# Positive and Negative Length-Bound Reachability Constraints

**Luis Quesada**
Insight Centre for Data Analytics, School of Computer Science, University College Cork, Ireland

**Kenneth N. Brown**
Insight Centre for Data Analytics, School of Computer Science, University College Cork, Ireland

## Abstract

In many application problems, including physical security and wildlife conservation, infrastructure must be configured to ensure or deny paths between specified locations. We model the problem as sub-graph design subject to constraints on paths and path lengths, and propose length-bound reachability constraints. Although reachability in graphs has been modelled before in constraint programming, the interaction of positive and negative reachability has not been studied in depth. We prove that deciding whether a set of positive and negative reachability constraints are satisfiable is NP complete. We show the effectiveness of our approach on decision problems, and also on optimisation problems. We compare our approach with existing constraint models, and we demonstrate significant improvements in runtime and solution costs, on a new problem set.

**2012 ACM Subject Classification** Theory of computation → Constraint and logic programming

**Keywords and phrases** Reachability Constraints, Graph Connectivity, Constraint Programming

**Digital Object Identifier** 10.4230/LIPIcs.CP.2021.46

**Funding** This publication has emanated from research conducted with the financial support of Science Foundation Ireland under grant numbers 16/SP/3804 and 12/RC/2289-P2, which are co-funded under the European Regional Development Fund.

**Acknowledgements** We thank Seán Óg Murphy, Liam O'Toole and Cormac J. Sreenan for initial discussions on the application that motivated this research, and Jaime Arias for sharing his experience with Visual Studio Code. Finally, we thank the anonymous referees for their help in improving the paper.

## 1 Introduction

Many application problems require reasoning about reachability, including road network design [10], in-building access control [19, 16] and ecosystem management [6]. In each case, paths must be ensured or denied between sets of locations. Often there are further constraints on the lengths of those paths. For example, in buildings, from every location there must exist an accessible path to a fire escape of less than a specified limit [9], while physical security may require protected assets to be kept at least a minimum distance away from unauthorised users [11]. In some cases, solutions may need to be dynamic, responding to movements of either assets, hazards, or users in order to maintain the reachability requirements. In each case, the problem can be considered as sub-graph design, enabling or disabling edges in a larger graph. Some applications have optimisation criteria, including minimising the number of edges (e.g. maintenance cost), or minimising sums of path lengths (e.g. expected travel distance). In this paper, we model the problems in constraint programming, including constraints on reachability and on path length.

Constraint-based graph design for reachability has been studied before [21, 20, 8, 4, 1], including constraints on the costs of paths [22, 5]. However, these papers focus on positive constraints, requiring paths between pairs of nodes. Little attention has been paid to the

interaction of positive and negative constraints. The interaction of the two makes the problem significantly more complex, and we show that determining whether a graph contains a subgraph that satisfies mixed positive and negative reachability constraints is NP-complete. To incorporate the restrictions on path length, we introduce length-bound reachability constraints, and to support large sets of constrained pairs, we implement propagation based on upper and lower bounds on all-pairs shortest paths. We evaluate our constraints on random instances based on road networks, and consider both decision and optimisation problems. We compare to the Dreachable constraint [4], and we show significantly faster runtimes for decision problems and orders of magnitude improvement in costs for time-limited optimisation problems.

In the remainder of the paper, we briefly summarise related work, and we then establish the problem complexity of mixed positive and negative reachability sub-graph design. We describe the length bound reachability constraint, including transitive closure and dominators. Finally, we present the empirical evaluation, showing the behaviour of different versions of the constraints on random problems.

## 2 Related Work

Reachability constraints [20, 4] enforce a graph variable to specify a graph that contains paths between designated vertices. Path constraints [22, 7, 4] enforce a graph variable to represent a path in the graph between two specified vertices, including in some cases bounds on the path lengths [22, 5]. Our interest is in being able to express both positive and negative reachability – that is, to deny connections between some vertex pairs, while enforcing connections between others. Reachability can be expressed in terms of path constraints by having a path constraint for each positive pair, and a negated path constraint for each negative pair. However, having one constraint for each reachability pair leads to redundant computation as the constraints do not share information on their selected paths. For instance, if a path constraint enforcing reachability from $i$ to $j$ discovers that $i$ must reach $k$, the other path constraints cannot take advantage of that knowledge because they only communicate via the decision variables. That is, not only are the space and time complexity of path constraints higher, but the level of pruning achieved is much less than could be expected from a single global constraint that incorporate all positive and negative reachability constraints on the same graph. One-to-many (e.g., [8, 4]) and many-to-many (e.g., [20, 1]) approaches have been proposed to mitigate redundancy and improve pruning when handling several reachability constraints. However, their focus is still on positive reachability constraints rather than on the combination of both positive and negative reachability constraints. We are not aware of any research handling simultaneous positive and negative reachability constraints.

## 3 Positive and Negative Reachability Constraints

In this section we describe the input and the output of the problems we study, and then establish the complexity of the core problem.

### 3.1 Input and Output

- **Input**. We have the following input:
  - A directed graph $G = (V, E)$, where each edge $e \in E$ has an integer cost $e_c$[1].

[1] We use the terms *cost* and *length* interchangeably.

- A set of positive reachability constraints (*PRC*). *A* $prc \in PRC$ *is represented with a tuple* $(i, j, \lambda)$ *meaning that there is at least one path from* $i$ *to* $j$ *in the resulting graph whose cost is less than or equal to* $\lambda$.
- *A set of negative reachability constraints (NRC). An* $nrc \in NRC$ *is represented with a tuple* $(i, j, \lambda)$ *meaning that there is no path from* $i$ *to* $j$ *in the resulting graph whose length is less than* $\lambda$.

In what follows we may omit $\lambda$ in a positive reachability constraints if there is no bound on the length of the path from $i$ to $j$. Similarly, we may omit it in a negative reachability constraint if all paths from $i$ to $j$ are to be denied.

- **Output**. We consider three possible outputs:
  - **Obj 1**. Find $G'$ subgraph of $G$ where all *prcs and nrcs are respected.*
  - **Obj 2**. *Find* $G'$ *subgraph of* $G$ *where all prcs and nrcs are respected, and the sum of the costs of edges of* $G'$ *is minimised.*
  - **Obj 3**. *Find* $G'$ *subgraph of* $G$ *where all prcs and nrcs are respected, and the sum of the costs of the shortest paths in* $G'$ *connecting the vertices in the prcs is minimised.*

## 3.2 Complexity of length-bound reachability constraint problems

In this section section we discuss the complexity of the decision problems involved in length-bound reachability constraint problems.

### 3.2.1 Positive and negative reachability constraints

If we only have positive reachability constraints (i.e., $NRC = \emptyset$), checking whether the set of reachability constraints in *PRC* is satisfiable is straightforward: we just just need to check the existence of a path for every positive reachability constraint. The case where we only have negative reachability constraints is trivial since a totally unconnected graph would satisfy all of them. However, a mix of positive and negative reachability constraints is more challenging. Let us formally define the problem as follows:

▶ **Definition 1** (The Positive and Negative Reachability Constraints problem (*PNRC*)). *Given a directed graph* $G$*, a set of unbounded positive reachability constraints* $PRC$*, and a set of unbounded negative reachability constraints* $NRC$*, is there a sub graph* $G'$ *of* $G$ *that satisfies all constraints in* $PRC$ *and* $NRC$*?*

▶ **Theorem 2.** *PNRC is NP complete*

**Proof.** First, we show PNRC is in NP. We use $G'$ as the certificate, and run Floyd Warshall [3] on it to get the lengths of all-pairs shortest paths. If there is no path between two vertices, it returns $\infty$ as the length. For each $(s, t) \in PRC$, check that the length of their path is less than $\infty$; for each $(p, q) \in NRC$, check that the length is equal to $\infty$. This is polynomial.

We now give a reduction from 3SAT [12] to PNRC. We map a SAT instance to a directed graph (following the approach in [15]) with reachability constraints. We start with a SAT instance $\bigwedge_{i=1}^{n} x_{i1} \vee x_{i2} \vee x_{i3}$, where each $x_{ij}$ is a literal (i.e., it is either a variable or a negation of a variable), and construct a directed graph $G = (V, E)$, where the vertices are associated with levels from 0 to $n + 1$.

1. Create vertices $s, t \in V$. $s$ is the only vertex at level 0. $t$ is the only one at level $n + 1$.
2. Now create a vertex for each literal, add them to V, and assign the vertices for literals of clause $i$ to level $i$. That is, the three vertices of level $i$ are $x_{i1}$, $x_{i2}$, and $x_{i3}$.
3. Add to E a directed edge from $s$ to each vertex of level 1.

4. For each level $i$ from 1 to $n-1$, add to E an edge from each vertex of level $i$ to each vertex of level $i+1$.
5. Add to E a directed edge from each vertex of level $n$ to $t$.

To ensure that the SAT instance is satisfied, we add a positive reachability constraint from $s$ to $t$. A path from $s$ to $t$ represents an assignment of values to the Boolean variables in the literals represented by the vertices in the path. That is, it assigns 1 to the Boolean variable if the literal is positive and 0 otherwise. A path from $s$ to $t$ satisfies all the clauses, and it is consistent if it does not assign two different values to the same variable. To ensure that the assignment to the SAT instance is consistent we add negative reachability constraints as follows. For any two vertices $x_{ij}$ and $x_{kl}$ $(i < k)$ representing literals that negate each other, we must ensure that one is not reachable from each other. Since the directed edges only ascend the levels, we only need to add $(x_{ij}, x_{kl})$ to $NRC$.

We now show that the SAT instance is satisfiable if and only if there is a subgraph of G that satisfies the constraints. If the SAT instance has a solution, then from it pick one TRUE literal in each clause. These literals define a path from s to t in the graph (so satisfies the *prc* in the *PNRC* instance). We select those vertices (literals) and edges as $G'$. There cannot be any conflicting literals selected (since it is a 3SAT solution), and so no *nrc* can be violated, and so the *PNRC* instance has a solution. If the *PNRC* instance has a solution, there is an s-t path. This path has one TRUE literal in each clause. The graph obeys the *nrc*s, and we have not added reachability, so there can be no conflicting literals in the path. Every 3SAT variable not yet determined is then set to 0. This is a solution to the SAT instance.

Finally, we show that the construction is polynomial. If there are $m$ clauses in the 3SAT instance, then there are $n = 3 * m$ occurrences of literals. Each clause is processed in turn, building each layer in the graph, with one vertex per occurrence of a literal. For each vertex, we add incoming edges from the vertices in the previous layer, which is $3 + 3 * (n-3) + 3$ edges. We add one *prc* for $(s, t)$. For the *nrc*s, each time we add a vertex to the graph, we sweep through the previous clauses and their literals. For each previous literal that negates the current one we add an *nrc* between the corresponding previous vertex and the current vertex. That requires $\mathcal{O}(n^2)$ checks and additions of *nrc*s. ◀

### 3.2.2 Positive and negative reachability constraints with bounds on the length of the paths

As the problem is already NP-complete without considering bounds on the lengths of the paths, it follows that it is also NP-complete when considering bounds. Note, however, that what makes the problem hard is the interaction between positive and negative constraints. If $NRC = \emptyset$, the decision problem reduces to checking the lengths of the shortest paths for every pair of vertices in $PRC$. As mentioned in the previous section, we are also interested in minimising the sum of the costs of the selected edges and minimising the sum of the costs of the paths connecting the vertices in $PRC$. Both optimisation problem are clearly NP-Hard as both involve solving decision problems that are NP-complete.

## 4 CP approaches

In this section we present three approaches to model and solve the problem. $G$ refers to the input graph, and $G'$ refers to the resulting graph. In each model we separate the constraints into two groups: essential and redundant. Essential constraints are required for the solution to be sound. Redundant constraints are added to reduce the search space.

## 4.1 Common variables

The models presented have the following common variables:

- We associate a Boolean variable $be_e$ with edge $e$. $be_e = 1$ means $e \in G'$
- Each pair of vertices $(i, j)$ is associated with a Boolean variable $br$. $br_{ij} = 1$ means $i$ reaches $j$ in $G'$
- We have an integer variable $ep_e$ per edge $e$. This variable represents the penalty associated with the edge, which is dependent on the selection of the edge. More precisely, $ep_e$ is either equal to $e_c$ if $e_c \in G'$ or $\infty$ if $e_c \notin G'$.
- Each pair of vertices $(i, j)$ is associated with an integer variable $pc$, which represents the cost of the shortest path from $i$ to $j$. That is, $pc_{ij}$ is either equal to the shortest path cost from $i$ to $j$ in $G'$ or $\infty$ if there is no path in $G'$ going from $i$ to $j$.

## 4.2 The length-bound reachability constraint (LBRC)

All constraints are essential in this approach. This approach relies on the connection between the reachability of vertex $j$ from a vertex $i$ and the shortest path from $i$ to $j$. If there is no path from $i$ to $j$ then we define the length of the shortest path to be $\infty$.

- The $PRC$ constraints are modelled in terms of the $pc$ variables:

$$(i, j, \lambda) \in PRC \Rightarrow pc_{ij} \leq \lambda \tag{1}$$

- The $NRC$ constraints are modelled in terms of the $pc$ variables:

$$(i, j, \lambda) \in NRC \Rightarrow pc_{ij} \geq \lambda \tag{2}$$

- The cost of the shortest path to a reachable node is less than $\infty$:

$$br_{ij} = 1 \Leftrightarrow pc_{ij} < \infty \tag{3}$$

- The penalty of an edge $e$ is either the cost of the edge, if the edge is selected, or $\infty$:

$$be_{ij} = 1 \Leftrightarrow ep_e = e_c \tag{4}$$

- The cost of the shortest path from $i$ to $j$ must be the edge $(i, j)$ or, for some in-neighbour $x$ of $j$, a shortest path from $i$ to $x$ followed by the edge $(x, j)$:

$$pc_{ij} = \min(\{ep_{\langle i,j\rangle}\} \cup \{pc_{ix} + ep_{\langle x,j\rangle} | x \in in(j)\}) \tag{5}$$

where $in(j)$ refers to the incoming neighbours of $j$, and for each pair $(i, j)$, we have $1 + |in(j)|$ cost variables. This maintain bounds on the costs of the shortest paths, and as edges are denied or enforced, the bounds are updated and propagated.

We define the length-bound reachability constraint (LBRC) as the constraint that keeps variables $be$, $ep$, $pc$, and $br$ consistent in the way described above. More formally,

▶ **Definition 3.**

$$\begin{aligned} LBRC(be, ep, pc, br) \quad \equiv \quad & (i, j, \lambda) \in PRC \Rightarrow pc_{ij} < \lambda \ \wedge \\ & (i, j, \lambda) \in NRC \Rightarrow pc_{ij} \geq \lambda \ \wedge \\ & br_{ij} = 1 \Leftrightarrow pc_{ij} < \infty \ \wedge \\ & be_{ij} = 1 \Leftrightarrow ep_e = e_c \ \wedge \\ & pc_{ij} = \min(\{ep_{\langle i,j\rangle}\} \cup \{pc_{ix} + ep_{\langle x,j\rangle} | x \in V\}) \end{aligned}$$

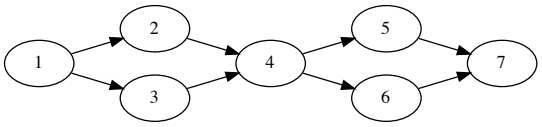



**Figure 1** A simple directed graph with vertex 4 dominating vertex 7 on paths from vertex 1.

## 4.3 The length-bound reachability constraint with Transitive closure (LBRC+TC)

Consider the graph in Figure 1, and assume that we have:

$$br_{17} = 0 \wedge br_{14} = 1 \wedge br_{47} = 1$$

These reachability constraints are unsatisfiable. However we cannot detect that unsatisfaction by pure propagation with LBRC. Notice that:

$$\begin{array}{lll} br_{14} = 1 & \Rightarrow & pc_{14} < \infty \\ & \Rightarrow & (ep_{12} + ep_{24} < \infty) \vee (ep_{13} + ep_{34} < \infty) \end{array}$$

Similarly, we have that:

$$\begin{array}{lll} br_{47} = 1 & \Rightarrow & pc_{47} < \infty \\ & \Rightarrow & (ep_{45} + ep_{57} < \infty) \vee (ep_{46} + ep_{67} < \infty) \end{array}$$

And we also have that:

$$\begin{array}{lll} br_{17} = 0 & \Rightarrow & pc_{17} = \infty \\ & \Rightarrow & pc_{15} + ep_{57} = \infty \\ & \Rightarrow & pc_{16} + ep_{67} = \infty \end{array}$$

However, we cannot go any further since both $pc_{15}$ and $pe_{57}$ can be set to $\infty$ (i.e., we have a disjunction). We have the same situation with $pc_{16}$ and $pe_{67}$. The propagators behind these sum constraints will just wait until one of the variables become less than $\infty$ to set the other variable to $\infty$, or until one of the variables becomes $\infty$ to declare entailment.

In order to address this lack of propagation, in addition to the essential constraints of LBRC we add redundant constraints implementing the transitive closure of the output graph:

$$br_{ij} = 1 \wedge br_{jk} = 1 \Rightarrow br_{ik} = 1 \tag{6}$$

Taking into account these redundant constraints, we have that $br_{17} = 0$ implies $br_{14} = 0 \vee br_{47} = 0$, which is in direct contradiction with $br_{14} = 1 \wedge br_{47} = 1$.

## 4.4 The length-bound reachability constraint with Dominators (LBRC+Dom)

Consider again the graph in Figure 1. Suppose now that we have:

$$br_{17} = 1 \wedge br_{14} = 0$$

These two constraints are clearly unsatisfiable given the structure of the graph. However, we cannot detect that unsatisfiability just by pure propagation with LBRC+TC. To do that we need to take into account that all paths from vertex 1 to vertex 7 go through vertex 4.

In this approach we use a constraint from [20], which relies on the notion of dominators:

▶ **Definition 4.** *Given a directed graph $G = (V, E)$, and vertices $i, j, k \in V$, $j$ is a dominator of $k$ with respect to $i$ if all paths from $i$ to $k$ in $G$ go through $j$*

For Figure 1 we see that vertex 4 is a dominator of vertex 7 with respect to vertex 1.

▶ **Definition 5.** *The DomReach$(be, dom, br)$ constraint holds iff $br$ represents the transitive closure of $G'$,the graph represented by $be$, and $dom$ is a 3D array representing the dominators of $G'$, i.e., $dom_{ijk} = 1$ iff $j$ is a dominator of $k$ with respect to $i$ in $G'$.*

We implemented the *DomReach* constraint following the same ideas of [20], with two main differences. First, we omit the pruning rules associated with the relation between the graph and its transitive closure, as we achieve this through the implementation of Equation 5. Second, we maintain dominators from all sources. In [20], the focus is on computing a single path with mandatory nodes, but in our case reachability constraints could involve all possible sources, which justify maintaining dominators from all sources.

We replace Constraint 6 in LBRC+TC with the following constraints:

- One *DomReach* constraint to enforce the transitive closure and the dominator relation:

$$DomReach(be, dom, br) \tag{7}$$

- For all $i, j, k$:

$$(dom_{ijk} = 1 \wedge br_{ik} = 1) \Rightarrow (br_{ij} = 1 \wedge br_{jk} = 1) \tag{8}$$

With these redundant constraints, we have that $br_{17} = 1$ and $dom_{147} = 1$ implies $br_{14} = 1$, which contradicts $br_{14} = 0$.

## 5 Dreachable approach

We can also model unbounded positive reachability constraints using *Dreachable* [4].

▶ **Definition 6.** *The Dreachable$(G, s, G^*)$ constraint holds iff $G^*$ is a subgraph of $G$ such that all vertices and edges of $G^*$ are reachable from $s$ in $G^*$.*

- The (unbounded) *PRC* constraints are modelled in terms of the $br$ variables:

$$(i, j, \_) \in PRC \Rightarrow br_{ij} = 1 \tag{9}$$

- The (unbounded) *NRC* constraints are modelled in terms of the $br$ variables:

$$(i, j, \_) \in NRC \Rightarrow br_{ij} = 0 \tag{10}$$

- One *Dreachable* constraint per source in *PRC* is posted:

$$\forall i \in sources(PRC) : Dreachable(G, i, G^i) \tag{11}$$

  where $G^i$ is equal to the projection of $G'$ (the output graph) on the vertices that are reachable from $i$, i.e., $G'[\{j | br_{ij} = 1\}]$.
- The transitive closured is enforced:

$$br_{ij} = 1 \wedge br_{jk} = 1 \Rightarrow br_{ik} = 1 \tag{12}$$

  Note that in this model the transitive closure constraints are essential to ensure that the negative reachability constraints are respected.

In [4] it is stated that dominators are used in *Dreachable* to deal with cases like the one in Section 4.4, so this approach should achieve the same level as pruning of LBRC+Dom.

## 6 Empirical Evaluation

We performed our experiments on machines with Intel(R) Xeon(R) CPU with 2.40GHz running on Ubuntu 18.04. The version of Minizinc [17] used in the experiments is 2.5.3, which comes with Gecode [13] 6.3.0 and Chuffed [2] 0.10.4. LBRC, LBRC+TC and LBRC+Dom were implemented directly in Gecode 6.2.0. However, LBRC was also implemented in Minizinc to be able to compare the same model in both Chuffed and Gecode.

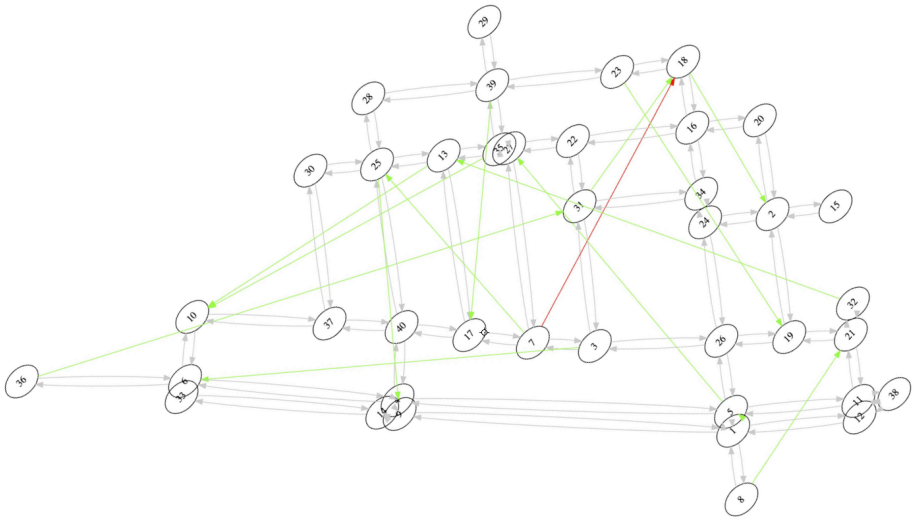


**Figure 2** An instance of 40 vertices, 14 positive reachability constraints (coloured in green) and 1 negative reachability constraint (coloured in red).

### 6.1 Instances

The graphs considered in the evaluation are planar graphs extracted from a real-world road network generated using the GIS-F2E tool [14]. The generated road network has 137626 nodes and 194996 (undirected) links. From this network we randomly select subgraphs by choosing a random node and running Breadth First Search from that node, stopping the search when reaching the specified number of nodes for the subgraph. The approaches evaluated are based on directed graphs so the undirected graphs are converted to directed graphs by adding symmetric edges. One instance is shown in Figure 2. In what follows we use $\mathcal{G}$ to refer to the directed version of the generated road network.

The instances are characterised in terms of the following features:

- *size*: the number of nodes of the graph. The set of sizes considered is $\{20, 24, 28, 32, 36, 40\}$. We randomly select a subgraph of *size* vertices from $\mathcal{G}$. For each size, we generate 10 graphs for the experiments in Figures 3, 4 and 5, and 100 for the other experiments.
- *Cs*: the percentage of selected constraints. For each graph we randomly select *Cs*% of the possible pairs. Each pair denotes a positive or negative reachability constraint.
- $(pos, neg)$: the ratio of positive and negative reachability constraints. For each set of selected reachability constraints, *pos*% are labelled as positive and *neg*% as negative.
- $(pb, nb)$: the bounds on the positive and negative reachability constraints. Let *maxp* be the maximum of the lengths of the shortest paths between the positive reachability pairs. All positive reachability constraints are subject to an upper bound of $maxp \times (1 + pb/100)$. *maxn* is the maximum of the lengths of the shortest paths between the negative reachability pairs, and all negative reachabililty constraints have a lower bound of $maxn \times (1 + nb/100)$.

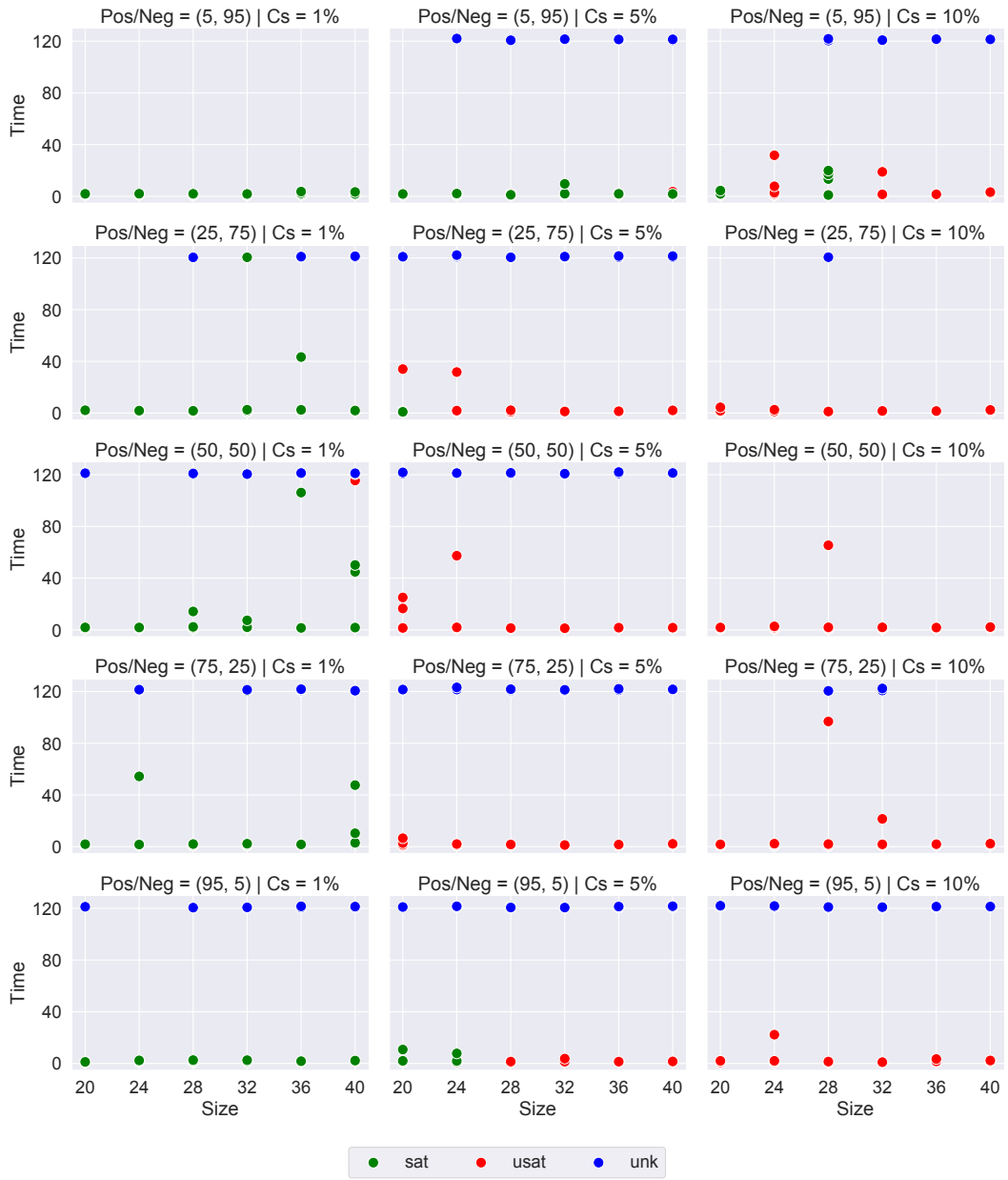



**Figure 3** Performance of LBRC in Gecode. Instances are classified as satisfiable (sat), unsatisfiable (usat), and unknown (unk).

For example, consider the case where $size = 40$, $Cs = 1$, and $(pos, neg) = (50, 50)$, with no length bounds. There are $40 \times 39 = 1560$ possible pairs. We randomly choose 1% of the pairs, which amounts to 16 (after rounding up). Of these 16 pairs, 8 (50%) are labelled as positive reachability constraints, and 8 are labelled as negative reachability constraints.

The first set of experiments, whose results are shown in Figures 3 and 4, correspond to using LBRC on all the instances with both Gecode and Chuffed via Minizinc. We were interested in real-time solutions for our application, therefore we set the objective to the minimisation of path lengths (Obj 3), and the timeout to 120 seconds. These experiments let us assess the role that the features described play in the difficulty of solving the instances. A high $Cs$ value usually led to trivially unsatisfiable instances, so we focused on small values.

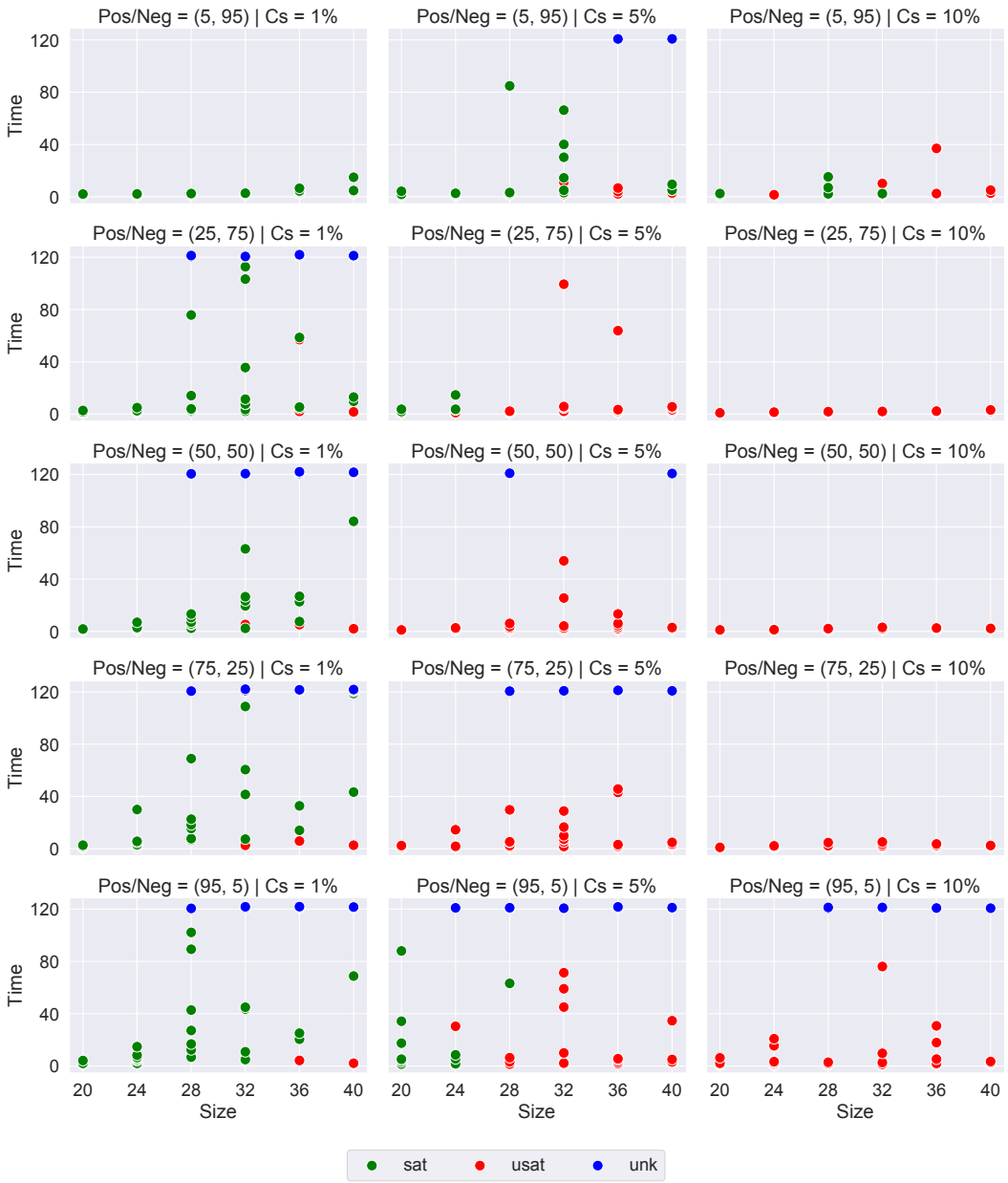
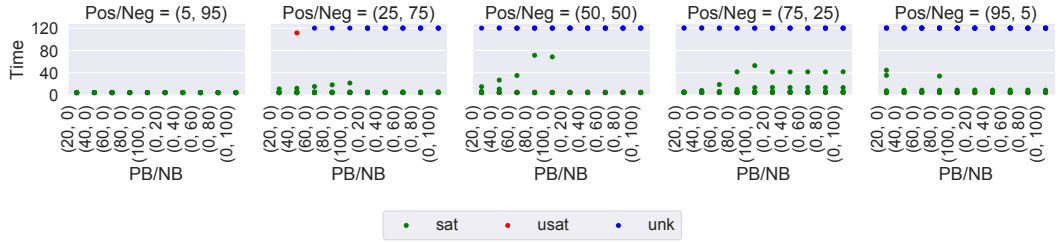


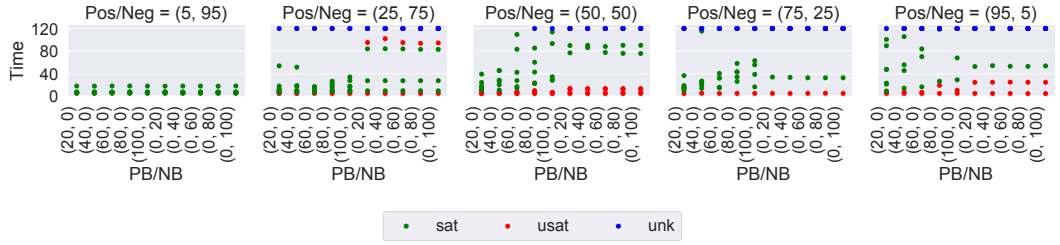

**Figure 4** Performance of LBRC in Chuffed. Instances are classified as satisfiable (sat), unsatisfiable (usat), and unknown (unk).

The behaviour of Gecode was extreme: in most cases either it solved the instances very quickly or did not solve them at all. In particular, unsatisfiable instances are challenging for Gecode. Chuffed was better at dealing with the unsatisfiable instances, but its performance was inferior when dealing with satisfiable cases.

We note that a small number of negative reachability constraints are enough to create challenging instances. For example, when $size = 40$, $Cs = 1$, and $(pos, neg) = (95, 5)$, there is only one negative reachability constraint, but most of the instances in this class remained unsolved. We focus on this case in further experiments.




**(a)** Gecode.




**(b)** Chuffed.

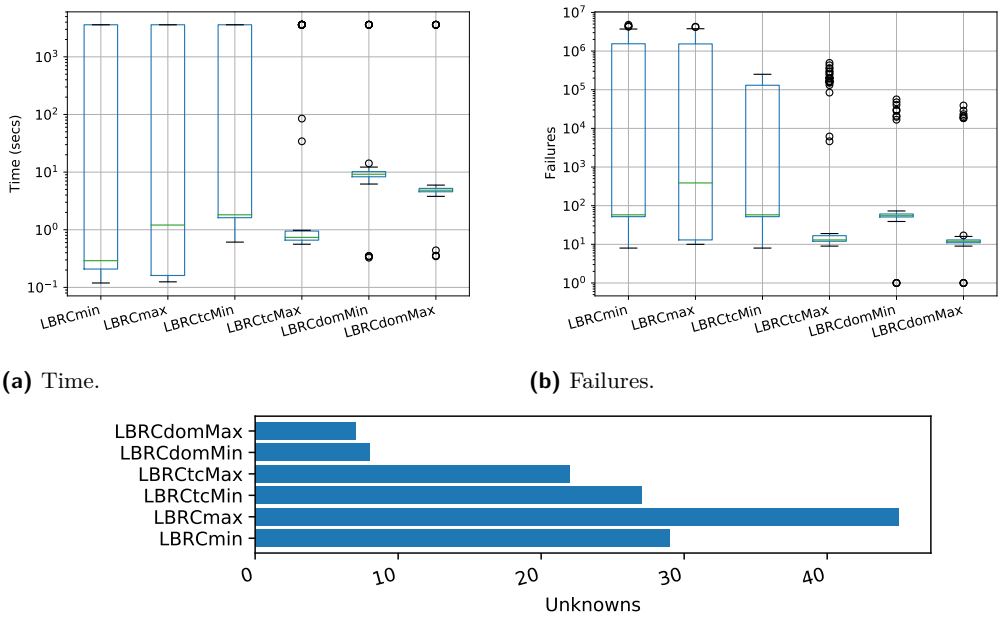
**Figure 5** Performance of LBRC on length-bound instances. Instances are classified as satisfiable (sat), unsatisfiable (usat), and unknown (unk).

In Figure 5 we show the impact of the path-length bounds on the runtime for the instances of $size = 40$, $Cs = 1$, and $(pos, neg) \in \{(5, 95), (25, 75), (50, 50), (75, 25), (95, 5)\}$. For both $pb$ and $nb$ we consider values in $\{0, 20, 40, 60, 80, 100\}$, where $pb = 0$ means that the positive constraint is unbounded, and $nb = 0$ means that all paths are denied. While there is not much change when varying $nb$, we observe that the instances tend to get harder when increasing $pb$.

## 6.2 Performance of the different LBRC versions

In this section we consider the different versions of the length-bound reachability constraint: LBRC, LBRC+TC, and LBRC+Dom. We created 100 graphs of 40 vertices, and for each graph we generated an instance of this class (i.e., where $Cs = 1$, and $(pos, neg) = (95, 5)$) without length bounds. Figure 2 shows one of these instances. As the purpose is to assess the additional pruning obtained by the use of explicit transitive closure and dominators, we focus on the decision problem. The decision variables are the $be$ variables as the determination of these variables fully determines the other variables. For each of the approaches we are considering two variable orderings: setting the variable to its minimum value in the domain first ($min$), and setting the variable to its maximum value in the domain first ($max$). Notice that $min$ corresponds to excluding the corresponding edge from the set of edges of the output graph and $max$ to adding it to the set. In what follows we refer to the approaches obtained when considering the variable orderings as LBRCmin, LBRCmax, LBRCtcMin, LBRCtcMax, LBRCdomMin and LBRCdomMax.

The results of the tests are shown in Figure 6, where Figure 6a refers to the running time, Figure 6b to the number of failures, and Figure 6c to the number of instances that the approach could not solve. The first thing to remark is the gain in pruning when we use LBRC+TC and LBRC+Dom, which positively affects the running time. We also observe that LBRC and LBRC+TC are more sensitive to the variable ordering than LBRC+Dom. Better results are observed with $max$ when using LBRC+TC and LBRC+Dom. We believe




**(a)** Time.

**(b)** Failures.

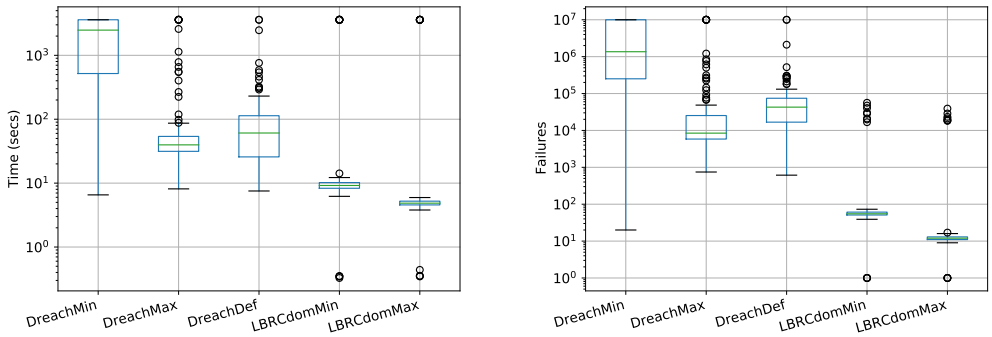
**(c)** Number of unsolved instances out of the 100 instances tried.

**Figure 6** Comparing different versions of the length-bound reachability constraint. The box plots in Figures 6a and 6b show median, inter-quartile range (IQR), bounds of +/− 1.5*IQR beyond the box, and outliers, using the DataFrame.boxplot function of Pandas[18].

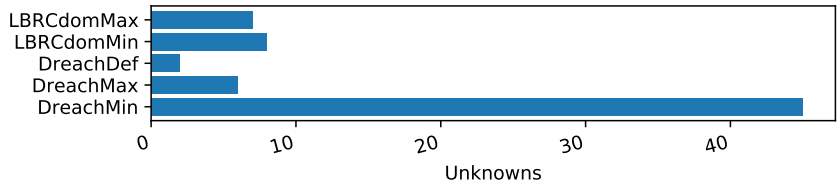
this is because there is more opportunity for the transitive closure rule to play a role as we have more edges. As LBRC does not have the transitive closure pruning, adding edges first actually leads to poorer performance as there are more chances of getting trapped in unsatisfiable cases that are easily detectable by the transitive closure rule. When comparing the performance of the approaches using *min* we see that LBRCtcMin is almost as good as LBRCmin and LBRCdomMin performs much better than both of them, in particular if we look at the number of unsolved instances (see Figure 6c). The advantage of reasoning about dominators is clear, since vertices may become dominators when we take the decision of removing an edge. As discussed in the next section, *min* is useful when optimising the sum of the cost of the edges of the output graph, so it is important to perform well with *min*. Similarly, *max* brings us closer to the optimal solution when minimising the sum of the costs of the paths since the more edges the more chances to connect vertices through the shortest paths. On the easy instances, LBRC perform better than the other two approaches since the overhead of the transitive closure and dominator pruning is not justified. As LBRC+Dom is the best version of the length-bound reachability constraint, we restrict attention to this version for the remaining experiments.

## 6.3 LBRC+Dom vs Dreachable

We now compare our LBRC+Dom approach against the Dreachable approach of Section 5.




**(a)** Time. **(b)** Failures.




**(c)** Number of unsolved instances out of the 100 instances tried.

**Figure 7** LBRC+Dom vs Dreachable - Decision benchmarks on positive and negative reachability constraints.

### 6.3.1 Positive and Negative Reachability Constraints

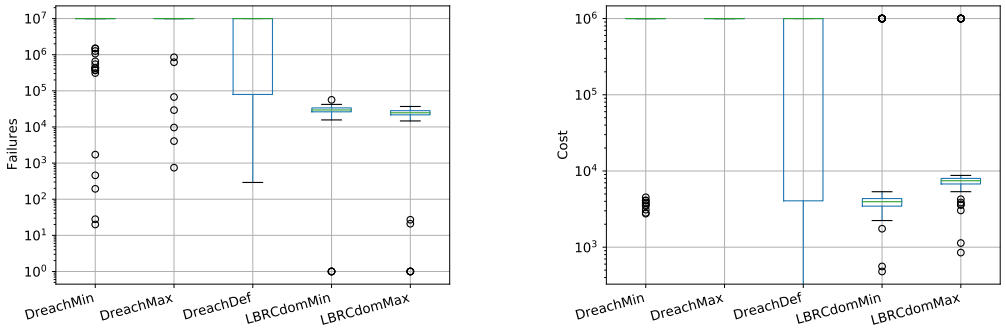
We revisit the instances from Section 6.2. We consider both the decision problem (see Figure 7) and the minimisation of the sum of the costs of the edges of the output graph (see Figure 8). In these experiments we also considered the default search strategy for Chuffed (DreachDef), which is not documented, but led to better results.

The results of the decision problem show that our LBRC+Dom approach is able to find solution faster because we are failing much less. Even though [4] states that dominators are used in the implementation of the Dreachable constraints, it is not clear that is happening in the version provided by Minizinc. Still it is important to mention that DreachMax did manage to close slightly more instances than both LBRCdomMin and LBRCdomMax (see Figure 7c).

The results of the optimisation problem follow the same trend observed in the decision case with respect to the number of failures. However, both approaches ended timing out in most of the cases. Despite that, LBRC+Dom approach managed to get better costs in general. This was mostly due to the good performance observed when using *min*, which tends to minimise the sum of the costs of the edges by selecting fewer edges. It is important to note that in Chuffed we are using both restarts and nogood learning, while in Gecode we have disabled those options. We expect to improve our results even further when considering these options in Gecode.

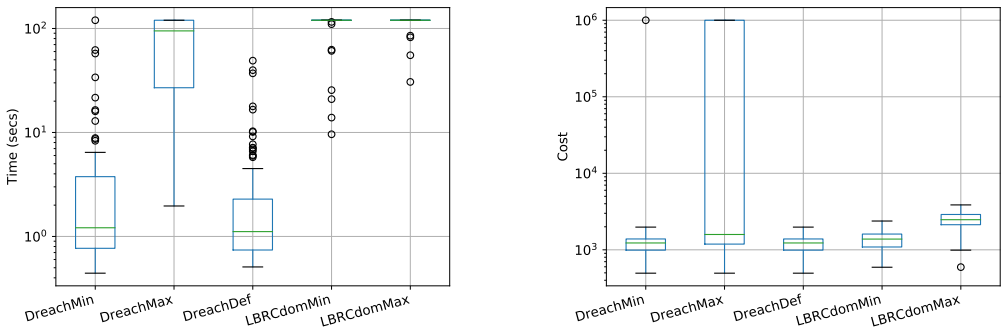
### 6.3.2 Positive Reachability Constraints

As mentioned before, if there is no negative reachability constraint, satisfying a set of positive reachability constraints is straightforward. However, if there is an upper bound on the sum of the costs of the selected edges, the problem is NP-complete [1]. This makes the corresponding optimisation problem, i.e., satisfying the set of positive reachability constraints while minimising the sum of the costs of the selected edges, challenging.




**(a)** Failures. **(b)** Costs.

**Figure 8** LBRC+Dom vs Dreachable – Optimisation benchmarks on positive and negative reachability constraints.




**(a)** Time. **(b)** Costs.

**Figure 9** LBRC+Dom vs Dreachable – Optimisation benchmarks on positive reachability constraints.

We have created a new set of instances following the same procedure as above, except instead of randomly selecting a set of pairs, we follow the approach of [1] and randomly select a subset of vertices that are to be fully connected, i.e., for each pair of vertices in this set there should be a path in both directions. Figure 9 shows the results for 100 graphs of 20 vertices. In each case we randomly selected 8 vertices to be fully connected.

The LBRC+Dom approach is outperformed by the Dreachable approach in these instances. Not only does it prove optimality for most of the instances, but it has significantly lower runtime. The main issue with the LBRC+Dom approach has to do with proving optimality. As it can be observed in Figure 9b, the costs obtained by the LBRC approach are very close to the ones obtained by the Dreachable approach but it spends most of the time trying to prove optimality.

## 7 Conclusions and Future Work

Many practical applications impose positive and negative constraints on reachability, further enhanced with upper and lower bound on the minimum cost paths between pairs of nodes. We have shown that the interaction between positive and negative reachability constraints

leads to complex problems. We have proposed three approaches to modelling these problem as a global constraint on graph variables, which differ on the level of pruning achieved, and demonstrated empirically that the additional pruning plays a key role in solving the problems. We have also studied the dependency between the level of pruning and the search strategy and concluded that the convenience of the search strategy depends on the level of pruning. We have compared our best approach with an existing state of the art approach and shown that when both positive and negative reachability constraints are present, our best approach, incorporating propagation on the transitive closure and dominators, allows significantly lower runtimes, and significantly lower costs for time-limited solving. On the other hand, for problems with only positive reachability constraints, the existing Dreachable constraint is significantly faster.

We believe the improvement offered by the new constraint can be increased by incorporating nogood learning techniques and restarts. Our primary focus in this paper has been on the interaction of positive and negative reachability constraints. Future work will focus on pruning rules to get tighter bounds for the cost, to make the new constraint more competitive in cases where the complexity is driven by the bound on the cost.

## References


1 Christian Bessiere, Emmanuel Hebrard, George Katsirelos, and Toby Walsh. Reasoning about connectivity constraints. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2568–2574. AAAI Press, 2015. URL: `http://ijcai.org/Abstract/15/364`.

2 Geoffrey Chu. *Improving combinatorial optimization*. PhD thesis, University of Melbourne, Australia, 2011. URL: `http://hdl.handle.net/11343/36679`.

3 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

4 Diego de Uña. *Discrete optimization over graph problems*. PhD thesis, University of Melbourne, Parkville, Victoria, Australia, 2018. URL: `http://hdl.handle.net/11343/217321`.

5 Diego de Uña, Graeme Gange, Peter Schachte, and Peter J. Stuckey. A bounded path propagator on directed graphs. In Michel Rueher, editor, *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 189–206. Springer, 2016. `doi:10.1007/978-3-319-44953-1_13`.

6 Bistra Dilkina and Carla P. Gomes. Solving connected subgraph problems in wildlife conservation. In Andrea Lodi, Michela Milano, and Paolo Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010, Bologna, Italy, June 14-18, 2010. Proceedings*, volume 6140 of *Lecture Notes in Computer Science*, pages 102–116. Springer, 2010. `doi:10.1007/978-3-642-13520-0_14`.

7 Grégoire Dooms. *The CP(Graph) computation domain in constraint programming*. PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2006. URL: `http://hdl.handle.net/2078.1/107275`.

8 Jean-Guillaume Fages and Xavier Lorca. Revisiting the tree constraint. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, volume 6876 of *Lecture Notes in Computer Science*, pages 271–285. Springer, 2011. `doi:10.1007/978-3-642-23786-7_22`.

9 Department for Communities and Local Government. *Fire safety risk assessment: offices and shops*. The Stationery Office, UK, 2006.

10 Markus Friedrich. Functional structuring of road networks. *Transportation Research Procedia*, 25:568–581, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016. doi:10.1016/j.trpro.2017.05.439.

11 M.L. Garcia. *Design and Evaluation of Physical Protection Systems*. Elsevier Science, 2007.

12 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

13 Gecode Team. Gecode: Generic constraint development environment, 2019. Available from http://www.gecode.org.

14 Alireza Karduni, Amirhassan Kermanshah, and Sybil Derrible. A protocol to convert spatial polyline data to network formats and applications to world urban road networks. *Scientific Data*, 3(1):160046, 2016. doi:10.1038/sdata.2016.46.

15 Petr Kolman and Ondrej Pangrác. On the complexity of paths avoiding forbidden pairs. *Discret. Appl. Math.*, 157(13):2871–2876, 2009. doi:10.1016/j.dam.2009.03.018.

16 Seán Óg Murphy, Liam O'Toole, Luis Quesada, Kenneth N. Brown, and Cormac J. Sreenan. Ambient access control for smart spaces: dynamic guidance and zone configuration. In *The 12th International Conference on Ambient Systems, Networks and Technologies (ANT), March 23-26, 2021, Warsaw, Poland*, pages 330–337, 2021.

17 Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741, pages 529–543. Springer, 2007. doi:10.1007/978-3-540-74970-7_38.

18 The pandas development team. pandas-dev/pandas: Pandas, 2020. doi:10.5281/zenodo.3509134.

19 Liliana Pasquale, Carlo Ghezzi, Edoardo Pasi, Christos Tsigkanos, Menouer Boubekeur, Blanca Florentino-Liaño, Tarik Hadzic, and Bashar Nuseibeh. Topology-aware access control of smart spaces. *Computer*, 50(7):54–63, 2017.

20 Luis Quesada. *Solving Constrained Graph Problems using Reachability Constraints based on Transitive Closure and Dominators*. PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2006.

21 Luis Quesada, Peter Van Roy, Yves Deville, and Raphaël Collet. Using dominators for solving constrained path problems. In Pascal Van Hentenryck, editor, *Practical Aspects of Declarative Languages, 8th International Symposium, PADL 2006, Charleston, SC, USA, January 9-10, 2006, Proceedings*, volume 3819 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2006. doi:10.1007/11603023_6.

22 Meinolf Sellmann. Cost-based filtering for shorter path constraints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 694–708. Springer, 2003. doi:10.1007/978-3-540-45193-8_47.