to min, that uses the polynomials $\mathcal{P}_1$ and $\mathcal{P}_2$ of degree $d$, we post the lexicographic ordering constraint $\langle M_{1,1}, \ldots, M_{1,\binom{n+d}{d}} \rangle <_{\text{LEX}} \langle M_{2,1}, \ldots, M_{2,\binom{n+d}{d}} \rangle$ between the monomial coefficients of $\mathcal{P}_1$ and $\mathcal{P}_2$. Note that, for space reason, besides constraints (4b), (4c), and (5c), we omit in Table 4 the symmetry and simplification constraints related to functions that are different from min.

**Equational constraints**   For each row $r$ of the bound table $\mathcal{T}$ we post some constraints linking the selected characteristics $\mathcal{C}_j$ with (i) the value variable $U\_VAL_{i,r}$ of each unary term $\mathcal{U}_i$, (ii) the value variable $B\_VAL_{i,r}$ of each binary term $\mathcal{B}_i$, and (iii) the value variable $P\_VAL_{i,r}$ of each polynomial $\mathcal{P}_i$. For each row $r$ we also post an equality constraint linking the value of the candidate formula $\mathcal{F}$ on row $r$ with the corresponding bound value on the same row. Finally, for a binary function min between two polynomial $\mathcal{P}_1$ and $\mathcal{P}_2$, we impose that for at least one of the entries of the bound table the value of $\mathcal{P}_1$ is strictly less than the value of $\mathcal{P}_2$ on the same entry, and that the converse applies for another entry of the table. To avoid unnecessarily complex formulae, we minimise the sum of the absolute values of the coefficients of a candidate formula $\mathcal{F}$.

## 4   Evaluation of the Bound Seeker

We focus on constructing 16 maps on the lower and upper-bounds of the number of vertices, the number of arcs, the number of connected (resp. strongly connected) components, and their minimum and maximum sizes. The components of the system are written in SICStus Prolog and consist of 10000 lines of code for the Data Generation, the Metadata Generation, the Meta Metadata Generation, the Candidate Formulae Generation, the CP Model Generation, and the Test phase. The Data Generation phase generates a total of 1944 bound tables (occupying 2 Gb) for each maximum number of digraph vertices ranging from 2 to 26; each bound table gives the lower or upper-bound of a characteristic wrt different subsets of input characteristics. We evaluate the Bound Seeker from several standpoints:

- The percentage of conjectures that, while acquired from the size selected by the Meta Metadata, still hold for all entries of the largest generated bound tables, i.e. the tables of digraphs containing up to 26 vertices.
- The percentage of bounds from the database of invariants in [2] that was retrieved (resp. not found).

| Constraints | Comments |
|---|---|
| (1a) $\text{TABLE}(\langle C_1, \ldots, C_c \rangle, \text{FD\_TABLE})$ | restrict the char. used in $\mathcal{F}$ |
| (2a) $\forall i \in [1, n_u] : \sum_{j=1}^{j=n_c} U_{i,j} = 1$ | $\mathcal{U}_i$ uses 1 char. |
| (3a) $\forall i \in [1, n_b] : \sum_{j=1}^{j=n_c} B_{i,j} = 2$ | $\mathcal{B}_i$ uses 2 char. |
| (4a) $\forall i \in [1, n_p] : \sum_{j \in [1,n]} P_{i,j} \geq 1$ | $\mathcal{P}_i$ uses at least one char., or at least one unary or binary term |
| (5a) $\forall j \in [1, n_c] : \ C_j = \bigvee_{i \in [1,n_u]} U_{i,j} \vee$ $\bigvee_{i \in [1,n_b]} B_{i,j} \vee \bigvee_{i \in [1,n_p]} P_{i,j}$ | link $U_{i,j}$, $B_{i,j}$, and $P_{i,j}$ to $C_j$ |
| (6a) $\forall j \in [n_c + 1, n] : \sum_{i \in [1,n_p]} P_{i,j} > 0$ | force each unary/binary term to be used by at least 1 polynomial |
| (7a) $\forall i \in [1, n_p] : (\sum_{k=1}^{k < \binom{n+d_i}{d_i}} [M_{i,k} \neq 0]) > 0$ | polynomials are not constant |
| (8a) $\forall i \in [1, n_p] : (\sum_{k=1}^{k \leq \binom{n+d_i}{d_i}} [M_{i,k} \neq 0]) \leq maxz$ | each polynomial has a maximum number of non-zeros coefficients |
| (1b) $\text{ELEMENT}(U\_IND_i, \langle U_{i,1}, \ldots, U_{i,n_c} \rangle, 1)$ | get index of used char. |
| (2b) $\text{ELEMENT}(U\_IND_i, \langle min_1, \ldots, min_{n_c} \rangle, U\_MIN_i)$ | get min. value of used char. |
| (3b) $\text{ELEMENT}(U\_IND_i, \langle max_1, \ldots, max_{n_c} \rangle, U\_MAX_i)$ | get max. value of used char. |
| (4b) $u_{f_i} \in \{min\} \Rightarrow \begin{cases} U\_CST_i > U\_MIN_i \\ U\_CST_i < U\_MAX_i \end{cases}$ | cannot simplify unary term $\mathcal{U}_i$, as otherwise could remove $\mathcal{U}_i$ |
| (1c) $\text{ELEMENT}(B\_IND1_i, \langle B_{i,1}, \ldots, B_{i,n_c} \rangle, 1)$ | get index of first used char. |
| (2c) $\text{ELEMENT}(B\_IND2_i, \langle B_{i,1}, \ldots, B_{i,n_c} \rangle, 1)$ | get index of second used char. |
| (3c) $B\_IND1_i < B\_IND2_i$ | indexes are ordered |
| (4c) $b_{f_i} \in \{min\} \Rightarrow B\_O_i = 0$ | fix order of the 2 arguments as $min$ is a commutative function |
| (5c) $b_{f_i} \in \{min\} \Rightarrow \text{TABLE}\left( \begin{array}{c} \langle B\_IND1_i, B\_IND2_i \rangle, \\ \text{TABLE\_UNORDERED} \end{array} \right)$ | assign two char. whose values are not ordered |

■ **Table 4** (**Top**) Constraints on the structure of a formula $\mathcal{F}$; FD\_TABLE is the list of characteristics combinations that may be used by $\mathcal{F}$, created by the candidate formulae generator, while $maxz$ is the maximum number of non-zero coefficients of a polynomial. (**Mid**) Constraints on a unary term $\mathcal{U}_i$ (with $i \in [1, n_u]$), where $u_{f_i}$ is the function assigned to $\mathcal{U}_i$, $min_j$ (with $j \in [1, n_c]$), is the smallest value of the $j$-th characteristic. (**Bottom**) Constraints on a binary term $\mathcal{B}_i$ (with $i \in [1, n_b]$), where $b_{f_i}$ is the function assigned to $\mathcal{B}_i$, and TABLE\_UNORDERED is the set of pairs of characteristics indices such that the 1st characteristic is not always smaller, or greater, than 2nd characteristic; char. is an abbreviation for characteristic.

▬ Besides the conjectures retrieved from the global constraint catalogue database, we manually proved ten new conjectures. Using WolframAlpha, we also checked the consistency of 105 projections of a sharp bound $B_1$ onto a sharp bound $B_0$ involving one less input characteristic, by substituting in $B_1$ the input characteristic to be eliminated, by the expression defined by the corresponding maximality conjecture.

As the complexity of a formula increases with the number of input characteristics, we limit our evaluation to up to 3 input characteristics. All experiments to acquire the conjectures for the 16 maps were done using the same system parameters, i.e. none of the components have been tuned manually to behave differently depending on the considered map. Out of 350 (resp. 202) combinations of input characteristics for which the Bound Seeker tried to find a sharp lower (resp. upper) bound, using only polynomials, it got at least one sharp bound for 279 (resp. 149) combinations of characteristics, as well as 1236 (resp. 975) minimality (resp. maximality) conjectures. Using also Boolean and conditional expressions it found 3 extra lower bounds and 93 new maximality/minimality conjectures. Table 5 provides the

| Maps | only Poly | | Bool/Cond/Poly | | | | Maps | only Poly | | Bool/Cond/Poly | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #P1 | Time | #B2 | #C2 | #P2 | Time | | #P1 | Time | #B2 | #C2 | #P2 | Time |
| $\mathcal{M}_\mathcal{P}^{c\geq}$ | 259 | 257 | 47 | 25 | 194 | 533 | $\mathcal{M}_\mathcal{P}^{c\leq}$ | 100 | 218 | 9 | 17 | 77 | 439 |
| $\mathcal{M}_\mathcal{P}^{\bar{c}\geq}$ | 129 | 230 | 0 | 13 | 120 | 476 | $\mathcal{M}_\mathcal{P}^{\bar{c}\leq}$ | 153 | 193 | 32 | 31 | 94 | 542 |
| $\mathcal{M}_\mathcal{P}^{\bar{s}\geq}$ | 97 | 130 | 0 | 8 | 90 | 306 | $\mathcal{M}_\mathcal{P}^{\bar{s}\leq}$ | 102 | 392 | 15 | 31 | 60 | 999 |
| $\mathcal{M}_\mathcal{P}^{a\geq}$ | 367 | 1248 | 38 | 102 | 255 | 3180 | $\mathcal{M}_\mathcal{P}^{a\leq}$ | 384 | 2505 | 46 | 84 | 264 | 5939 |
| $\mathcal{M}_\mathcal{P}^{\underline{c}\geq}$ | 63 | 167 | 10 | 27 | 27 | 388 | $\mathcal{M}_\mathcal{P}^{\underline{c}\leq}$ | 130 | 223 | 16 | 14 | 102 | 457 |
| $\mathcal{M}_\mathcal{P}^{\underline{s}\geq}$ | 43 | 54 | 0 | 18 | 25 | 226 | $\mathcal{M}_\mathcal{P}^{\underline{s}\leq}$ | 48 | 171 | 1 | 8 | 40 | 365 |
| $\mathcal{M}_\mathcal{P}^{s\geq}$ | 263 | 474 | 37 | 31 | 198 | 813 | $\mathcal{M}_\mathcal{P}^{s\leq}$ | 93 | 100 | 5 | 17 | 73 | 267 |
| $\mathcal{M}_\mathcal{P}^{v\geq}$ | 294 | 368 | 30 | 75 | 205 | 1570 | $\mathcal{M}_\mathcal{P}^{v\leq}$ | 14 | 7 | 0 | 2 | 12 | 90 |

**Table 5** Number of minimum/maximum and minimality/maximality conjectures found for each of the 16 maps and time in min. using only polynomials (see only Poly), and using Booleans, conditionals and polynomials (see Bool/Cond/Poly)

| Number of input characteristics | 1 | 2 | 3 | **Total** | **Percentage** |
|---|---|---|---|---|---|
| Number of equivalent sharp bounds retrieved by BS | 22 | 14 | 4 | 40 | **66,66%** |
| Number of sharper bounds than the GCC found by BS | 1 | 3 | 0 | 4 | **6,66%** |
| Number of generalised sharp bounds found by BS | 0 | 6 | 0 | 6 | **10%** |
| Number of erroneous bounds found in the GCC by BS | 1 | 1 | 1 | 3 | **5%** |
| Number of bounds in the GCC not retrieved by BS | 0 | 0 | 7 | 7 | **11,66%** |
| Total bounds of the GCC per column | 24 | 24 | 12 | 60 | |

**Table 6** Comparing the conjectures on the bounds found by the Bound Seeker (BS) with the database of invariants of the global constraint catalogue (GCC)

results for the 16 maps using SICStus 4.6.0 on a 2015 iMac with a 4 GHz Core i7 and 32Gb of memory: for each map, we give the number of formulae found using only polynomials (see col. #P1), then using Boolean, conditional, and polynomial (see col. #B2, #C2 and #P2). Using Boolean and conditional expressions generates 3.8% new formulae compared to when using polynomials alone; moreover, 31.07% of the formulae that use polynomials are replaced by simpler formulae that use Boolean or conditionals expressions. The time spent is explained by a significant number of candidate formulae tested, as it comes from the combination of minimal functional dependencies and grammar rules. Moreover, arithmetic constraints like div and mod with multiple occurrences of the same variable are handled poorly by CP solvers. The datasets used in the experiments and the sixteen maps found will be available for download in a technical report.

**Evaluation of the acquired conjectures wrt the largest data sets**  Of the 3625 conjectures acquired when only using polynomials, we found 5 invalid conjectures when tested against all samples of the largest data set, i.e. all digraphs up to 26 vertices. Of the 3264 conjectures acquired when also using Boolean and conditional expressions, we found 16 invalid conjectures. Note that in this setting the Bound Seeker does not try to find polynomial formulae if it already found a Boolean or a conditional formula.

**Comparing the conjectures founds with proved bounds of the constraint catalogue**  As shown in Table 6, the Bound Seeker retrieves 66.66% of the bounds of the constraint catalogue, even if the resulting formulae have sometimes a different form: e.g., the upper-bound on the

number of arcs $a$ wrt the number of vertices $v$, connected components $c$, and strongly connected components $s$ in the catalogue is expressed as $a \leq c-1+(v-s+1)\cdot(v-c+1)+\lfloor\frac{(s-c+1)\cdot(s-c)}{2}\rfloor$, while the Bound Seeker finds the equivalent inequality $a \leq \lfloor\frac{r^2+\overline{s}^2+v\cdot\underline{c}+r-\overline{s}+v}{2}\rfloor$, with $\underline{c} = \max(2\cdot v-v\cdot c,1)$, $\overline{s} = v-s+1$ and $r = v\cdot[c \geq 2]-c\cdot[c \geq 2]$; $r$ is a secondary characteristic corresponding to $v-c\cdot\underline{c}$. Unlike the bound given by [2], the bound found by the Bound Seeker defines the size $\underline{c}$ of the smallest connected component, and the size $\overline{s}$ of the largest strongly connected component of those extreme digraphs for which the upper-bound is reached.

An example of a generalised bound found by the Bound Seeker is the lower bound $a \geq ((v-\overline{c}) \leq 1\,?\,\max(v-1,1)\,:\,v-2)$, with $v = (\underline{c} = \overline{c}\,?\,\underline{c}\,:\,\underline{c}+\overline{c})$ which extends the catalogue bound $\underline{c} \neq \overline{c} \Rightarrow a \geq \underline{c}+\overline{c}-2+(\underline{c} = 1)$. An example of correct bound found by the Bound Seeker replacing the erroneous bound $(i)$ $a \geq v-\lfloor\frac{s-1}{2}\rfloor$ of the catalogue is $(ii)$ $a \geq v-c_{\in\{2,3\}}$ with $c_{\in\{2,3\}} = (v = s\,?\,\lfloor\frac{v}{2}\rfloor\,:\,\lfloor\frac{s-1}{2}\rfloor)$: for the edge condition $v = s = 2$, $(i)$ returns 2, rather than 1 as $(ii)$ does. Bound $(ii)$ $a \geq v-c_{\in\{2,3\}}$ can be interpreted as follows: to minimise the number of arcs, one has to maximise the number of connected components of the form •→•, •→•→•, •→•←• or •←•→• . The missing bounds of the catalogue are partially explained by the limited complexity of the common subexpressions (see BTerm, UTerm in Sect. 3.2.1) of our polynomials, and by the lack of some secondary characteristics.

## 5    Related work

While there exist several discovery programs in the context of mathematics devoted to set theory, number theory, finite algebra and knot theory [12, 23, 13], only a few systems focus on finding bounds between characteristics of a combinatorial object. The two most notable systems are S. Fajtlowicz's Graffiti program [14] and P. Hansen's AutoGraphiX system [1, 17]. The first difference is that the Bound Seeker attempts to systematically construct a set of sharp bounds on all possible combinations of a set of input characteristics. The second main difference is that the Bound Seeker introduces secondary characteristics and searches for key properties of extreme combinatorial objects for which the bounds are reached.

In slightly different domains, recent work in CP uses machine learning techniques to estimate the domain boundaries of an objective function [30] of an optimisation problem. Some other work uses CP to extract equations from a spreadsheet [18, 25], and some recent work investigates how to integrate integer programming solvers within neural networks [15, 26].

The specificity of our approach compared to machine learning and constraint acquisition [7] is twofold: $(i)$ we can generate our input data, but we need to ensure that these data contain the correct values of the sharp bounds we consider, as otherwise, we would necessarily obtain wrong maximal conjectures; moreover, maximality conjectures only make sense for sharp bounds; $(ii)$ we have to learn concise conjectures that fit perfectly to all available data, as minimising an error measure would be irrelevant for acquiring conjectures on sharp bounds.

## 6    Conclusion

We introduce a structure that connects a set of sharp bounds. Based on this structure, we propose a constructive approach to acquire a set of interrelated conjectures on sharp bounds. We show the relevance of using a variety of types of formulae, i.e., Boolean, conditionals, and polynomials with shared sub-expressions, to acquire simpler conjectures. This work opens a new application domain for CP for automated conjectures-making systems. It creates a new line of research to those already reported in a recent survey on machine learning for combinatorial optimisation [5].

────── **References** ──────

1   Mustapha Aouchiche, Gilles Caporossi, Pierre Hansen, and M. Laffay. Autographix: a survey. *Electron. Notes Discret. Math.*, 22:515–520, 2005. `doi:10.1016/j.endm.2005.06.090`.

2   Nicolas Beldiceanu, Mats Carlsson, and Jean-Xavier Rampon. Global Constraint Catalog, 2nd Edition (revision a). Technical Report T2012-03, Swedish Institute of Computer Science, 2012. Available at `http://ri.diva-portal.org/smash/get/diva2:1043063/FULLTEXT01.pdf`.

3   Nicolas Beldiceanu, Mats Carlsson, Jean-Xavier Rampon, and Charlotte Truchet. Graph invariants as necessary conditions for global constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2005.

4   Nicolas Beldiceanu and Helmut Simonis. A Model Seeker: Extracting Global Constraint Models from Positive Examples. In Michela Milano, editor, *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, volume 7514 of *Lecture Notes in Computer Science*, pages 141–157. Springer, 2012. `doi:10.1007/978-3-642-33558-7\_13`.

5   Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2):405–421, 2021.

6   Christian Bessière, Emmanuel Hebrard, George Katsirelos, Zeynep Kızıltan, Émilie Picard-Cantin, Claude-Guy Quimper, and Toby Walsh. The balance constraint family. In Barry O'Sullivan, editor, *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 174–189. Springer, 2014.

7   Christian Bessière, Frédéric Koriche, Nadjib Lazaar, and Barry O'Sullivan. Constraint acquisition. *Artif. Intell.*, 244:315–342, 2017. `doi:10.1016/j.artint.2015.08.001`.

8   V. Brankov, P. Hansen, and D. Stevanović. Automated conjectures on upper bounds for the largest laplacian eigenvalue of graphs. *Linear Algebra and its Applications*, 414(2):407–424, 2006.

9   Jure Brence, Ljupčo Todorovski, and Sašo Džeroski. Probabilistic grammars for equation discovery. *Knowledge-Based Systems*, 224:107077, 2021. URL: `https://www.sciencedirect.com/science/article/pii/S0950705121003403`, `doi:https://doi.org/10.1016/j.knosys.2021.107077`.

10  Céline Brouard, Simon de Givry, and Thomas Schiex. Pushing data into CP models using graphical model learning and solving. In Helmut Simonis, editor, *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*, volume 12333 of *Lecture Notes in Computer Science*, pages 811–827. Springer, 2020. `doi:10.1007/978-3-030-58475-7\_47`.

11  John William Charnley, Simon Colton, and Ian Miguel. Automatic generation of implied constraints. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 73–77. IOS Press, 2006. URL: `http://www.booksonline.iospress.nl/Content/View.aspx?piid=1649`.

12  Simon Colton, Andreas Meier, Volker Sorge, and Roy L. McCasland. Automatic generation of classification theorems for finite algebras. In David A. Basin and Michaël Rusinowitch, editors, *Automated Reasoning - Second International Joint Conference, IJCAR 2004, Cork, Ireland, July 4-8, 2004, Proceedings*, volume 3097 of *Lecture Notes in Computer Science*, pages 400–414. Springer, 2004. `doi:10.1007/978-3-540-25984-8\_30`.

13  Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli. Advancing mathematics by guiding
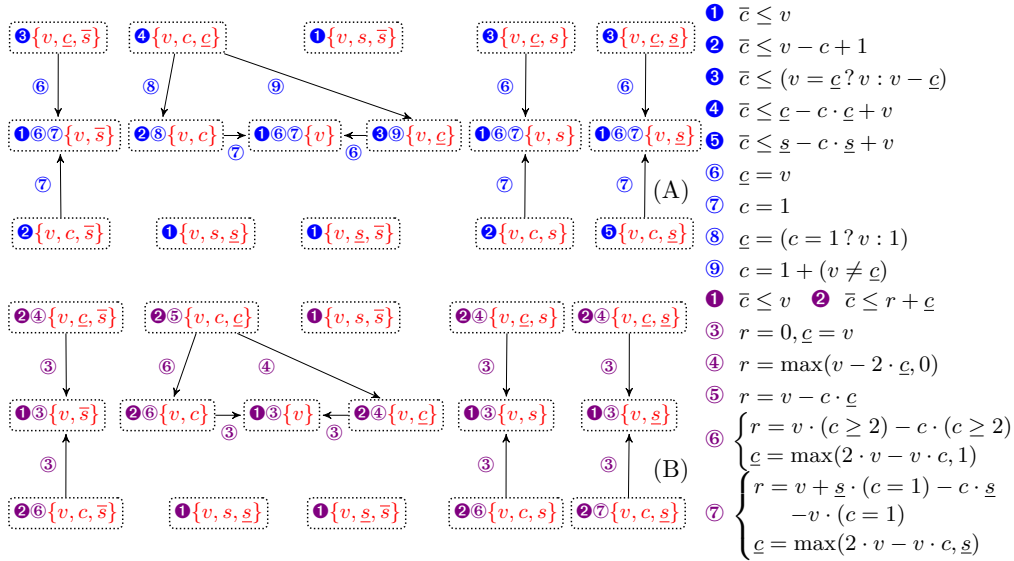
human intuition with ai. *Nature*, 600(7887):70–74, 2021. URL: https://doi.org/10.1038/s41586-021-04086-x.

14    Siemion Fajtlowicz. On conjectures of Graffiti. *Discret. Math.*, 72(1-3):113–118, 1988. doi:10.1016/0012-365X(88)90199-9.

15    Aaron M. Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1504–1511. AAAI Press, 2020. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5509.

16    Minh Hoàng Hà, Claude-Guy Quimper, and Louis-Martin Rousseau. General bounding mechanism for constraint programs. In Gilles Pesant, editor, *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, volume 9255 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 2015.

17    Pierre Hansen and Gilles Caporossi. Autographix: An automated system for finding conjectures in graph theory. *Electron. Notes Discret. Math.*, 5:158–161, 2000. doi:10.1016/S1571-0653(05)80151-9.

18    Samuel Kolb, Sergey Paramonov, Tias Guns, and Luc De Raedt. Learning constraints in spreadsheets and tabular data. *Mach. Learn.*, 106(9-10):1441–1468, 2017. doi:10.1007/s10994-017-5640-x.

19    Mohit Kumar, Stefano Teso, and Luc De Raedt. Acquiring integer programs from data. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1130–1136. ijcai.org, 2019. doi:10.24963/ijcai.2019/158.

20    Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=S1eZYeHFDS.

21    Craig E. Larson and Nicolas Van Cleemput. Automated conjecturing I: Fajtlowicz's Dalmatian heuristic revisited. *Artif. Intell.*, 231:17–38, 2016. doi:10.1016/j.artint.2015.10.002.

22    Jimmy Ho-Man Lee, Ka Lun Leung, and Yu Wai Shum. Consistency techniques for polytime linear global cost functions in weighted constraint satisfaction. *Constraints*, 19(3):270–308, 2014.

23    Doug Lenat. *AM: An artificial intelligence approach to discovery in mathematics*. PhD thesis, Stanford University, 1976.

24    Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proc. VLDB Endow.*, 8(10):1082–1093, 2015. URL: http://www.vldb.org/pvldb/vol8/p1082-papenbrock.pdf, doi:10.14778/2794367.2794377.

25    Sergey Paramonov, Samuel Kolb, Tias Guns, and Luc De Raedt. Tacle: Learning constraints in tabular data. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2511–2514. ACM, 2017. doi:10.1145/3132847.3133193.

26    Anselm Paulus, Michal Rolínek, Vít Musil, Brandon Amos, and Georg Martius. Comboptnet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints, 2021. arXiv:2105.02343.

27    Émilie Picard-Cantin, Mathieu Bouchard, Claude-Guy Quimper, and Jason Sweeney. Learning the parameters of global constraints using branch-and-bound. In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017,*

*Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10416 of *Lecture Notes in Computer Science*, pages 512–528. Springer, 2017. `doi:10.1007/978-3-319-66158-2\_33`.

**28** Steve Prestwich. Robust constraint acquisition by sequential analysis. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 355–362. IOS Press, 2020. `doi:10.3233/FAIA200113`.

**29** Gal Raayoni, Shahar Gottlieb, Yahel Manor, George Pisha, Yoav Harris, Uri Mendlovic, Doron Haviv, Yaron Hadad, and Ido Kaminer. Generating conjectures on fundamental constants with the Ramanujan Machine. *Nature*, 590:67–73, 2021. URL: `https://doi.org/10.1038/s41586-021-03229-4`.

**30** Helge Spieker and Arnaud Gotlieb. Learning objective boundaries for constraint optimization problems. In Giuseppe Nicosia, Varun Kumar Ojha, Emanuele La Malfa, Giorgio Jansen, Vincenzo Sciacca, Panos M. Pardalos, Giovanni Giuffrida, and Renato Umeton, editors, *Machine Learning, Optimization, and Data Science - 6th International Conference, LOD 2020, Siena, Italy, July 19-23, 2020, Revised Selected Papers, Part II*, volume 12566 of *Lecture Notes in Computer Science*, pages 394–408. Springer, 2020. `doi:10.1007/978-3-030-64580-9\_33`.

**31** Ljupco Todorovski. Equation discovery. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 327–330. Springer, 2010. `doi:10.1007/978-0-387-30164-8\_258`.

**32** Hao Wang. Toward mechanical mathematics. In Jörg Siekmann and Graham Wrightson, editors, *Automation of Reasoning: Classical Papers on Computational Logic 1957–1966*, pages 244–264. Springer-Verlag, Berlin, 1983.

## A    Map example

The legend of the figure lists the following conjectures:

$$❶\quad \bar{c} \le v$$
$$❷\quad \bar{c} \le v - c + 1$$
$$❸\quad \bar{c} \le (v = \underline{c}\,?\,v : v - \underline{c})$$
$$❹\quad \bar{c} \le \underline{c} - c \cdot \underline{c} + v$$
$$❺\quad \bar{c} \le \underline{s} - c \cdot \underline{s} + v$$
$$⑥\quad \underline{c} = v$$
$$⑦\quad c = 1$$
$$⑧\quad \underline{c} = (c = 1\,?\,v : 1)$$
$$⑨\quad c = 1 + (v \ne \underline{c})$$

$$❶\quad \bar{c} \le v \qquad ❷\quad \bar{c} \le r + \underline{c}$$
$$③\quad r = 0,\ \underline{c} = v$$
$$④\quad r = \max(v - 2 \cdot \underline{c}, 0)$$
$$⑤\quad r = v - c \cdot \underline{c}$$
$$⑥\quad \begin{cases} r = v \cdot (c \ge 2) - c \cdot (c \ge 2) \\ \underline{c} = \max(2 \cdot v - v \cdot c, 1) \end{cases}$$
$$⑦\quad \begin{cases} r = v + \underline{s} \cdot (c = 1) - c \cdot \underline{s} \\ \qquad - v \cdot (c = 1) \\ \underline{c} = \max(2 \cdot v - v \cdot c, \underline{s}) \end{cases}$$

**Part (A)** nodes:

- ❸$\{v, \underline{c}, \overline{s}\}$ → ⑥ → ❶⑥⑦$\{v, \overline{s}\}$ → ⑦ → ❷$\{v, c, \overline{s}\}$
- ❹$\{v, c, \underline{c}\}$ → ⑧ → ❷⑧$\{v, c\}$ ; ❶$\{v, s, \underline{s}\}$
- ❶$\{v, s, \overline{s}\}$ → ⑨ → ❶⑥⑦$\{v\}$ ← ⑥ ← ❸⑨$\{v, \underline{c}\}$ ; ❶$\{v, \underline{s}, \overline{s}\}$
- ❸$\{v, \underline{c}, s\}$ → ⑥ → ❶⑥⑦$\{v, s\}$ → ⑦ → ❷$\{v, c, s\}$
- ❸$\{v, \underline{c}, \underline{s}\}$ → ⑥ → ❶⑥⑦$\{v, \underline{s}\}$ → ⑦ → ❺$\{v, c, \underline{s}\}$

**Part (B)** nodes:

- ❷④$\{v, \underline{c}, \overline{s}\}$ → ③ → ❶③$\{v, \overline{s}\}$ → ③ → ❷⑥$\{v, c, \overline{s}\}$
- ❷⑤$\{v, c, \underline{c}\}$ → ⑥ → ❷⑥$\{v, c\}$ ; ❶$\{v, s, \underline{s}\}$
- ❶$\{v, s, \overline{s}\}$ → ④ → ❶③$\{v\}$ ← ③ ← ❷④$\{v, \underline{c}\}$ ; ❶$\{v, \underline{s}, \overline{s}\}$
- ❷④$\{v, \underline{c}, s\}$ → ③ → ❶③$\{v, s\}$ → ③ → ❷⑥$\{v, c, s\}$
- ❷④$\{v, \underline{c}, \underline{s}\}$ → ③ → ❶③$\{v, \underline{s}\}$ → ③ → ❷⑦$\{v, c, \underline{s}\}$

**Figure 4** Map $\mathcal{M}^{\bar{c} \le}_{\{v, c, \underline{c}, s, \underline{s}, \overline{s}\}}$ of upper-bounds of the output characteristic $\bar{c}$ found by the Bound Seeker, where each dotted node contains, from left to right, a reference to the maximum conjecture ❶,…,❺, ❶,❷, possibly a set of maximality conjectures ⑥,…,⑨,③,…,⑦, and the set of input characteristics in red; Part (A) corresponds to the bounds found while only using the input characteristics, and Part (B) refers to the bounds found using also the secondary characteristics.