Graph Invariants as Necessary Conditions for Global Constraints

Nicolas Beldiceanu[1], Mats Carlsson[2], Jean-Xavier Rampon[3], and Charlotte Truchet[3]

[1] LINA FRE CNRS 2729, École des Mines de Nantes, 44307 Nantes Cedex 3, France.
Nicolas.Beldiceanu@emn.fr
[2] SICS, P.O. Box 1263, SE-164 29 Kista, Sweden.
Mats.Carlsson@sics.se
[3] LINA FRE CNRS 2729, 2 rue de la Houssinière, BP-92208, 44322 Nantes Cedex 3, France.
{Jean-Xavier.Rampon,Charlotte.Truchet}@lina.univ-nantes.fr

May 10, 2005

SICS Technical Report T2005:07

ISSN: 1100-3154

ISRN: SICS-T–2005/07-SE

Abstract. This report presents a database of about 200 graph invariants for deriving systematically necessary conditions from the graph properties based representation of global constraints. This scheme is based on invariants on the graph characteristics used in the description of a global constraint. A SICStus Prolog implementation based on arithmetic and logical constraints as well as on indexicals is available.

Keywords: global constraint, implied constraint, graph invariant.

1 Introduction

Adding necessary conditions to a constraint program has been recognized in the early time of constraint programming [1] as a key point in order to enhance efficiency. However this was usually done manually after a careful analysis of the problem under consideration or by identifying typical constraints patterns [2]. Beldiceanu presented in [3] a systematic description of global constraints in terms of graph properties: among the 227 constraints of the catalog of global constraints [3], about 200 constraints are described as a conjunction of graph properties where each graph property has the form $P\ op\ V$, where P is a graph characteristic, op is a comparison operator in $\{\leq, \geq, =, \neq\}$, and V a variable that ranges over a finite set of integers (a *domain variable*). Within this context, this report presents a database of graph invariants: given a specification of a constraint C in terms of graph properties, we can automatically extract, from that database, graph invariants that mention the graph characteristics used in the specification of C, and post these invariants as necessary conditions for the feasibility of C.

Example 1. Consider the `nvalue(`N`,` $\{x_1, ..., x_m\}$`)` constraint [4], where $N, x_1, ..., x_m$ are domain variables. The `nvalue` constraint holds iff the number of distinct values assigned to the variables in $\mathcal{X} = \{x_1, ..., x_m\}$ is equal to N. It can been seen as enforcing the following graph property: the number of strongly connected components of the *intersection graph* $G(\mathcal{X}, E)$, where $E = \{x_i \in \mathcal{X}, x_j \in \mathcal{X} : x_i = x_j\}$, is equal to N. From Bessière et al. [5] we have the necessary condition $\mathbf{NSCC} \geq \left\lceil \frac{\mathbf{NVERTEX}^2}{\mathbf{NARC}} \right\rceil$ (see Turán [6]) relating the number of arcs $\mathbf{NARC}$, the number of vertices $\mathbf{NVERTEX}$ and the number of strongly connected components $\mathbf{NSCC}$ of the *intersection graph*.

Using graph invariants is especially useful when a global constraint mentions more than one graph property in its description. In this context, these graph properties involve several graph characteristics that cannot vary independently.

Example 2. Consider again the `nvalue` constraint introduced in Example 1, and assume we want to put a restriction on the minimum and the maximum number of occurrences (respectively denoted by $\underline{occ}$ and by $\overline{occ}$) of each value that is effectively used. In terms of the intersection graph, this can be interpreted as putting a restriction on the number of vertices of its strongly connected components. Let $\mathbf{MIN\_NSCC}$ and $\mathbf{MAX\_NSCC}$ respectively denote the number of vertices of the smallest and the largest strongly connected components of the intersection graph. Our initial constraint on the minimum and maximum number of occurrences is now expressed by $\mathbf{MIN\_NSCC} \geq \underline{occ}$ and $\mathbf{MAX\_NSCC} \leq \overline{occ}$. We have recast our original balanced assignment problem to the search of a digraph on which we restrict its number of vertices $\mathbf{NVERTEX}$[1], its number of strongly connected components $\mathbf{NSCC}$, and the sizes $\mathbf{MIN\_NSCC}$ and $\mathbf{MAX\_NSCC}$ of its smallest and largest strongly connected components. By querying our database of invariants in order to extract those graph invariants that only mention the four graph characteristics $\mathbf{NVERTEX}$, $\mathbf{NSCC}$, $\mathbf{MIN\_NSCC}$ and $\mathbf{MAX\_NSCC}$ we get the following invariants $\mathbf{NVERTEX} \leq \max(0, \mathbf{NSCC} - 1) \cdot \mathbf{MAX\_NSCC} + \mathbf{MIN\_NSCC}$ and $\mathbf{NVERTEX} \geq \max(0, \mathbf{NSCC} - 1) \cdot \mathbf{MIN\_NSCC} + \mathbf{MAX\_NSCC}$, which are necessary conditions for the balanced assignment constraint.

Section 2 recalls the graph-based representation of global constraints. Section 3 introduces graph invariants, while Section 4 presents the database of graph invariants. The database and its 200 graph invariants and their corresponding proofs is available in Chapter 3 of [3]. Finally, Section 4 provides an evaluation of the approach on two constraints, which mention various graph characteristics.

2 Graph-Based Representation of Global Constraint

This section summarizes the representation of global constraints as graph properties in [3] and illustrates this framework on the `group` [7] and the `change_continuity` [3] constraints, which will be used throughout this report. They both correspond to timetabling constraints which allow for expressing conditions on sliding sequences of consecutive working days of a given person.

[1] In fact, $\mathbf{NVERTEX}$ is fixed to the number of variables of the `nvalue` constraint.

The graph-based representation. A global constraint C is represented as an initial digraph $G_i = (\mathcal{X}_i, E_i)$: to each vertex in $\mathcal{X}_i$ corresponds a variable involved in C, while to each arc e in E_i corresponds a binary constraint involving the variables at both extremities of e. To generate G_i from the parameters of C, the set of arc generators described in [3] is used. Figure 1 illustrates the most commonly used arc generators by depicting the initial digraph generated from a sequence of four vertices. When all variables of C are fixed, we remove from G_i all binary constraints that do not hold as well as isolated vertices, i.e., vertices that are not extremities of an arc. This final digraph is denoted by G_f. C is equivalent to a conjunction of graph properties which should be satisfied by G_f. Within the global constraint catalog [3], commonly used graph characteristics on the final digraph G_f are:

- $\mathbf{NARC}$ and $\mathbf{NVERTEX}$ denote the number of arcs and vertices,
- $\mathbf{NCC}$ and $\mathbf{NSCC}$ denote the number of connected and strongly connected components,
- $\mathbf{MIN\_NCC}$ and $\mathbf{MAX\_NCC}$ (resp. $\mathbf{MIN\_NSCC}$ and $\mathbf{MAX\_NSCC}$) respectively denote the number of vertices of the smallest and the largest connected components (resp. the strongly connected components).

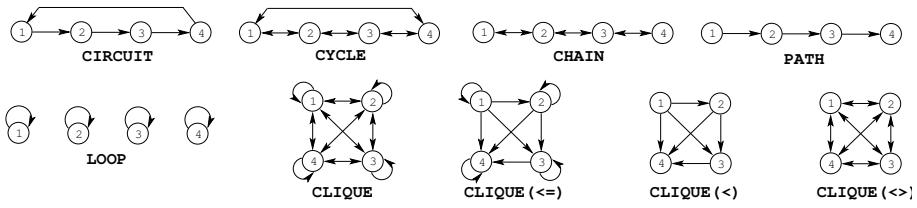

Fig. 1. Examples of arc generators

Illustrative examples of the graph-based representation. We now define the `group` and the `change_continuity` constraints and present their links with the graph-based description. Since they respectively use 6 and 8 graph characteristics these constraints can potentially benefit from the use of graph invariants.

Example 3. The first six parameters of the `group(NGROUP,MIN_SIZE,MAX_SIZE, MIN_DIST,MAX_DIST,NVAL,VARIABLES,VALUES)` constraint are domain variables, while `VARIABLES` is a sequence of domain variables and `VALUES` a finite set of integers. Let n denote the number of variables of the sequence `VARIABLES`. Let $X_i, X_{i+1}, \ldots, X_j$ $(1 \leq i \leq j \leq n)$ be consecutive variables of the sequence `VARIABLES` such that all the following conditions simultaneously apply: (1) All variables $X_i, \ldots, X_j$ take their value in the set of values `VALUES`, (2) $i = 1$ or X_{i-1} does not take a value in `VALUES`, (3) $j = n$ or X_{j+1} does not take a value in `VALUES`. We call such a set of variables a *group*. The constraint `group` is fulfilled if all the following conditions hold:

- There are exactly `NGROUP` groups of variables,
- `MIN_SIZE` and `MAX_SIZE` are the number of variables of the smallest and largest group,

- MIN_DIST and MAX_DIST are the minimum and maximum number of variables between two consecutive groups or between one border and one group,
- NVAL is the number of variables that take their value in the set of values VALUES.

group$(2, 2, 4, 1, 2, 6, \langle 0, 0, 1, 3, 0, 2, 2, 2, 3\rangle, \{1, 2, 3\})$ holds since the sequence $\langle 0, 0, 1, 3, 0, 2, 2, 2, 3\rangle$ contains 2 groups $\langle 1, 3\rangle$ and $\langle 2, 2, 2, 3\rangle$ of non-zero values of size 2 and 4, 2 groups $\langle 0, 0\rangle$ and $\langle 0\rangle$ of zeros, and 6 non-zero values. The graph-based description of the group constraint uses two graph constraints which respectively mention the graph properties $\mathbf{NCC}$ = NGROUP, $\mathbf{MIN\_NCC}$ = MIN_SIZE, $\mathbf{MAX\_NCC}$ = MAX_SIZE, $\mathbf{NVERTEX}$ = NVAL and $\mathbf{MIN\_NCC}$ = MIN_DIST, $\mathbf{MAX\_NCC}$ = MAX_DIST. The leftmost part of Figure 2 depicts the initial graph of well as the two final graphs associated to the two graph constraints of the example given for the group constraint.

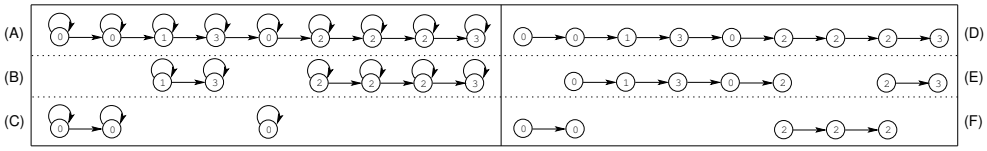

Fig. 2. Initial (A) and final graphs (B,C) of group$(2, 2, 4, 1, 2, 6, \langle 0, 0, 1, 3, 0, 2, 2, 2, 3\rangle, \{1, 2, 3\})$. Initial (D) and final graphs (E,F) of change_continuity$(2, 2, 2, 5, 2, 3, 5, 3, \langle 0, 0, 1, 3, 0, 2, 2, 2, 3\rangle, \neq)$.

Example 4. The first eight parameters of the change_continuity(NB_PERIOD_CHANGE, NB_PERIOD_CONTINUITY,MIN_SIZE_CHANGE,MAX_SIZE_CHANGE,MIN_SIZE_CONTINUITY,MAX_SIZE_CONTINUITY,NB_CHANGE,NB_CONTINUITY,VARIABLES,CTR) constraint are domain variables, while VARIABLES is a sequence of domain variables and CTR a binary constraint in $\{=, \neq, \leq, >, \geq, <\}$. A *change* (resp. *continuity*) is defined by the fact that constraint CTR holds (resp. does not hold) between two consecutive variables of the sequence VARIABLES. Let n denote the number of variables of the sequence VARIABLES, and let $X_i, X_{i+1}, \ldots, X_j (1 \leq i < j \leq n)$ be consecutive variables of the sequence VARIABLES. $X_i, X_{i+1}, \ldots, X_j$ corresponds to a *period of change* if X_k CTR X_{k+1} holds for all $k \in [i, j-1]$, and if $i = 1$ or X_{i-1} CTR X_i does not hold, and if $j = n-1$ or X_j CTR X_{j+1} does not hold. A *period of continuity* is defined in a similar way by considering the negation of CTR . The constraint change_continuity holds if and only if:

- NB_PERIOD_CHANGE and NB_PERIOD_CONTINUITY are respectively equal to the number of periods of change and of continuity,
- MIN_SIZE_CHANGE and MAX_SIZE_CHANGE are respectively equal to the number of variables of the smallest and largest period of change,
- MIN_SIZE_CONTINUITY and MAX_SIZE_CONTINUITY are respectively equal to the number of variables of the smallest and largest period of continuity,
- NB_CHANGE and NB_CONTINUITY are respectively equal to the total number of changes and continuities.

change_continuity$(2, 2, 2, 5, 2, 3, 5, 3, \langle 0, 0, 1, 3, 0, 2, 2, 2, 3\rangle, \neq)$ holds since the sequence $\langle 0, 0, 1, 3, 0, 2, 2, 2, 3\rangle$ contains 2 periods of changes $\langle 0, 1, 3, 0, 2\rangle$ and $\langle 2, 3\rangle$ of minimum and maximum size 2 and 5, 2 periods of continuities $\langle 0, 0\rangle$ and $\langle 2, 2, 2\rangle$

of minimum and maximum size 2 and 3. Finally, the total number of changes and continuities are respectively equal to 5 and 3. The graph-based description of the change_continuity(NB_PERIOD_CHANGE,NB_PERIOD_CONTINUITY,MIN_SIZE_-CHANGE,MAX_SIZE_CHANGE,MIN_SIZE_CONTINUITY,MAX_SIZE_CONTINUITY,NB_-CHANGE,NB_CONTINUITY,VARIABLES,CTR) constraint uses two graph constraints which respectively mention the graph properties $\mathbf{NCC}$ = NB_PERIOD_CHANGE, $\mathbf{MIN\_NCC}$ = MIN_SIZE_CHANGE, $\mathbf{MAX\_NCC}$ = MAX_SIZE_CHANGE, $\mathbf{NARC}$ = NB_CHANGE and $\mathbf{NCC}$ = NB_PERIOD_CONTINUITY, $\mathbf{MIN\_NCC}$ = MIN_SIZE_CONTINUITY, $\mathbf{MAX\_NCC}$ = MAX_SIZE_CONTINUITY, $\mathbf{NARC}$ = NB_CONTINUITY. The rightmost part of Figure 2 depicts the initial graph of well as the two final graphs associated to the two graph constraints of the example given for the change_continuity constraint.

3 Graph Invariants

Within the scope of the graph-based description this section introduces implied constraints which are systematically linked to the description of a global constraint:

- We then describe the different contexts where graph invariants can be used.
- Finally, we show how to get sharper graph invariants by taking advantage of the structure of the global constraint under consideration.

Since no final digraph contains isolated vertices, the database of graph invariants considers digraphs for which each vertex has at least one arc.

Context for Using Graph Invariants. They can be used in the following contexts:

- Quite often, it happens that one wants the final digraph to satisfy more than one graph property. This was illustrated by the balanced assignment constraint (see Example 2) as well as by the group and change_continuity constraints. In this context, these graph properties involve several graph characteristics which cannot vary independently.
- Even if the description of a global constraint involves one single graph characteristic $\mathbf{C}$, we can introduce the number of vertices, $\mathbf{NVERTEX}$, and the number of arcs, $\mathbf{NARC}$, of the final digraph. In this context, we can take advantage of graph invariants linking $\mathbf{C}$, $\mathbf{NARC}$ and $\mathbf{NVERTEX}$. This is in fact what was done for the nvalue constraint in Example 1.
- It also happens that we enforce two graph constraints $\mathcal{GC}_1$ and $\mathcal{GC}_2$, which have the same initial digraph $\mathcal{G}$. In this context we consider the following situations:
  - Each arc of $\mathcal{G}$ belongs to one of the final digraphs associated to $\mathcal{GC}_1$ or to $\mathcal{GC}_2$ (but not to both). An example of such global constraints is the change_continuity constraint depicted by Example 4.
  - Each vertex of $\mathcal{G}$ belongs to one of the final digraphs associated to $\mathcal{GC}_1$ or to $\mathcal{GC}_2$ (but not to both). An example of such global constraint is the group constraint depicted by Example 3.

  In these situations the graph properties associated to the two graph constraints are not independent. This will be illustrated by Example 12.

Graph Classes. By definition, a graph invariant has to hold for any final digraph. For instance, we have the graph invariant $\mathbf{NARC} \leq \mathbf{NVERTEX}^2$, which relates the number of arcs and the number of vertices of any digraph. This invariant is sharp since the equality is reached for a clique. However, by considering the structure of a final digraph, we can get sharper invariants. For instance, if our final digraph is a subset of an elementary path (e.g. we use the $PATH$ arc generator depicted by Figure 1) we have that $\mathbf{NARC} \leq \mathbf{NVERTEX} - 1$, which is a tighter bound of the maximum number of arcs since $\mathbf{NVERTEX} - 1 < \mathbf{NVERTEX}^2$. For this reason, we consider recurring graph classes that show up for different global constraints. For a given global constraint, a graph class specifies a general property which holds on all its final digraphs. In addition, we also consider graph constraints such that their final digraph is a subset of the digraph generated by the arc generators depicted by Figure 1.

Example 5. We provide typical examples of graph classes and, for each of them, we point to some global constraints that fit in that class:

- `acyclic`: graph constraint for which the final digraph doesn't have any circuit (e.g. `change` [7], `change_continuity` [3], `common` [3]).
- `apartition`: constraint defined by two graph constraints having the same initial digraph, where each arc of the initial digraph belongs to one of the final digraphs (but not to both) (e.g. `change_continuity` [3]).
- `bipartite`: graph constraint for which the final digraph is bipartite (e.g. `alldifferent_on_intersection` [3], `common` [3]).
- `consecutive_loops_are_connected`: denotes the fact that the graph constraints of a global constraint use only the $PATH$ and the $LOOP$ arc generators and that their final digraphs do not contain consecutive vertices which have a loop and which are not connected together by an arc (e.g. `group` [3]).
- `equivalence`: graph constraint for which the final digraph is reflexive, symmetric and transitive (e.g. `balance` [3], `nvalue` [5]).
- `no_loop`: graph constraint for which the final digraph doesn't have any loop (e.g. `change_continuity` [3], `common` [3]).
- `one_succ`: graph constraint for which all the vertices of the initial digraph belong to the final digraph and for which all vertices of the final digraph have exactly one successor (e.g. `alldifferent` [8], `cycle` [9], `tree` [10]).
- `symmetric`: graph constraint for which the final digraph is symmetric (e.g. `connect_points` [3]).
- `vpartition`: constraint defined by two graph constraints having the same initial digraph, where each vertex of the initial digraph belongs to one of the final digraphs (but not to both) (e.g. `group` [3]).

4 The Database of Graph Invariants

This section introduces the database of graph invariants we have built so far. It first provides a taxonomy of graph invariants and discusses their implementation. It then presents the organisation of the database. Finally, it explains how to use the database in order to automatically extract the relevant invariants for a given global constraint.

Taxonomy of Graph Invariants. Within the database of graph invariants we currently have seven categories of graph invariants. These categories steem from the structure of the formulae associated to the invariants.

I1. Invariants involving *one single* graph characteristics C, restricting the initial set of possible values of C.

Example 6. When the final digraph does not contain any loops, we have that $2 \cdot \mathbf{NCC} \leq \mathbf{NVERTEX}_{\mathtt{INITIAL}}$, where $\mathbf{NVERTEX}_{\mathtt{INITIAL}}$ is the number of vertices of the initial digraph and where $\mathbf{NCC}$ is the number of connected components of the final digraph. This invariant restricts the initial domain of $\mathbf{NCC}$ to $\left[0, \left\lfloor \frac{\mathbf{NVERTEX}_{\mathtt{INITIAL}}}{2} \right\rfloor\right]$.

I2. Invariants characterizing the *lower bound* (resp. *upper bound*) of a given graph characteristics C in terms of other graph characteristics $C_1, \ldots, C_n (n > 1, C_i \neq C)$. They are defined as an inequality of the form $C \geq f(C_1, \ldots, C_n)$ (resp. $C \leq f(C_1, \ldots, C_n)$), where $f(C_1, \ldots, C_n)$ is a formula involving the graph characteristics $C_1, \ldots, C_n$.

Example 7. As illustrated by Figure 3, the invariant $\mathbf{NARC} \geq \mathbf{NVERTEX} - \left\lfloor \frac{\mathbf{NSCC}-1}{2} \right\rfloor$ can be interpreted as the minimum number of arcs $\mathbf{NARC}$ of a digraph according to a fixed number of vertices $\mathbf{NVERTEX}$ and a fixed number of strongly connected components $\mathbf{NSCC}$.

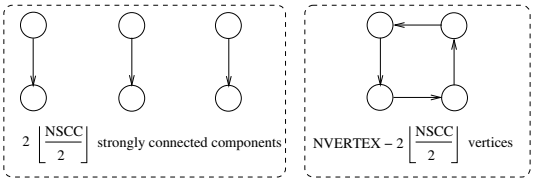

Fig. 3. A digraph which achieves the minimum number of arcs according to a fixed number of strongly connected components as well as to a fixed number of vertices ($\mathbf{NSCC} = 7, \mathbf{NVERTEX} = 10, \mathbf{NARC} = 10 - \left\lfloor \frac{7}{2} \right\rfloor = 7$)

I3. Invariants defining, for a given graph characteristics C, a *forbidden interval* of values of the form $[f_1(C_1, \ldots, C_n), f_2(C_{n+1}, \ldots, C_m)]$, where $f_1(C_1, \ldots, C_n)$ and $f_2(C_{n+1}, \ldots, C_m)$ are formulae involving graph characteristics distinct from C. These invariants usually come from a disjunction of the form $C \leq f_1(C_1, \ldots, C_n) - 1 \vee C \geq f_2(C_{n+1}, \ldots, C_m) + 1$.

Example 8. Consider the invariant $\mathbf{MIN\_NCC} \notin \left[\left\lfloor \frac{\mathbf{NVERTEX}}{2} \right\rfloor + 1, \mathbf{NVERTEX} - 1\right]$, which specifies that the number of vertices $\mathbf{MIN\_NCC}$ of the smallest connected component of a digraph does not belong to an interval defined according to the number of vertices $\mathbf{NVERTEX}$. This invariant stems from the following disjunction:

- On the one hand, if the digraph contains no more than one connected component, we have that $\mathbf{MIN\_NCC} \geq \mathbf{NVERTEX}$,
- On the other hand, if the digraph contains at least two connected components, we have that $\mathbf{MIN\_NCC} + \mathbf{MIN\_NCC} \leq \mathbf{NVERTEX}$.

I4. Invariants of the form $C \leq \max(f_1(C_1, \ldots, C_n), f_2(C_{n+1}, \ldots, C_m))$, where C is a graph characteristics and $f_1(C_1, \ldots, C_n)$ and $f_2(C_{n+1}, \ldots, C_m)$ are formulae involving graph characteristics distinct from C. These invariants usually come from a disjunction of two invariants $C \leq f_1(C_1, \ldots, C_n) \vee C \leq f_2(C_{n+1}, \ldots, C_m)$.

Example 9. Consider the invariant $\mathbf{MAX\_NCC} \leq \max(\mathbf{NVERTEX} - \mathbf{MIN\_NCC}, \mathbf{MIN\_NCC})$, which restricts the maximum number of vertices $\mathbf{MAX\_NCC}$ of the largest connected component according to the number of vertices in the smallest connected component and to the number of vertices $\mathbf{NVERTEX}$. This invariant stems from the following disjunction:

- On the one hand, if the digraph contains no more than one connected component, we have that $\mathbf{MAX\_NCC} \leq \mathbf{MIN\_NCC}$,
- On the other hand, if the digraph contains at least two connected components, we have that $\mathbf{NVERTEX} \geq \mathbf{MIN\_NCC} + \mathbf{MAX\_NCC}$ (i.e. $\mathbf{MAX\_NCC} \leq \mathbf{NVERTEX} - \mathbf{MIN\_NCC}$).

I5. Invariants described by an implication between two conditions. These invariants have the form $\mathcal{C}ond_1 \Rightarrow \mathcal{C}ond_2$ where $\mathcal{C}ond_1$ is a condition involving one or two graph characteristics, and where $\mathcal{C}ond_2$ is either a condition involving one or two graph characteristics, either an invariant of type **I2** or **I3**.

Example 10. As an example, consider the invariant $\mathbf{MIN\_NCC} \neq \mathbf{MAX\_NCC} \Rightarrow \mathbf{NCC} \geq 2$, which depicts the fact that, if the number of vertices of the smallest connected component is not equal to the size of the largest connected component, the number of connected components is at least 2.

I6. Invariants depicted by an equivalence between two given conditions where each condition involves one single graph characteristics.

Example 11. $\mathbf{MAX\_NCC} = 0 \Leftrightarrow \mathbf{MIN\_NCC} = 0$ is an instance of such invariant.

I7. Invariants involving graph characteristics coming from more than one graph constraint.

Example 12. $\texttt{apartition} \wedge \mathbf{arc\_gen} = \mathit{PATH} : |\mathbf{NCC}_1 - \mathbf{NCC}_2| \leq 1$ [2] is an invariant which can be applied when:

- As specified by `apartition`, a global constraint is defined by two graph constraints having the same initial digraph, where each arc of the initial digraph belongs to one of the final digraphs (but not to both),
- All the graph constraints of a global constraint use only the arc generator PATH.

This is in fact the situation of the `change_continuity` constraint introduced in Example 4: in this context, this invariant enforces the number of groups of changes `NB_CHANGE` and the number of groups of continuities `NB_CONTINUITY` to differ by at most 1.

[2] $\mathbf{NCC}_1$ and $\mathbf{NCC}_2$ respectively denote the number of connected components of the final digraph of a first graph constraint and the number of connected components of the final digraph of a second graph constraint.

Each graph invariant has a precondition which defines its applicability. The precondition consists of an, possibly empty, conjunction of elementary conditions which characterize the graph class for which it can be applied. An elementary condition is either one of the keywords `acyclic`, `bipartite`, `no_loop`, `one_succ`, `symmetric`, `equivalence`, `apartition`, `vpartition`, `consecutive_loops_are_connected` characterizing a specific graph class which was previously introduced, either an expression of the form `arc_gen` $=$ *arc generator*, where *arc generator* is an arc generator used for generating the arcs of the initial digraph.

Example 13. Consider the graph invariants $\mathbf{NARC} \leq \mathbf{NVERTEX}^2$ and `arc_gen` $=$ *PATH* : $\mathbf{NARC} \leq \mathbf{NVERTEX} - 1$ of type **I3** which both relate the number of arcs and the number of vertices of a digraph. The first one has no precondition and therefore holds on any digraph, while the second one applies only on those digraphs that are a subset of an elementary path.

Implementing Graph Invariants. Most graph invariants are usually directly implemented as constraints which directly reduce the domains of the graph characteristics they involve. For this purpose we use:

- The arithmetic constraints of SICStus, which include constraints over non linear expressions [11, page 501],
- Propositional formulae over arithmetic constraints [11, page 461].

Finally, we also use indexicals [12, 13] for implementing some graph invariants. An *indexical* is a reactive function rule of the form X in R, where X is a domain variable and R is a set valued range expression.

Indexicals are used for encoding invariants that define a forbidden interval of values for a given graph characteristics (e.g. category **I3**) and for explicitly implementing the propagation of some non-linear arithmetic constraints for which the existing constraint propagation is too weak. Invariants of category **I3** have the form $C \notin [f_1(C_1, \ldots, C_n), f_2(C_{n+1}, \ldots, C_m)]$, where $f_1(C_1, \ldots, C_n)$ and $f_2(C_{n+1}, \ldots, C_m)$ are formulae involving the graph characteristics $C_1, \ldots, C_m$ distinct from C. The idea is to evaluate the maximum value, U, of $f_1(C_1, \ldots, C_n)$ as well as the minimum value, L, of $f_2(C_{n+1}, \ldots, C_m)$ and to remove from C all values in $[U, L]$ when $U \leq L$. For this purpose we write range expressions for defining L and U.

Example 14. As an illustrative example of how to encode invariants defining a forbidden interval of values, consider the constraint $X \leq L \vee X \geq R$, which comes in handy for invariants such as $\mathbf{MIN\_NCC} \notin \left[\lfloor \frac{\mathbf{NVERTEX}}{2} \rfloor + 1, \mathbf{NVERTEX} - 1\right]$. This constraint can be encoded by three indexicals maintaining bounds consistency as follows:

```
not_strictly_between(X, L, U) +:
  X in (inf..max(L)) \/ (min(U)..sup),
  L in ((min(U)..max(X)) ? (inf   ..sup   )) \/ (min(X)..sup   ),
  U in ((min(X)..max(L)) ? (inf   ..sup   )) \/ (inf   ..max(X)).
```

Database Organisation. As we previously saw, we have graph invariants that hold for any digraph as well as tighter graph invariants for specific graph classes. As a consequence, we partition the database into groups of graph invariants. A *group of graph*

invariants corresponds to several invariants such that all invariants relate to the same subset of graph characteristics and are variations of the first invariant of the group taking into accounts the graph class. Thus, the first invariant of a group has no precondition, while all other invariants have a non-empty precondition that characterizes the graph class for which they hold.

Example 15. As a first example, consider the following group of invariants, which relate the number of arcs $\mathbf{NARC}$ to the number of vertices of the smallest and largest connected component (i.e. $\mathbf{MIN\_NCC}$ and $\mathbf{MAX\_NCC}$) of a digraph:

- $\mathbf{MIN\_NCC} \neq \mathbf{MAX\_NCC} \Rightarrow \mathbf{NARC} \geq \mathbf{MIN\_NCC} + \mathbf{MAX\_NCC} - 2 + (\mathbf{MIN\_NCC} = 1)$[3],
- `equivalence` : $\mathbf{MIN\_NCC} \neq \mathbf{MAX\_NCC} \Rightarrow \mathbf{NARC} \geq \mathbf{MIN\_NCC}^2 + \mathbf{MAX\_NCC}^2$.

On the one hand, since the first invariant has no precondition, it can be applied to any digraph. On the other hand, the second invariant specifies a tighter condition (since $\mathbf{MIN\_NCC}^2 + \mathbf{MAX\_NCC}^2 \geq \mathbf{MIN\_NCC} + \mathbf{MAX\_NCC} - 2 + (\mathbf{MIN\_NCC} = 1)$) which only holds for a digraph that is reflexive, symmetric and transitive.

Example 16. As a second example, consider the following group of invariants, which relate the number of arcs $\mathbf{NARC}$ to the number of vertices $\mathbf{NVERTEX}$ according to the arc generator (see Figure 1) used for generating the initial digraph:

- $\mathbf{NARC} \leq \mathbf{NVERTEX}^2$,
- $\mathbf{arc\_gen} = \mathit{CIRCUIT} : \mathbf{NARC} \leq \mathbf{NVERTEX}$,
- $\mathbf{arc\_gen} = \mathit{CHAIN} : \mathbf{NARC} \leq 2 \cdot \mathbf{NVERTEX} - 2$,
- $\mathbf{arc\_gen} = \mathit{CLIQUE}(\leq) : \mathbf{NARC} \leq \frac{\mathbf{NVERTEX} \cdot (\mathbf{NVERTEX}+1)}{2}$,
- $\mathbf{arc\_gen} = \mathit{CLIQUE}(<) : \mathbf{NARC} \leq \frac{\mathbf{NVERTEX} \cdot (\mathbf{NVERTEX}-1)}{2}$,
- $\mathbf{arc\_gen} = \mathit{CLIQUE}(\neq) : \mathbf{NARC} \leq \mathbf{NVERTEX}^2 - \mathbf{NVERTEX}$,
- $\mathbf{arc\_gen} = \mathit{CYCLE} : \mathbf{NARC} \leq 2 \cdot \mathbf{NVERTEX}$,
- $\mathbf{arc\_gen} = \mathit{PATH} : \mathbf{NARC} \leq \mathbf{NVERTEX} - 1$.

The database currently contains 13, 50, 34, 12, 2 groups of invariants respectively mentioning 1, 2, 3, 4 and 5 graph characteristics. It also contains groups of invariants relating the graph characteristics of two digraphs. It contains 8, 6, 4, 10, 2 groups respectively mentioning 2, 3, 4, 5, 6 graph characteristics.

Extracting the Relevant Invariants. Once we have the graph invariants we can use them systematically by applying the following steps:

- For a given graph constraint we extract all the graph characteristics occurring in its description. This can be done automatically by scanning the corresponding graph properties. Let $\mathcal{GC}$ denote this subset of graph characteristics. For each graph characteristic gc of $\mathcal{GC}$ we check if we have a graph property of the form $gc = var$ where var is a domain variable. If this is the case we record the pair (gc, var); if not, we create a new domain variable var and also record the pair (gc, var).

[3] The expression $(\mathbf{MIN\_NCC} = 1)$ is equal to 1 if $\mathbf{MIN\_NCC} = 1$ and 0 otherwise.

- We then search for all groups of graph invariants involving a subset of the previous graph characteristics $\mathcal{GC}$. For each selected group we filter out those graph invariants for which the preconditions are not compatible with the graph class of the graph constraint under consideration. In each group we finally keep those invariants that have the maximum number of preconditions (i.e. the most specialized graph invariants).
- Finally we state all the previously collected graph invariants as implied constraints. This is achieved by using the variables associated to each graph characteristic.

5 Experimental Results

This section illustrates the approach on the `group` as well as on the `change_continuity` global constraints, which were previously introduced. We have compared the following approaches:

- In a first approach each graph characteristic was handled independently. This was concretely done by constructing an automaton for each graph characteristic and by reformulating that automaton as a conjunction of constraints as described in [14].
- The second approach reuses the first one but, in addition, also exploits the database of graph invariants in order to generate invariants which link the graph characteristics used in the description of `group` and of `change_continuity`.

We first detail the automata used for the `group` constraint as well as the graph invariants. Since it is very similar to the `group` constraint, we then shortly discuss the implementation of the `change_continuity` constraint. Finally, we present the computational results obtained for the first and second approaches on the `group` as well as on the `change_continuity` constraints.

Implementing the `group` Constraint. Parts (A), (B), (C) and (D) of Figure 4 respectively depict the automata associated to the graph characteristics $\mathbf{NCC}$, $\mathbf{MIN\_NCC}$, $\mathbf{MAX\_NCC}$ and $\mathbf{NVERTEX}$ of the first graph constraint. Each automaton is applied to the sequence of variables corresponding to the `VARIABLES` parameter. A transition with a standard line depicts the fact that a variable takes its value within the set `VALUES`, while a thick line denotes the fact that a variable does not take its value within `VALUES`. Finally, a transition with a dashed line indicates the end of the sequence of variables. Since all the four automata use counters, we indicate how these counters are initialized in the initial state `s`, how a counter is unified to an argument of the `group` constraint in the final state `t`, and how they are possibly updated on a given transition. When there are several transitions between a given pair of states, we indicate with a dotted line or a standard line its type (see for instance the two transitions between `s` and `s` of the automaton depicted by part (C)).

The automata associated to $\mathbf{MIN\_NCC}$ = `MIN_DIST` and to $\mathbf{MAX\_NCC}$ = `MAX_DIST` are similar to the automata depicted by part (B) and (C), except that we change a thick line to a standard line and vice versa. The first approach for implementing the `group` constraint uses these six automata we just depicted. In the second

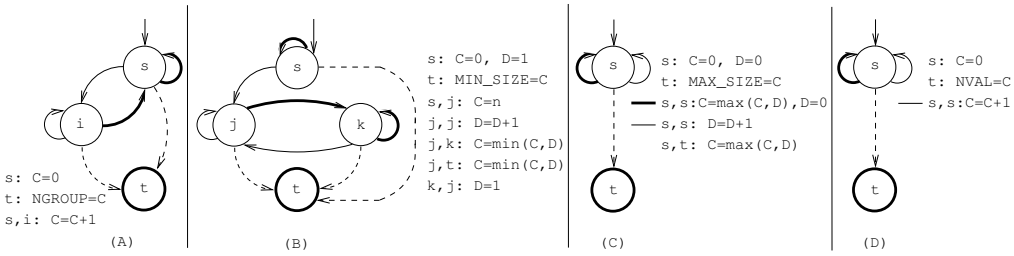

Fig. 4. Automata associated to the graph characteristics of the `group` constraint

approach we reuse the six automata and, in addition, extract the following set of 51 graph invariants from the database of invariants[4]:

$$2 \cdot \mathbf{NCC}_1 \leq n+1, \quad 2 \cdot \mathbf{NCC}_2 \leq n+1$$
$$\mathbf{MAX\_NCC}_1 = 0 \Leftrightarrow \mathbf{MIN\_NCC}_1 = 0, \quad \mathbf{MAX\_NCC}_2 = 0 \Leftrightarrow \mathbf{MIN\_NCC}_2 = 0$$
$$\mathbf{MIN\_NCC}_1 \leq \mathbf{MAX\_NCC}_1, \quad \mathbf{MIN\_NCC}_2 \leq \mathbf{MAX\_NCC}_2$$
$$\mathbf{MAX\_NCC}_1 = 0 \Leftrightarrow \mathbf{NVERTEX}_1 = 0, \quad \mathbf{MAX\_NCC}_2 = 0 \Leftrightarrow \mathbf{NVERTEX}_2 = 0$$
$$\mathbf{NVERTEX}_1 \geq \mathbf{MAX\_NCC}_1, \quad \mathbf{NVERTEX}_2 \geq \mathbf{MAX\_NCC}_2$$
$$(\mathbf{MIN\_NCC}_1 + 1) \cdot \mathbf{NCC}_1 \leq n+1, \quad (\mathbf{MIN\_NCC}_2 + 1) \cdot \mathbf{NCC}_2 \leq n+1$$
$$\mathbf{MIN\_NCC}_1 = 0 \Leftrightarrow \mathbf{NVERTEX}_1 = 0, \quad \mathbf{MIN\_NCC}_2 = 0 \Leftrightarrow \mathbf{NVERTEX}_2 = 0$$
$$\mathbf{NVERTEX}_1 \geq \mathbf{MIN\_NCC}_1, \quad \mathbf{NVERTEX}_2 \geq \mathbf{MIN\_NCC}_2$$
$$\mathbf{NCC}_1 = 0 \Leftrightarrow \mathbf{NVERTEX}_1 = 0, \quad \mathbf{NCC}_2 = 0 \Leftrightarrow \mathbf{NVERTEX}_2 = 0$$
$$\mathbf{NCC}_1 \leq \mathbf{NVERTEX}_1, \quad \mathbf{NCC}_2 \leq \mathbf{NVERTEX}_2$$
$$\mathbf{NVERTEX}_1 \leq n - (\mathbf{NCC}_1 - 1), \quad \mathbf{NVERTEX}_2 \leq n - (\mathbf{NCC}_2 - 1)$$
$$\mathbf{MIN\_NCC}_1 \neq \mathbf{MAX\_NCC}_1 \Rightarrow \mathbf{NCC}_1 \geq 2, \quad \mathbf{MIN\_NCC}_2 \neq \mathbf{MAX\_NCC}_2 \Rightarrow \mathbf{NCC}_2 \geq 2$$
$$\mathbf{MIN\_NCC}_1 \neq \mathbf{MAX\_NCC}_1 \Rightarrow \mathbf{NVERTEX}_1 \geq \mathbf{MIN\_NCC}_1 + \mathbf{MAX\_NCC}_1$$
$$\mathbf{MIN\_NCC}_2 \neq \mathbf{MAX\_NCC}_2 \Rightarrow \mathbf{NVERTEX}_2 \geq \mathbf{MIN\_NCC}_2 + \mathbf{MAX\_NCC}_2$$
$$\mathbf{NVERTEX}_1 \leq \mathbf{NCC}_1 \cdot \mathbf{MAX\_NCC}_1, \quad \mathbf{NVERTEX}_2 \leq \mathbf{NCC}_2 \cdot \mathbf{MAX\_NCC}_2$$
$$\mathbf{NVERTEX}_1 \geq \mathbf{MAX\_NCC}_1 + \max(0, \mathbf{NCC}_1 - 1), \quad \mathbf{NVERTEX}_2 \geq \mathbf{MAX\_NCC}_2 + \max(0, \mathbf{NCC}_2 - 1)$$
$$\mathbf{NVERTEX}_1 \geq \mathbf{NCC}_1 \cdot \mathbf{MIN\_NCC}_1, \quad \mathbf{NVERTEX}_2 \geq \mathbf{NCC}_2 \cdot \mathbf{MIN\_NCC}_2$$
$$\mathbf{NVERTEX}_1 \leq \max(0, \mathbf{NCC}_1 - 1) \cdot \mathbf{MAX\_NCC}_1 + \mathbf{MIN\_NCC}_1$$
$$\mathbf{NVERTEX}_2 \leq \max(0, \mathbf{NCC}_2 - 1) \cdot \mathbf{MAX\_NCC}_2 + \mathbf{MIN\_NCC}_2$$
$$\mathbf{NVERTEX}_1 \geq \max(0, \mathbf{NCC}_1 - 1) \cdot \mathbf{MIN\_NCC}_1 + \mathbf{MAX\_NCC}_1$$
$$\mathbf{NVERTEX}_2 \geq \max(0, \mathbf{NCC}_2 - 1) \cdot \mathbf{MIN\_NCC}_2 + \mathbf{MAX\_NCC}_2$$
$$\mathbf{MAX\_NCC}_1 < n \Leftrightarrow \mathbf{NCC}_2 > 0, \quad \mathbf{MAX\_NCC}_2 < n \Leftrightarrow \mathbf{NCC}_1 > 0$$
$$\mathbf{MIN\_NCC}_1 < n \Leftrightarrow \mathbf{NCC}_2 > 0, \quad \mathbf{MIN\_NCC}_2 < n \Leftrightarrow \mathbf{NCC}_1 > 0$$
$$\mathbf{NVERTEX}_1 + \mathbf{NVERTEX}_2 = n$$
$$\max(1, \mathbf{MIN\_NCC}_1) + \max(2, \mathbf{MIN\_NCC}_1 + 1, \mathbf{MAX\_NCC}_1) + \max(1, \mathbf{MIN\_NCC}_2) > n \Rightarrow \mathbf{MIN\_NCC}_1 = \mathbf{MAX\_NCC}_1$$
$$\max(1, \mathbf{MIN\_NCC}_2) + \max(2, \mathbf{MIN\_NCC}_2 + 1, \mathbf{MAX\_NCC}_2) + \max(1, \mathbf{MIN\_NCC}_1) > n \Rightarrow \mathbf{MIN\_NCC}_2 = \mathbf{MAX\_NCC}_2$$

[4] n is the number of variables of the sequence of variables `VARIABLES`

$$\max(1, \mathbf{MIN\_NCC}_1) + \max(1, \mathbf{MAX\_NCC}_1) + \max(1, \mathbf{MIN\_NCC}_2) > n \Rightarrow \mathbf{NCC}_1 \leq 1$$

$$\max(1, \mathbf{MIN\_NCC}_2) + \max(1, \mathbf{MAX\_NCC}_2) + \max(1, \mathbf{MIN\_NCC}_1) > n \Rightarrow \mathbf{NCC}_2 \leq 1$$

$$\mathbf{MIN\_NCC}_1 \cdot \max(0, \mathbf{NCC}_1 - 1) + \mathbf{MAX\_NCC}_1 + \mathbf{MIN\_NCC}_2 \cdot \max(0, \mathbf{NCC}_1 - 2) + \mathbf{MAX\_NCC}_2 \leq n$$

$$\mathbf{MAX\_NCC}_1 \cdot \max(0, \mathbf{NCC}_1 - 1) + \mathbf{MIN\_NCC}_1 + \mathbf{MAX\_NCC}_2 \cdot \mathbf{NCC}_1 + \mathbf{MIN\_NCC}_2 \geq n$$

$$\mathbf{MAX\_NCC}_2 \leq \max(\mathbf{MIN\_NCC}_2, n - \alpha), \texttt{with}:$$
$$\alpha = \mathbf{MIN\_NCC}_1 \cdot \max(0, \mathbf{NCC}_1 - 1) + \mathbf{MAX\_NCC}_1 + \mathbf{MIN\_NCC}_2 + \mathbf{MIN\_NCC}_2 \cdot \max(0, \mathbf{NCC}_1 - 3)$$

$$\mathbf{MIN\_NCC}_2 \cdot \max(0, \mathbf{NCC}_2 - 1) + \mathbf{MAX\_NCC}_2 + \mathbf{MIN\_NCC}_1 \cdot \max(0, \mathbf{NCC}_2 - 2) + \mathbf{MAX\_NCC}_1 \leq n$$

$$\mathbf{MAX\_NCC}_2 \cdot \max(0, \mathbf{NCC}_2 - 1) + \mathbf{MIN\_NCC}_2 + \mathbf{MAX\_NCC}_1 \cdot \mathbf{NCC}_2 + \mathbf{MIN\_NCC}_1 \geq n$$

$$\mathbf{MAX\_NCC}_1 \leq \max(\mathbf{MIN\_NCC}_1, n - \alpha), \texttt{with}:$$
$$\alpha = \mathbf{MIN\_NCC}_2 \cdot \max(0, \mathbf{NCC}_2 - 1) + \mathbf{MAX\_NCC}_2 + \mathbf{MIN\_NCC}_1 + \mathbf{MIN\_NCC}_1 \cdot \max(0, \mathbf{NCC}_2 - 3)$$

Implementing the `change_continuity` *Constraint.* As for the `group` constraint, we came up with one automaton for each graph property. Parts (A), (B), (C) and (D) of Figure 5 respectively depict the automata associated to the graph characteristics $\mathbf{NCC}$, $\mathbf{MIN\_NCC}$, $\mathbf{MAX\_NCC}$ and $\mathbf{NARC}$ of the first graph constraint. Each automaton is applied to the sequence of pairs of consecutive variables of the `VARIABLES` parameter. A transition with a standard (resp. thick) line depicts the fact that $\texttt{VAR}_\texttt{i}$ CTR $\texttt{VAR}_{\texttt{i+1}}$ holds (resp. does not hold). Finally, a transition with a dashed line indicates the end of the sequence of pairs of variables. Since all four automata use counters, we indicate how these counters are initialized in the initial state `s`, how a counter is unified to an argument of the `change_continuity` constraint in the final state `t`, and how they are possibly updated on a given transition. Since the second graph constraint of `change_continuity` is similar to the first graph constraint we don't give the four corresponding automata.

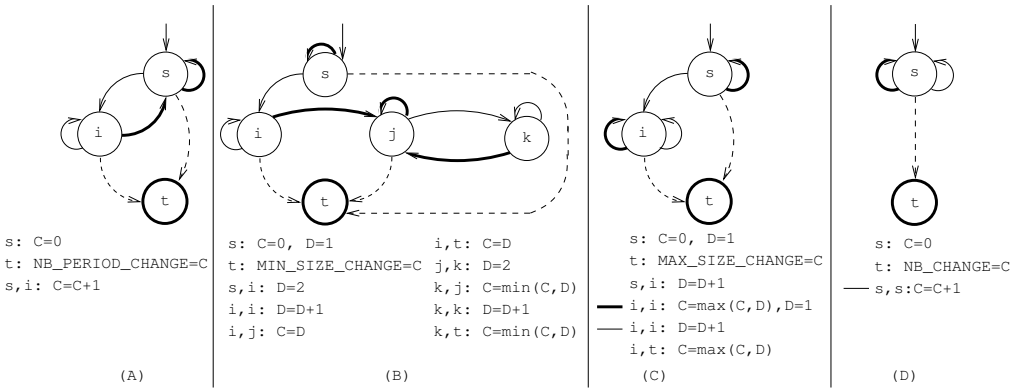

Fig. 5. Automata associated to the graph characteristics of the `change_continuity` constraint

The first approach for implementing the `change_continuity` constraint uses these eight automata. In the second approach we reuse the eight automata and, in addition, extract a set of 32 graph invariants from the database of invariants.

Performance. In order to evaluate the efficiency gained by adding graph invariants, we performed three experiments, generating random instances of the `group` and `change_continuity` constraints. `VARIABLES` was chosen as a sequence of N domains variables ranging over $[0, 1]$, `VALUES` as the singleton set $\{1\}$, and `CTR` as $=$. A constraint instance was generated by setting the initial domain of each domain variable to a randomly chosen interval.

In the first experiment, we computed the total domain size of the domain variables after posting, without invariants vs. with invariants, discarding infeasible instances, for $N = 8$. In the second experiment, we computed the time for posting the constraint instance and searching for all solutions, without invariants vs. with invariants, for $N = 8$. In the third experiment, we computed the time for posting the constraint instance and looking for the first solutions[5], without invariants vs. with invariants, for $N = 100$. Furthermore, with 10% probability, the variables in `VARIABLES` were fixed.

The results are presented in six scatter plots in Figure 6, one row per experiment. Each point represents a random instance, its X coordinate corresponding to excluding the invariants, and its Y coordinate corresponding to including them. The $X = Y$ line is shown in each graph. In the second and third rows, feasible and infeasible instances are denoted differently. Runtimes are in milliseconds.

From these experiments, we observe that the invariants significantly improve the domain reduction including detecting infeasible instances, but that they do not pay off for the purpose of just finding all solutions of feasible instances. However, in a more realistic setting, the improved domain reduction may well lead to savings in search effort that outweigh the overhead of the invariants.

6 Conclusion

The database of graph invariants introduced in this report can be seen as a way to automatically generate necessary conditions for global constraints that can be described in terms of graph properties. In fact, it complements the computation of lower and upper bounds for the graph characteristics presented in [15]. The key advantages of the approach are:

- Instead of developing a specific code for a given global constraint, we come up with graph invariants that can be applied to all global constraints sharing a given graph property.
- The database of graph invariants can be enriched incrementally and systematic experiments can point out missing graph invariants.

Finally, as demonstrated by our experiments on the `group` and the `change_continuity` constraints, it also clearly shows that the graph-based representation and the automaton-based representation of global constraints are not competing approaches for representing the meaning of a global constraint. In fact, when for a given global constraint, both representations are available[6] we can, without developing any specific code, get a filtering algorithm that takes advantage of both representations.

[5] Each constraint instance was run with a 10 seconds time limit.

[6] Out of the 227 constraints of the catalog of global constraints [3], more than 100 global constraints use both representations.

References

1. M. Dincbas, H. Simonis, and P. Van Hentenryck. Solving the car-sequencing problem in constraint logic programming. In Y. Kodratoff, editor, *8th European Conference on Artificial Intelligence, ECAI-88*, pages 290–295, Munich, Germany, August 1988. Pitmann Publishing, London.
2. A. Frisch, I. Miguel, and T. Walsh. Extensions to proof planning for generating implied constraints. In *Proceedings of Calculemus*, 2001.
3. N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global constraint catalog. Technical Report T2005-08, Swedish Institute of Computer Science, 2005.
4. N. Beldiceanu. Pruning for the *minimum* constraint family and for the *number of distinct values* constraint family. In T. Walsh, editor, *Principles and Practice of Constraint Programming (CP'2001)*, volume 2239 of *LNCS*, pages 211–224. Springer-Verlag, 2001. Preprint available as SICS Tech Report T2000-10.
5. C. Bessière, E. Hebrard, B. Hnich, Z. Kızıltan, and T. Walsh. Filtering algorithms for the *nvalue* constraint. In Romand Barták and Michela Milano, editors, *International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'05)*, Lecture Notes in Computer Science, Prague, Czech Republic, may 2005. Springer Verlag.
6. P. Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941. In Hungarian.
7. COSYTEC. *CHIP Reference Manual*, release 5.1 edition, 1997.
8. J.-C. Régin. A filtering algorithm for constraints of difference in CSP. In *12th National Conference on Artificial Intelligence (AAAI-94)*, pages 362–367, 1994.
9. N. Beldiceanu and E. Contejean. Introducing global constraints in CHIP. *Mathl. Comput. Modelling*, 20(12):97–123, 1994.
10. N. Beldiceanu, P. Flener, and X. Lorca. The tree constraint. In Romand Barták and Michela Milano, editors, *International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'05)*, Lecture Notes in Computer Science, Prague, Czech Republic, may 2005. Springer Verlag.
11. Mats Carlsson et al. *SICStus Prolog User's Manual*. Swedish Institute of Computer Science, 3.11.1 edition, February 2004. `http://www.sics.se/sicstus/`.
12. Pascal Van Hentenryck, Vijay Saraswat, and Yves Deville. Constraint processing in cc(FD). Manuscript, 1991.
13. Björn Carlson, Mats Carlsson, and Daniel Diaz. Entailment of finite domain constraints. In P. Van Hentenryck, editor, *ICLP'94, Int. Conf. on Logic Programming*, MIT Press Series in Logic Programming, S. Margherita Ligure, Italy, 1994. The MIT Press.
14. N. Beldiceanu, M. Carlsson, and T. Petit. Deriving filtering algorithms from constraint checkers. In M. Wallace, editor, *Principles and Practice of Constraint Programming (CP'2004)*, volume 3258 of *LNCS*, pages 107–122. Springer-Verlag, 2004.
15. N. Beldiceanu, T. Petit, and G. Rochart. Bornes de caractéristiques de graphes. In *JFPC*, Lance, France, june 2005. In French.

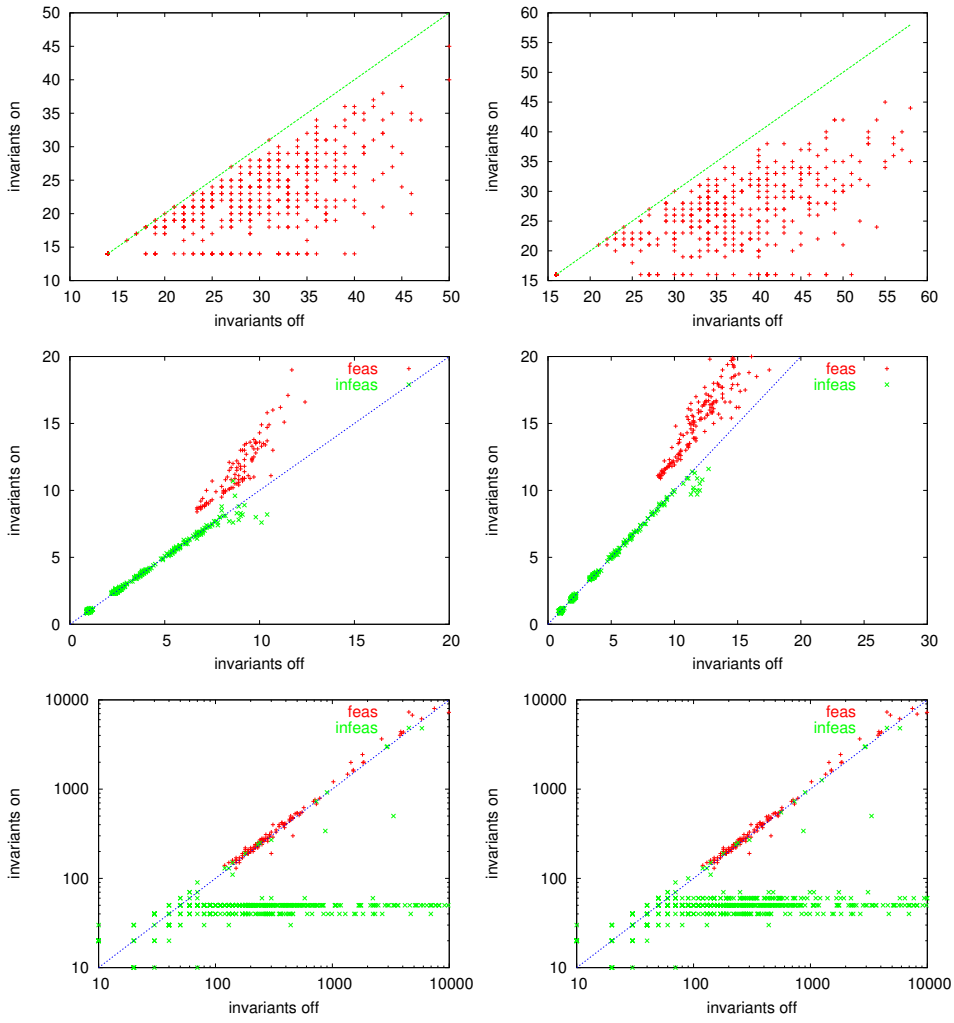

Fig. 6. Scatter plots of random instances. Top: comparing domain sizes. Middle: comparing runtime for finding all solutions. Bottom comparing runtime for finding first solution. Left: `group`. Right: `change_continuity`.