The Weighted Arborescence Constraint

Vinasetan Ratheil Houndji[1,2], Pierre Schaus[1], Mahouton Norbert Hounkonnou[2] and Laurence Wolsey[1]

[1]Université catholique de Louvain, Louvain-la-Neuve, Belgium
[2]Université d'Abomey-Calavi, Abomey-Calavi, Benin
{vinasetan.houndji, pierre.schaus, laurence.wolsey}@uclouvain.be
{ratheil.houndji}@ifri.uac.bj and {norbert.hounkounnou}@cipma.uac.bj

Abstract. For a directed graph, a Minimum Weight Arborescence (MWA) rooted at a vertex r is a directed spanning tree rooted at r with the minimum total weight. We define the `MinArborescence` constraint to solve constrained arborescence problems (CAP) in Constraint Programming (CP). A filtering based on the LP reduced costs requires $O(|V|^2)$ where $|V|$ is the number of vertices. We propose a procedure to strengthen the quality of the LP reduced costs in some cases, also running in $O(|V|^2)$. Computational results on a variant of CAP show that the additional filtering provided by the constraint reduces the size of the search tree substantially.

1 Introduction

In graph theory, the problem of finding a Minimum Spanning Tree (MST) [13] is one of the most known problem for undirected graphs. The corresponding version for directed graphs is called Minimum Directed Spanning Tree (MDST) or Minimum Weight Arborescence (MWA). It is well known that the graphs are good structures to model some real life problems. The MWA problem has many practical applications in telecommunication networks, computer networks, transportation problems, scheduling problems, etc. It can also be considered as a subproblem in many routing and scheduling problems [9]. For example, Fischetti and Toth ([8]) used MWA problem as a relaxation of Asymmetric Travelling Salesman Problem (ATSP).

Let us formally define the MWA problem. Consider a directed graph $G = (V, E)$ in which $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set and $E = \{(i, j) : i, j \in V\}$ is the edge set. We associate a weight $w(i, j)$ to each edge (i, j) and we distinguish one vertex $r \in V$ as a root. An arborescence A rooted at r is a directed spanning tree rooted at r. So A is a spanning tree of G if we ignore the direction of edges and there is a directed path in A from r to each other vertex $v \in V$. An MWA $A(G)^\star$ of the graph G is an arborescence with the minimum total cost. Without loss of generality, we can remove any edge entering in the root r. Consider a subset of vertices $S \subseteq V$. Let δ_S^{in} be the set of edges entering S: $\delta_S^{in} = \{(i, j) \in E : (i \in V \setminus S) \wedge (j \in S)\}$. For a vertex $k \in V$, δ_k^{in} is the set of edges that enter in k. Let V' be the set of vertices without the root r: $V' = V \setminus \{r\}$. The MWA problem can be formulated as follows [10]:

$$\text{(MWA)} \quad \begin{aligned} w(A(G)^{\star}) &= \min \sum_{(i,j)\in E} w(i,j)\cdot x_{i,j} && (1)\\ &\sum_{(i,j)\in \delta_j^{in}} x_{i,j} = 1, \forall j \in V' && (2)\\ &\sum_{(i,j)\in \delta_S^{in}} x_{i,j} \geq 1, \forall S \subseteq V' : |S| \geq 2 && (3)\\ &x_{i,j} \in \{0,1\}, \forall (i,j) \in E && (4) \end{aligned}$$

in which $x_{i,j} = 1$ if the edge (i,j) is in the optimal arborescence $A(G)^{\star}$ and $x_{i,j} = 0$ otherwise. The first group of constraints imposes that exactly one edge enters in each vertex $j \in V'$ and the constraints (3) enforce the existence of a path from the root r to all other vertices. Without loss of generality [9], we assume that $w(i,i) = \infty, \forall i \in V$ and $w(i,j) > 0, \forall (i,j) \in E$. Then the constraints (4) can be relaxed to $x_{i,j} \geq 0, \forall i,j(5)$ and the constraints (2) become redundant (see [6]).

Here, we address the Constrained Aborescence Problem (CAP) - that requires one to find an arborescence that satisfies other side constraints and is of minimum cost - and show how one can handle them in CP. After some theoretical preliminaries, we define the `MinArborescence` constraint and show how to filter the decision variables. Then we describe how Linear Programming (LP) reduced costs can be improved. Finally, we show some experimental results and conclude.

2 Background

Algorithms to compute an MWA $A(G)^{\star}$ of a given graph G were proposed independently by Chu and Liu ([3]), Edmonds ([6]) and Bock ([2]). A basic implementation of that algorithm is in $O(|V||E|)$. The associated algorithm is often called Edmonds' Algorithm. An $O(\min\{|V|^2, |E|\log|V|\})$ implementation of the Edmonds' algorithm is proposed by [23]. More sophisticated implementations exist (see for example [12,19]). Fischetti and Toth [9] proposed an $O(|V|^2)$ implementation to compute an MWA and also the associated linear programming reduced costs. We rely on this algorithm for filtering the `MinArborescence` constraint.

An MWA has two important properties that are used to construct it [16].

Proposition 1. *A subgraph $A = (V,F)$ of the graph G is an arborescence rooted at r if and only if A has no cycle, and for each vertex $v \neq r$, there is exactly one edge in F that enters v.*

Proposition 2. *For each $v \neq r$, select the cheapest edge entering v (breaking ties arbitrarily), and let $F^{\star}$ be this set of $n-1$ edges. If $(V,F^{\star})$ is an arborescence, then it is a minimum cost arborescence, otherwise $w(V,F^{\star})$ is a lower bound on the minimum cost arborescence.*

The LP dual problem $\mathcal{D}_{\text{MWA}}$ of MWA is [9]:

$$(\mathcal{D}_{\text{MWA}}) \quad \begin{aligned} &\max \sum_{S \subseteq V'} u_S \\ &w(i,j) - \sum_{(i,j) \in \delta_S^{in}, \forall S \subseteq V'} u_S \geq 0, \forall (i,j) \in E \\ &u_S \geq 0, \forall S \subseteq V' \end{aligned}$$

in which u_S is the dual variable associated to the subset of edges $S \subseteq V'$.

Let $rc(i,j)$ be the LP reduced cost associated to the edge (i,j). The necessary and sufficient conditions for the optimality of MWA (with primal solution $x^\star_{i,j}$) and $\mathcal{D}_{\text{MWA}}$ (with dual solution $u^\star_S$) are [9]:

1. primal solution $x^\star_{i,j}$ satisfies the constraints (3) and (5)
2. $u^\star_S \geq 0$ for each $S \subseteq V'$
3. reduced cost $rc(i,j) = w(i,j) - \sum_{(i,j) \in \delta_S^{in}, \forall S \subseteq V'} u^\star_S \geq 0$ for each $(i,j) \in E$
4. $rc(i,j) = 0$ for each $(i,j) \in E$ such that $x^\star_{i,j} > 0$
5. $\sum_{(i,j) \in \delta_S^{in}} x^\star_{i,j} = 1$ for each $S \subseteq V'$ such that $u^\star_S > 0$

Algorithm 1 [9] describes the global behavior of algorithms for computing an MWA $A(G)^\star$ for a graph G rooted at a given vertex r. Note that the optimality condition 5. implies that: for each $S \subseteq V'$, $u^\star_S > 0 \implies \sum_{(i,j) \in \delta_S^{in}} x^\star_{i,j} = 1$ and $\sum_{(i,j) \in \delta_S^{in}} x^\star_{i,j} > 1 \implies u^\star_S = 0$ (because $u^\star_S \geq 0$). The different values of dual variables $u_S, \forall S \subseteq V'$ are obtained during the execution of Edmonds algorithm. Actually, for each vertex $k \in V : u^\star_k = \arg\min_{(v,k) \in \delta_k^{in}} w(v,k)$ (line 8 of Algorithm 1). If a subset $S \subseteq V' : |S| \geq 2$ is a strong component[1], then $u^\star_S$ is the minimum reduced cost of edges in δ_S^{in}. Only these subsets can have $u^\star_S > 0$. All other subsets ($S \subseteq V' : |S| \geq 2$ and S is not a strong component) have $u^\star_S = 0$. So there are $O(|V|)$ subsets $S \subseteq V' : |S| \geq 2$ that can have $u^\star_S > 0$. A straightforward algorithm can compute $rc(i,j) = w(i,j) - \sum_{(i,j) \in \delta_S^{in}, \forall S \subseteq V'} u^\star_S, \forall (i,j)$ in $O(|V|^3)$ by considering, for each edge, the $O(|V|)$ subsets that are directed cycles. Fischetti and Toth [9] proposed an $O(|V|^2)$ algorithm to compute $rc(i,j), \forall (i,j) \in E$.

Consider the graph $G_1 = (V_1, E_1)$ in Fig. 1 with the vertex 0 as root. Figures 2, 3 and 4 show the different steps needed to construct $A(G_1)^\star$.

Related works in CP. Many works focused on filtering the Minimum Spanning Tree (MST) constraints on undirected weighted graphs (see for examples [22], [21], [5], [4]). About directed graphs, Lorca ([17]) proposed some constraints on trees and forests. In particular, the `tree` constraint [1,17] is about anti-arborescence on directed graph. By considering a set of vertices (called resource),

[1] A strong component of a graph G is a maximal (with respect to set inclusion) vertex set $S \subseteq V$ such that (i) $|S| = 1$ or (ii) for each pair of distinct vertices i and j in S, at least one path exists in G from vertex i to vertex j [9].

Algorithm 1: Computation of a minimum weight arborescence $A(G)^\star$ rooted at vertex r

```
   Input: $G = (V, E)$ ; $r \in V$ ; $w(e), \forall e \in E$
   // all primal and dual variables $x^\star_{i,j}$ and $u^\star_S$ are assumed to be zero
1  foreach each edge $(i, j) \in E$ do
2      $rc(i, j) \leftarrow w(i, j)$
3  $A_0 \leftarrow \emptyset$; $h \leftarrow 0$
   // Phase 1:
4  while $G_0 = (V, A_0)$ is not r-connected do
       // A graph is r-connected iff there is a path from the vertex r
          to each other vertex v in V.
5      $h \leftarrow h + 1$
6      Find any strong component $S_h$ of $G_0$ such that $r \notin S_h$ and $A_0 \cap \delta^{in}_{S_h} = \emptyset$
       // If $|S_h| > 1$, then $S_h$ is a directed cycle
7      Determine the edge $(i_h, j_h)$ in $\delta^{in}_{S_h}$ such that $rc(i_h, j_h) \leq rc(e), \forall e \in \delta^{in}_{S_h}$
8      $u^\star_{S_h} \leftarrow rc(i_h, j_h)$ // dual variable associated to $S_h$
9      $x^\star_{i_h,j_h} \leftarrow 1$;   $A_0 \leftarrow A_0 \cup \{i_h, j_h\}$
10     foreach each edge $(i, j) \in \delta^{in}_{S_h}$ do
11         $rc(i, j) \leftarrow rc(i, j) - u^\star_{S_h}$
   // Phase 2:
12 $t \leftarrow h$
13 while $t \geq 1$ do
       // Extend $A_0$ to an arborescence by letting all but one edge of
          each strong component $S$
14     if $x^\star_{i_t,j_t} = 1$ then
15         foreach each $q < t$ such that $j_t \in S_q$ do
16             $x^\star_{i_q,j_q} \leftarrow 0$
17             $A_0 \leftarrow A_0 \setminus \{(i_q, j_q)\}$
18     $t \leftarrow t - 1$
```

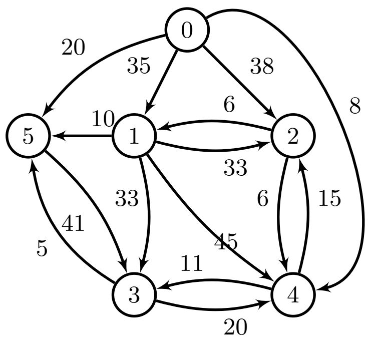

Fig. 1. Initial graph G_1

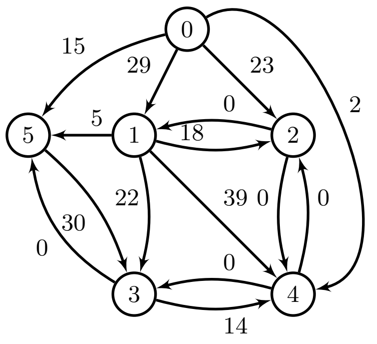

Fig. 2. Computation of $A(G_1)^{\star}$: Phase 1, after selection of all single vertices. $A_0 = \{(2,1),(4,2),(4,3),(2,4),(3,5)\}$.

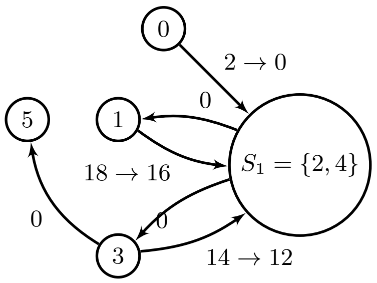

Fig. 3. Computation of $A(G_1)^{\star}$: Phase 1, after the detection of the size 2 strong component $\{2,4\}$. $A_0 = \{(2,1),(4,2),(4,3),(2,4),(3,5),(\mathbf{0},\mathbf{4})\}$.

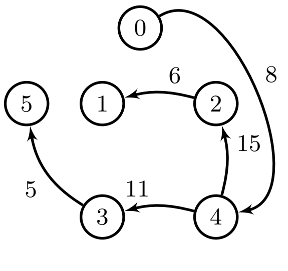

Fig. 4. Computation of $A(G_1)^{\star}$: Phase 2. $A_0 = \{(2,1),(4,2),(4,3),(3,5),(0,4)\}$. $(2,4)$ is removed.

the `tree` constraint partitions the vertices of a graph into a set of disjoint anti-arborescences such that each anti-arborescence points to a resource vertex. The author proposed a GAC filtering algorithm in $O(|E||V|)$ that is in $O(|V|^3)$. Further, Lorca and Fages [18] propose an $O(|E| + |V|)$ filtering algorithm for this constraint.

Here, we focus on an optimization oriented constraint for MWA mentioned in [11]. The authors consider MWA as a relaxation of the Traveling Salesman Problem (TSP) and use LP reduced costs to filter inconsistent values. In this paper, we formally define the optimization constraint for MWA and propose an algorithm to improve the LP reduced costs of MWA.

3 The `MinArborescence` Constraint

To define the `MinArborescence` constraint, we use the predecessor variable representation of a graph. The arborescence is modeled with one variable X_i for each vertex i of G representing its predecessor. The initial domain of a variable X_i is thus the neighbors of i in G: $j \in D(X_i) \equiv (j,i) \in E$. The constraint `MinArborescence(X, w, r, K)` holds if the set of edges $\{(X_i, i) : i \neq r\}$ is a valid arborescence rooted at r with $\sum_{i \neq r} w(X_i, i) \leq K$.

The consistency of the constraint is achieved by computing an exact MWA $A(G)^\star$ rooted at r and verifying that $w(A(G)^\star) \leq K$. The value $w(A(G)^\star)$ is an exact lower bound for the variable K: $K \geq w(A(G)^\star)$. The filtering of the edges can be achieved based on the reduced costs. For a given edge $(i,j) \notin A(G)^\star$, if $w(A(G)^\star) + rc(i,j) > K$, then $X_i \leftarrow j$ is inconsistent. In section 4, we propose a procedure to strengthen the quality of the LP reduced costs in $O(|V|^2)$ in some cases.

Note that, a GAC filtering would require exact reduced costs, that to the best of our knowledge can only be obtained by recomputing an MWA from scratch in $O(|E||V|^2)$ which is in $O(|V|^4)$.

The basic decomposition of the `MinArborescence` constraint does not scale well due to the exponential number of constraints in equation (3). We propose a light constraint (called the `Arborescence` constraint) to have a scalable baseline model for experiments.

Decomposing `MinArborescence`. The constraint `Arborescence(X, r)` holds if the set of edges $\{(X_i, i) : \forall i \in V'\}$ is a valid arborescence rooted at r. We introduce an incremental forward checking like incremental filtering procedure for this constraint in Algorithm 2. Our algorithm is inspired by the filtering described in [20] for enforcing a Hamiltonian circuit. During the search, the bound variables form a forest of arborescences. Eventually when all the variables are bound, a unique arborescence rooted at r is obtained. The filtering maintains for each node:

1. a reversible integer value $localRoot[i]$ defined as $localRoot[i] = i$ if X_i is not bound, otherwise it is recursively defined as $localRoot[X_i]$, and

2. a reversible set $leafNodes[i]$ that contains the leaf nodes of i in the forest formed by the bound variables.

The filtering prevents cycles by removing from any successor variable X_v all the values corresponding to its leaf nodes (i.e. the ones in the set $leafNodes[v]$). Algorithm 2 registers the filtering procedure to the bind events such that `bind(i)` is called whenever the variable X_i is bind. This `bind` procedure then finds the root lr of the (sub) arborescence at line 15. Notice that j is not necessarily the root as vertex j might well have been the leaf node of another arborescence that is now being connected by the binding of X_i to value j. This root lr then inherits all the leaf nodes of i. None of these leaf nodes is allowed to become a successor of lr, otherwise a cycle would be created.

Algorithm 2: Class Arborescence

```
1  X: array of $|V|$ variables;
2  localRoot: array of $|V|$ reversible int;
3  leafNodes: array of $|V|$ reversible set of int;
4  Method init()
5      $X_r \leftarrow r$ ;                                        // self loop on $r$
6      foreach each vertex $v \in V'$ do
7          $leafNodes[v].insert(v)$ ;
8          $X_v.removeValue(v)$ ;                                   // no self loop
9          if $X_v.isBound$ then
10             bind(v);
11         else
12             $X_v.registerOnBindChanges()$ ;
13 Method bind($i$: int)
14     $j \leftarrow X_i$ ;                                 // edge $(j, i) \in$ arborescence
15     $lr \leftarrow localRoot[j]$ ;                                // local root of $j$
16     foreach each $v \in leafNodes[i]$ do
17         $localRoot[v] \leftarrow lr$ ;
18         $leafNodes[lr].insert(v)$ ;
19         $X_{lr}.removeValue(v)$ ;
```

The `MinArborescence(X, w, r, K)` constraint can be decomposed as the `Arborescence(X, r)` constraint plus $\sum_{i \neq r} w(X_i, i) \leq K$, a sum over element constraints. .

4 Improved Reduced Costs

Let $A(G)^{\star}_{i \to j}$ be an MWA of the graph G when the edge (i, j) is forced to be in it. We know that the LP reduced cost $rc(i, j)$ gives a lower bound on the associated cost increase: $w(A(G)^\star_{i\to j}) \geq w(A(G)^\star) + rc(i,j)$. However, this lower bound on $w(A(G)^\star_{i\to j})$ can be improved in some cases.

Let us use the following notation to characterize an MWA $A(G)^\star$.
Let $pred[v], \forall v \in V'$, be the vertex in V such that the edge $(pred[v], v) \in A(G)^\star$: $x^\star_{pred[v],v} = 1$. For example, in the graph G_1, $pred[1] = 2$ and $pred[5] = 3$. Let $parent[S]$ be the smallest cycle strictly containing the subset $S \subseteq V'$: $parent[S]$ is the smallest subset $> |S|$ such that $\sum_{(i,j)\in\delta^{in}_{parent[S]}} x^\star_{i,j} = 1$ and $S \subset parent[S]$. In other words, $parent[S]$ is the first directed cycle that includes the subset S found during the execution of the Edmonds' algorithm. We assume that $parent[S] = \emptyset$ if there is no such cycle and $parent[\emptyset] = \emptyset$. In the graph G_1, $parent[1] = parent[3] = parent[5] = \emptyset$ and $parent[2] = parent[4] = \{2,4\}$. Here, $parent[parent[k]] = \emptyset, \forall k \in V'$.

The next three properties give some information to improve the LP reduced costs when all vertices involved do not have a parent.

Property 1. Assume that there is a path $\mathcal{P} = (j, \ldots, i)$ from the vertex j to vertex i in $A(G)^\star$ such that $\forall k \in \mathcal{P} : parent[k] = \emptyset$. If the edge (i,j) is forced to be in $A(G)^\star$, then the cycle $c = (k : k \in \mathcal{P})$ will be created during the execution of Edmonds' algorithm.

Proof. $parent[k] = \emptyset$ means that $pred[k]$ is such that $w(pred[k], k) = \min\{w(v,k) : (v,k) \in \delta^{in}_k\}$ and then $pred[k]$ will be first selected by Edmonds' algorithm $\forall k \in \mathcal{P} \setminus \{j\}$. On the other hand, if the edge (i,j) is forced into the MWA it implies that all other edges entering j are removed. Consequently, the cycle $c = (k : k \in \mathcal{P})$ will be created. □

Let us use the following notations to evaluate the improved reduced costs. Let $min1[k] = \arg\min_{(v,k)\in\delta^{in}_k} w(v,k)$ be the minimum cost edge entering the vertex k. If there is more than one edge with the smallest weight, we choose one of them arbitrarily. Also, let $min2[k] = \arg\min_{(v,k)\in\delta^{in}_k \wedge (v,k)\neq min1} w(v,k)$ be the second minimum cost edge entering k. For each vertex $k \in V$, let $bestTwoDiff[k]$ be the difference between the best two minimum costs of edges entering the vertex k: $\forall k \in V, bestTwoDiff[k] = w(min2[k]) - w(min1[k])$. For instance, in the graph G_1: $min1[5] = (3,5)$, $min2[5] = (1,5)$ and $bestTwoDiff[5] = 10 - 5 = 5$.

Property 2. Consider the cycle $c = (i, \ldots, j)$ obtained by forcing the edge (i,j) such that $parent[k] = \emptyset, \forall k \in c$ (see Property 1). The minimum cost increase if the cycle is broken/connected by the vertex $k' \in c$ is $bestTwoDiff[k']$.

Proof. For a given $k' \in c$, $parent[k'] = \emptyset$ implies that the edge $(pred[k'], k') = min1[k']$. Then the cheapest way to break the cycle by k' is to use the edge with the second minimum cost $min2[k']$. Hence the minimum cost increase if the cycle is broken by the vertex k' is $w(min2[k']) - w(min1[k']) = bestTwoDiff[k']$. □

When $parent[j] = \emptyset$, the LP reduced costs $rc(i,j)$ are simple and can be easily interpreted.

Property 3. Consider a vertex $j \in V'$ such that $parent[j] = \emptyset$. For all $i \in V \setminus \{j\}$ with $(i, j) \notin A(G)^\star$: $rc(i, j) = w(i, j) - w(pred[j], j)$.

Proof. We know that if $parent[j] = \emptyset$, then for each $S \subseteq V'$ with $j \in S$ and $|S| \geq 2$, $u^\star_S = 0$ (because none of them is a directed cycle). Then $rc(i, j) = w(i, j) - u^\star_j$. On the other hand, since $parent[j] = \emptyset$, the edge $min1[j]$ is the one used to connect j in $A(G)^\star$. So $u^\star_j = w(min1[j]) = w(pred[j], j)$ and $rc(i, j) = w(i, j) - w(pred[j], j)$. □

By considering an MWA $A(G)^\star$, the interpretation of $rc(i, j)$ when $parent[j] = \emptyset$ is that the edge (i, j) is forced into $A(G)^\star$ and the edge $(pred[j], j)$ that is used to connect j is removed. Intuitively, if this process induces a new cycle, this latter has to be (re)connected to the rest of the arborescence from a vertex in the cycle different from j. Proposition 3 established below, gives a first improved reduced cost expression when a new cycle c is created by forcing (i, j) into the arborescence and $\forall k \in c, parent[k] = \emptyset$. Note that such new cycle will be created only if there is already a path in $A(G)^\star$ from the vertex j to the vertex i.

Proposition 3. *Assume that there is a path $\mathcal{P} = (j, \ldots, i)$ from the vertex j to vertex i in $A(G)^\star$ such that $\forall k \in \mathcal{P} : parent[k] = \emptyset$. We have: $w(A(G)^\star_{i \to j}) \geq w(A(G)^\star) + rc(i, j) + \min_{k \in \mathcal{P} \setminus \{j\}}\{bestTwoDiff[k]\}$.*

Proof. Without loss of generality, we assume that the cycle $c = (k : k \in \mathcal{P})$ is the first one created by the Edmonds' algorithm (see Property 1). After this step, the new set of vertices is $V' = \{c\} \cup \{v \in V : v \notin c\}$. In $A(G)^\star$, the edges assigned to vertices in c do not influence the choice of edges for each vertex $v \in V : v \notin c$ (because $parent[k] = \emptyset, \forall k \in c$). So $w(A(G)^\star_{i \to j}) \geq \sum_{k \in V \wedge k \notin c} w(pred[k], k) + w(c)$ in which $w(c)$ is the minimum sum of costs of edges when exactly one edge is assigned to each vertex in c without cycle. The cheapest way to connect all vertices in c such that each one has exactly one entering edge is to use all cheapest entering edges $(min1[k], \forall k \in c)$. The cycle obtained must be broken in the cheapest way. To do so, the vertex used: 1) must be different from j (because (i, j) is already there) and 2) have to induce the minimal cost increase. Then, from Property 2, a lower bound on the minimum cost increase is $\min_{k \in \mathcal{P} \setminus \{j\}}\{bestTwoDiff[k]\}$. In addition, we have to add the cost of the forced edge (i, j) and remove the cost of the edge in $A(G)^\star$ that enters in j: $w(c) \geq \sum_{k \in c} w(pred[k], k) + \min_{k \in c \setminus \{j\}}\{bestTwoDiff[k]\} + w(i, j) - w(pred[j], j)$. We know that $\sum_{k \in c} w(pred[k], k) + \sum_{k \in V \wedge k \notin c} w(pred[k], k) = w(A(G)^\star)$ and $rc(i, j) = w(i, j) - w(pred[j], j)$ (see Property 3).
Thus $w(A(G)^\star_{i \to j}) \geq w(A(G)^\star) + rc(i, j) + \min_{k \in \mathcal{P} \setminus \{j\}}\{bestTwoDiff[k]\}$. □

Example 1. Consider the graph G_1 presented in Fig. 1 and its MWA (Fig. 4). We want to force $(5, 3)$ to be into the MWA:

- $rc(5, 3) = w(5, 3) - w(4, 3) = 41 - 11 = 30$. This operation leads to the graph shown in Fig. 5 (a). We can see that the new cycle created $c = (5, 3)$ must be broken from a vertex different from 3.

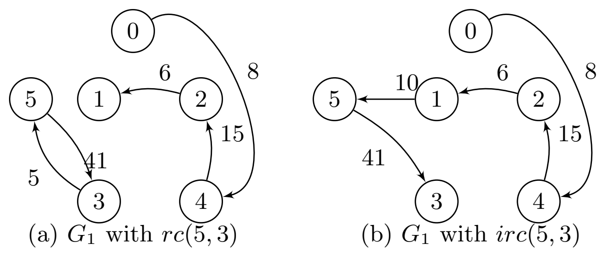

Fig. 5. G_1 with $rc(5,3)$ and $irc(5,3)$

– $irc(5,3) = rc(5,3) + bestTwoDiff[5] = 30 + 5 = 35$. The corresponding graph is shown in Fig. 5 (b), that actually is the new MWA.

Property 2 and Proposition 3 can be generalized into Property 4 and Proposition 4 below to include some vertices that have one parent.

Property 4. Consider an ordered set of vertices $(k_1, k_2, \ldots, k_n)$ in $A(G)^\star$ such that $c = (k_1, k_2, \ldots, k_n)$ is a directed cycle connected (broken) by the vertex $k^\star$ and $parent[c] = \emptyset$. The minimum cost increase if the cycle is broken by another vertex $k' \in c \setminus \{k^\star\}$ is $\geq bestTwoDiff[k'] - u_c^\star$.

Proof. We know that $parent[c] = \emptyset$ implies $u_c^\star = \min\{w(min2[k]) - w(min1[k]) : k \in c\} = w(min2[k^\star]) - w(min1[k^\star])$. So if the cycle is now connected by another vertex than $k^\star$, the edge $min1[k^\star]$ can be used instead of $min2[k^\star]$ and decreases the cost by $w(min2[k^\star]) - w(min1[k^\star]) = u_c^\star$. On the other hand, $\forall k_i \in c \setminus \{k^\star\}$, $(pred[k_i], k_i) = min1[k_i]$. The cheapest way to use k' to connect c is to use $min2[k']$. Hence, a lower bound on the total cost induced is $min2[k'] - min1[k'] - u_c^\star = bestTwoDiff[k'] - u_c^\star$. □

Now the improved reduced costs can be formulated as follows.

Proposition 4. *Assume that there is a path $\mathcal{P} = (j, \ldots, i)$ from the vertex j to vertex i in $A(G)^\star$ such that $\forall k \in \mathcal{P} : parent[parent[k]] = \emptyset$. Then $w(A(G)^\star_{i \to j}) \geq w(A(G)^\star) + rc(i,j) + \min_{k \in \mathcal{P} \setminus \{j\}} \{bestTwoDiff[k] - u^\star_{parent[k]}\}$.*

Proof. Note that if $\forall k \in \mathcal{P} : parent[k] = \emptyset$ (that implies that $u^\star_{parent[k]} = 0$), the formula is the same as the one of Proposition 3. Let Z denote the set of vertices in $\mathcal{P}$ and in all other cycles linked to $\mathcal{P}$. Formally, $Z = \{v \in V : v \in \mathcal{P}\} \cup \{k \in V : \exists v \in \mathcal{P} \wedge parent[k] = parent[v]\}$. We know that, in $A(G)^\star$, the edges assigned to vertices in Z do not influence the choice of edges for each vertex $k \in V \setminus Z$. So $w(A(G)^\star_{i \to j}) \geq \sum_{k \in V \setminus Z} w(pred[k], k) + w(Z)$ in which $w(Z)$ is the minimum sum of the costs of edges when we assign exactly one edge to each vertex in Z without cycle. The reasoning is close to the proof of Proposition 3. The differences here are:

1. $\exists k \in \mathcal{P} \setminus \{j\} : parent[k] \neq \emptyset$. Assume that we want to break the cycle by one vertex $v^\star$ in $\mathcal{P} \setminus \{j\}$. If the vertex $v^\star$ used is such that $parent[v^\star] = \emptyset$, then the minimum cost to pay is $\geq bestTwoDiff[v^\star]$ (here $u^\star_{parent[v^\star]} = 0$ because $parent[v^\star] = \emptyset$). If $v^\star$ is such that $parent[v^\star] \neq \emptyset \wedge parent[parent[v^\star]] = \emptyset$, then from Property 4, the cost to pay is $\geq bestTwoDiff[v^\star] - u^\star_{parent[v^\star]}$. By considering all vertices in $\mathcal{P} \setminus \{j\}$, the cost to pay is then $\geq \min_{k \in \mathcal{P} \setminus \{j\}} \{bestTwoDiff[k] - u^\star_{parent[k]}\}$.
2. the vertex j may have one parent. Let *connect* be the vertex that is used to connect the cycle $parent[j]$ in $A(G)^\star$.
   Case 1: $parent[i] \neq parent[j]$. If we force the edge (i, j), then the cycle $parent[j]$ should not be created because it is as if all edges entering j but (i, j) are removed. First, assume that $j \neq connect$. The edge in $parent[j]$ not in $A(G)^\star$ should be used and the cost won is $u^\star_{parent[j]}$ (as in the proof of Property 4). Thus, a lower bound on the cost to break the cycle $parent[j]$ by j is: $w(i, j) - w(pred[j], j) - u^\star_{parent[j]}$. This lower bound is equal to $rc(i, j)$ because $w(pred[j], j) = w(min1[j])$. Now assume that $j = connect$. In this case $(pred[j], j) = min2[j]$ (because $parent[parent[j]] = \emptyset$). Using the edge (i, j) instead of $(pred[j], j)$ induces the cost $w(i, j) - w(min2[j]) = w(i, j) - w(min1[j]) - w(min2[j]) + w(min1[j]) = w(i, j) - w(min1[j]) - u^\star_{parent[j]} = rc(i, j)$.
   Case 2: $parent[i] = parent[j] \neq \emptyset$. This means that the edge (i, j) is the one of the cycle that is not in the MWA. In this case $rc(i, j) = 0$, and the new cycle created should be broken as described above (1.).

Hence, a lower bound on $w(Z)$ is

$$\sum_{k \in Z} w(pred[k], k) + \min_{k \in \mathcal{P} \setminus \{j\}} \{bestTwoDiff[k] - u^\star_{parent[k]}\} + rc(i, j)$$

and $w(A(G)^\star_{i \to j}) \geq w(A(G)^\star) + \min_{k \in \mathcal{P} \setminus \{j\}} \{bestTwoDiff[k] - u^\star_{parent[k]}\} + rc(i, j)$. □

Note that $parent[parent[k]] = \emptyset$ if k is not in a cycle or k is in a cycle that is not contained in a larger cycle. The formula of Proposition 4 is available only if $\forall k \in \mathcal{P} : parent[parent[k]] = \emptyset$. Let $irc(i, j)$ denote the improved reduced cost of the edge (i, j): $irc(i, j) = rc(i, j) + \max\{\min_{k \in \mathcal{P} \wedge k \neq j} \{bestTwoDiff[k] - u^\star_{parent[k]}\}, 0\}$ if the assumption of Proposition 4 is true and $irc(i, j) = rc(i, j)$ otherwise.

Example 2. Consider the graph G_1 in Fig. 1 and its MWA $A(G_1)^\star$ in Fig. 4. For the contruction of $A(G_1)^\star$, the cycle $c_1 = \{2, 4\}$ is created. We want to force into the MWA the edge:

1. $(1, 2)$: $rc(1, 2) = w(1, 2) - u^\star_2 - u^\star_{c_1} = w(1, 2) - w(4, 2) - (w(0, 4) - w(2, 4))$. $rc(1, 2) = 16$. The corresponding graph is presented in Fig. 6 (a). Of course, the new cycle $(1, 2)$ created must be broken from the vertex 1. $irc(1, 2) = rc(1, 2) + (w(0, 1) - w(2, 1)) = 16 + 29 = 45$. Actually, that is the exact

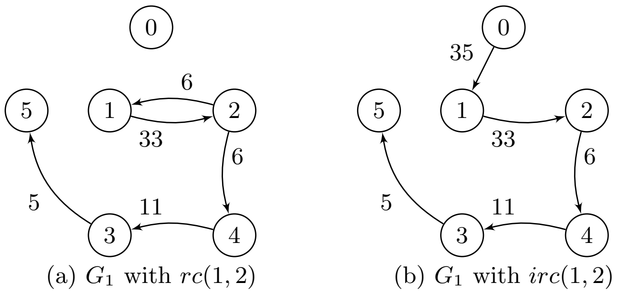

Fig. 6. G_1 with $rc(1,2)$ and $irc(1,2)$

reduced cost since the new graph obtained is an arborescence (see Fig. 6 (b)).

2. $(1,4)$: $rc(1,4) = w(1,4) - u^\star_4 - u^\star_{c_1} = w(1,4) - w(2,4) - (w(0,4) - w(2,4))$. $rc(1,4) = 37$. The corresponding graph is presented in Fig. 7 (a). But $irc(1,4) = rc(1,4) + \min\{w(0,1) - w(2,1) = 29, w(1,2) - w(4,2) - u^\star_{c_1} = 16\}$. So $irc(1,4) = 37 + 16 = 53$. Here (see Fig. 7 (b)), the graph obtained is not an arborescence and $irc(1,4)$ is a lower bound.

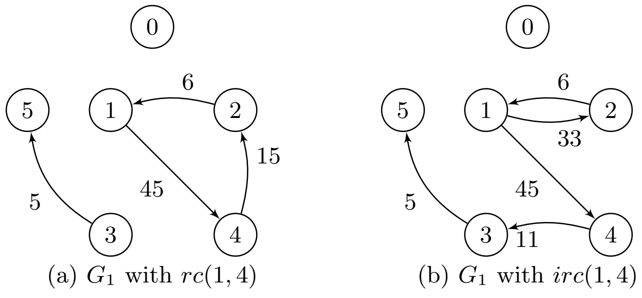

Fig. 7. G_1 with $rc(1,4)$ and $irc(1,4)$

Algorithm 3 computes $irc(i,j), \forall(i,j)$ in $O(|V|^2)$. First, it initializes each $irc(i,j)$ to $rc(i,j)$, $\forall(i,j) \in E$. Then, for each edge (i,j) involved in the assumption of Proposition 4 (Invariant (a)), Algorithm 3 updates its $irc(i,j)$ according to the formula of Proposition 4.

Algorithm 3: Computation of improved reduced costs $irc(i, j), \forall(i, j) \in E$

```
Input: $parent[k], \forall k \in V$ ; $pred[k], \forall k \in V$ ; $u^\star_{c_i}, \forall c_i \in \mathcal{C}$ and
       $bestTwoDiff[k], \forall k \in V$ that can be computed in $O(|V|^2)$
Output: $irc(i, j), \forall(i, j) \in E$
1  foreach each edge $(i, j) \in E$ do
2      $irc(i, j) \leftarrow rc(i, j)$
3  foreach each vertex $i \in V$ do
4      if $parent[parent[i]] = \emptyset$ then
5          $min \leftarrow bestTwoDiff[i] - u^\star_{parent[i]}$
6          $j = pred[i]$
7          while $(parent[parent[j]] = \emptyset) \wedge min > 0 \wedge j \neq r$ do
               // Invariant (a): there is a path $\mathcal{P}$ from $j$ to $i$ such that
                   $\forall k \in \mathcal{P} : parent[parent[k]] = \emptyset$
               // Invariant (b):
                   $min = \min_{k \in \mathcal{P} \setminus \{j\}} \{bestTwoDiff[k] - u_{parent[k]^\star}\}$
8              $irc(i, j) \leftarrow irc(i, j) + min$
9              if $bestTwoDiff[j] - u_{parent[j]^\star} < min$ then
10                 $min \leftarrow bestTwoDiff[j] - u^\star_{parent[j]}$
11             $j = pred[j]$
```

5 Experimental Results

As a first experiment, we evaluate the proportion of reduced costs affected by Proposition 4. Therefore we randomly generated two classes of 100 instances $w(i, j) \in [1, 100]$ with different values of the number of vertices:

- $class1$: for each $i \in V$, $parent[parent[i]] = \emptyset$;
- $class2$: many vertices $i \in V$ are such that $parent[parent[i]] \neq \emptyset$.

The $class1$ was obtained by filtering out the random instances not satisfying the property. Let $exactRC(i, j)$ be the exact reduced cost associated to the edge $(i, j) \in E$. Table 1 shows, for each class of instances (with respectively $|V| = 20$, $|V| = 50$ and $|V| = 100$), the proportion of instances of each class and for each group of instances: 1) the proportion of edges $(i, j) \in E$ such that $rc(i, j) < exactRC(i, j)$; 2) the proportion of edges that have $irc(i, j) > rc(i, j)$; and 3) the proportion of edges such that $irc(i, j) = exactRC(i, j)$. Note that, for this benchmark, at least 37% of 300 instances are $class1$ instances and at least 45% of LP reduced costs (with $rc(i, j) < exactRC(i, j)$) are improved for these instances. Of course, the results are less interesting for the $class2$ instances.

To test the `MinArborescence` constraint, experiments were conducted on an NP-Hard variant of CAP: the Resource constrained Minimum Weight Arborescence Problem (RMWA) [14,10]. The RMWA problem is to find an MWA under the resource constraints for each vertex $i \in V$: $\sum_{(i,j)\in\delta^+_i} a_{i,j} \cdot x_{i,j} \leq b_i$ where δ^+_i is the set of outgoing edges from i, $a_{i,j}$ is the amount of resource

	$\|V\| = 20$		$\|V\| = 50$		$\|V\| = 100$	
	Class1	Class2	Class1	Class2	Class1	Class2
%: instances of class$k$ ($k \in \{1, 2\}$)	48	52	31	69	32	68
%: $rc(i, j) < exactRC(i, j)$	19.3	38.1	9.8	26.7	2.1	16.6
%: $irc(i, j) > rc(i, j)$	12.6	1.9	4.6	0.9	1.2	0.2
%: $irc(i, j) = exactRC(i, j)$	9.9	1.17	3.49	0.79	1.19	0.2

Table 1. Proportion of reduced costs affected by Proposition 4

uses by the edge (i, j) and b_i is the resource available at vertex i. RMWA can be modeled in CP with a `MinArborescence` constraint (or one of its decompositions) for the MWA part of problem and the `binaryKnapsack` constraint [7] together with `weightedSum` constraint for the resource constraints. We have randomly generated the different costs/weights as described in [14]: $a_{i,j} \in [10, 25]$, $w(i, j) \in [5, 25]$. To have more available edges to filter, we have used $b_i = 2 \cdot \lfloor \frac{\sum_{(i,j)\in\delta_i^+} a_{i,j}}{|\delta_i^+|} \rfloor$ (instead of $b_i = \lfloor \frac{\sum_{(i,j)\in\delta_i^+} a_{i,j}}{|\delta_i^+|} \rfloor$) and 75% graph density.

In order to avoid the effect of the dynamic first fail heuristic interfering with the filtering, we use the approach described in [25] to evaluate global constraints. This approach consists in recording the search tree with the weakest filtering as a baseline. It is obtained with the decomposition model using the `Arborescence` constraint. This search tree is then replayed with the stronger reduced cost based filtering for `MinArborescence`. The recorded search tree for each instance corresponds to an exploration of 30 seconds. As an illustration for the results, Table 2 details the computational results for `MinArborescence` constraint with filtering based on improved reduced costs (MinArbo_IRC), reduced costs (MinArbo_RC), the decomposition with `Arborescence` constraint (Arbo) and Arbo+filtering only based on lower bound on MWA (Arbo+LB) on 4 (arbitrarily chosen) out of the 100 randomly instances with $|V| = 50$. We also report the average results for the 100 instances. On average, the search space is divided by ≈ 460 with the reduced costs based filtering `MinArborescence` constraint (wrt Arbo) and by ≈ 81 with Arbo+LB. This demonstrates the benefits brought by the `MinArborescence` global constraints described in this paper.

To further differentiate the filtering of MinArbo_IRC, we now use MinArbo_RC as a baseline filtering for recording the search tree on another set of 100 randomly generated instances of $class1$ with $|V| = 50$. Fig. 8 shows the corresponding performance profiles wrt the number of nodes visited and the time used respectively. For $\approx 30\%$ of instances the search space is divided by at least 1.5 and for $\approx 7\%$ the search space is divided by at least 4. On the other hand, the average gain for MinArbo_IRC (wrt MinArbo_RC) is 1.7 wrt the number of nodes visited and 1.4 wrt time. Unfortunately, as was expected, the average gain is limited as only $\approx 5\%$ of LP reduced costs can be improved.

Instance	MinArbo_IRC		MinArbo_RC		Arbo+LB		Arbo	
	Nodes	Time(s)	Nodes	Time	Nodes	Time	Nodes	Time
1	20259	0	20259	0	40061	1	5879717	28
2	12552	0	12552	0	16794	0	6033706	27
3	13094	0	13094	0	121290	2	6383651	28
4	62607	0	62607	0	283854	6	7316899	29
Average	**14385**	**0**	**14385**	**0**	**81239**	**1.4**	**6646748**	**28**

Table 2. Results: MinArbo_IRC, MinArbo_RC, Arbo+LB and Arbo

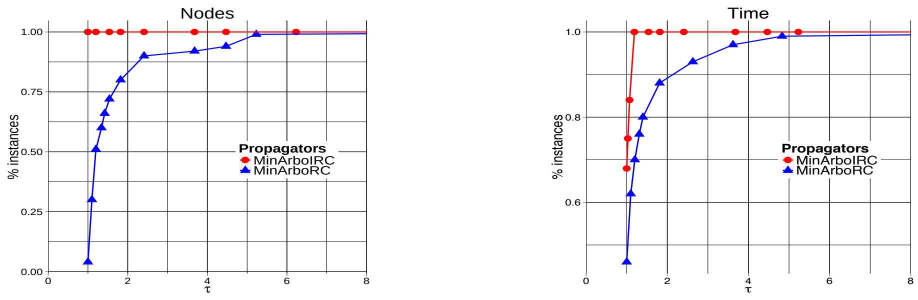

Fig. 8. Performance profiles wrt number of nodes visited and time

The implementations and tests have been realized within the OscaR open source solver [24]. Our source-code and the instances are available at [15]. Our CP model is able to solve and prove optimality of RMWA instances with up to $|V| = 50$. Similar instances can be solved using the Lagrangian decomposition approach of [14]. The branch and cut algorithm of [10] reports results solving instances with up to $|V| = 500$. We believe this lack of performance of CP wrt to the branch and cut approach is due to the $|V|$ independent knapsack constraints inducing a weaker pruning. We do hope this first result will trigger more research in the future to make CP more competitive on this challenging problem.

6 Conclusion

We have defined the `MinArborescence` constraint based on the reduced costs to filter the edges for a constrained arborescence problem. We have proposed an algorithm to improve the LP reduced costs of the minimum weighted arborescence in some cases. Finally, we have demonstrated experimentally the interest of improved reduced costs in some particular graphs and the efficiency of the cost-based filtering on the resource constrained arborescence problem. As future works, we would like to: 1) think about a global constraint wrt resource constraints 2) study the incremental aspects of the `MinArborescence` constraint and 3) propose specialized search heuristics.

References

1. N. Beldiceanu, P. Flener, and X. Lorca. The tree constraint. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 64–78. Springer, 2005.
2. F. Bock. An algorithm to construct a minimum directed spanning tree in a directed network. *Developments in Operations Research*, pages 29–44, 1971.
3. Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Sci. Sin., Ser. A*, 14:1396–1400, 1965.
4. G. Dooms and I. Katriel. The minimum spanning tree constraint. In *Principles and Practice of Constraint Programming - CP 2006*, pages 152–166. Springer, 2007.
5. G. Dooms and I. Katriel. The not-too-heavy spanning tree constraint. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - CPAIOR 2007*, pages 59–70. Springer, 2007.
6. J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau for Standards*, 71B(4):125–30, 1967.
7. T. Fahle and M. Sellmann. Cost based filtering for the constrained knapsack problem. *Annals of Operations Research*, 115(1-4):73–93, 2002.
8. M. Fischetti and P. Toth. An additive bounding procedure for asymmetric travelling salesman problem. *Mathematical Programming*, 53:173 – 197, 1992.
9. M. Fischetti and P. Toth. An efficient algorithm for min-sum arborescence problem on complete digraphs. *Management Science*, 9(3):1520–1536, 1993.
10. M. Fischetti and D. Vigo. A branch-and-cut algorithm for the resource-constrained minimum-weight arborescence problem. *Network*, 29:55–67, 1997.
11. F. Focacci, A. Lodi, M. Milano, and D. Vigo. Solving tsp through the integration of or and cp techniques. *Electronic notes in discrete mathematics*, 1:13–25, 1999.
12. H. N. Gabow, Z. Galil, T. H. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(3):109–22, 1986.
13. R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *History of computing*, 7:13–25, 1985.
14. M. Guignard and M. B. Rosenwein. An application of lagrangean decomposition to the resource-constrained minimum weighted arborescence problem. *Network*, 20:345–359, 1990.
15. V. R. Houndji and P. Schaus. Cp4cap: Constraint programming for constrained arborescence problem. Available from `https://bitbucket.org/ratheilesse/cp4cap`.
16. J. Kleinberg and E. Tardos. *Algorithm Design*, chapter 4: Minimum-Cost Arborescences: A multi-phase greedy algorithm. Tsinghua University Press, 2005.
17. X. Lorca. *Contraintes de Partitionnement de Graphe*. PhD thesis, Université de Nantes, 2010.
18. X. Lorca and J-G. Fages. Revisiting the tree constraint. In *Principles and Practice of Constraint Programming - CP 2011*, volume 6876, pages 271–285, 2011.
19. R. Mendelson, R.E. Tarjan, M. Thorup, and U. Zwick. Melding priority queues. *Proceedings of SWAT 04*, 3111(3):223–235, 2004.
20. G. Pesant, M. Gendreau, J-Y. Potvin, and J-M. Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1):12–29, 1998.
21. J-C. Régin. Simpler and incremental consistency checking and arc consistency filtering algorithms for the weighted spanning tree constraint. In *Integration of AI*

and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - CPAIOR 2008, pages 233–245. Springer, 2008.

22. J-C. Régin, L-M. Rousseau, M. Rueher, and W. van Hoeve. The weighted spanning tree constraint revisited. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - CPAIOR 2010*, pages 233–245. Springer, 2008.
23. R. E. Tarjan. Finding optimum branchings. *Networks*, 7(3):25–35, 1977.
24. OscaR Team. Oscar: Scala in or. `https://bitbucket.org/oscarlib/oscar`, 2012.
25. S. Van Cauwelaert, M. Lombardi, and P. Schaus. Understanding the potential of propagators. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - CPAIOR 2015*, pages 427–436. Springer, 2015.