algorithms

Article

Compact Models to Solve the Precedence-Constrained Minimum-Cost Arborescence Problem with Waiting Times

Mauro Dell'Amico [1,2], Jafar Jamal [1] and Roberto Montemanni [1,2,*]

1 Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola 2, 42122 Reggio Emilia, RE, Italy; mauro.dellamico@unimore.it (M.D.)
2 Interdepartmental Center En&Tech, University of Modena and Reggio Emilia, Capannone 19 Tecnopolo, Piazza Europa 1, 42122 Reggio Emilia, RE, Italy
* Correspondence: roberto.montemanni@unimore.it; Tel.: +39-0522-522-126

Abstract: The minimum-cost arborescence problem is a well-studied problem. Polynomial-time algorithms for solving it exist. Recently, a new variation of the problem called the Precedence-Constrained Minimum-Cost Arborescence Problem with Waiting Times was presented and proven to be $\mathcal{NP}$-hard. In this work, we propose new polynomial-size models for the problem that are considerably smaller in size compared to those previously proposed. We experimentally evaluate and compare each new model in terms of computation time and quality of the solutions. Several improvements to the best-known upper and lower bounds of optimal solution costs emerge from the study.

Keywords: arborescences; precedence constraints; mixed integer linear programming; constraint programming

Citation: Dell'Amico, M.; Jamal, J.; Montemanni, R. Compact Models to Solve the Precedence-Constrained Minimum-Cost Arborescence Problem with Waiting Times. *Algorithms* **2024**, *17*, 12. https://doi.org/10.3390/a17010012

Academic Editor: Frank Werner

Received: 2 December 2023
Revised: 23 December 2023
Accepted: 26 December 2023
Published: 27 December 2023

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

The *Minimum-Cost Arborescence* (MCA) problem asks to find a directed minimum-cost spanning tree (i.e., an arborescence) rooted at vertex r—the *root* of the arborescence—in a directed graph. Chu and Liu [1] and Edmonds [2] proposed the same polynomial time algorithm to solve the problem, independently from each other. A faster implementation was later proposed by Gabow and Tarjan [3], and a different polynomial time algorithm [4] that operates directly on the cost matrix was introduced by Bock [5].

Several variations of the MCA problem were introduced in the literature since its introduction. Given a directed graph with a resource associated to each of its vertices, the *Resource-Constrained Minimum-Weight Arborescence problem* aims at retrieving an arborescence with minimum total cost, with the additional constraint that outgoing arcs from each vertex have to have a cost at most equal to that of the resource of the vertex itself (Fischetti and Vigo [6]). Given a weighted directed graph $G = (V, A)$ with a vertex $r \in V$ identified as the root and an integer p, the *p-Arborescence Star problem* asks to identify a minimum-cost arborescence rooted at r. The arborescence spans the set of vertices $H \subseteq V \backslash \{r\}$ of size p, and there must be an assignment between each vertex $v \in V \backslash \{H \cup r\}$ and one of the vertices in H (Pereira et al. [7], Morais et al. [8], Hakimi [9]). Given a directed graph with a number assigned to each vertex, the *Restricted Fathers Tree problem* seeks a minimum-cost arborescence, with the constraint that each path between a vertex and the root has to touch vertices with ranking not lower than the vertex (Guttmann-Beck and Hassin [10]). Given a directed graph with a vertex r designed as the root and a set $A_v \subset V$ for every $v \in V \backslash \{r\}$ such that $r \in A_v$, the *Restricted Ancestors Tree problem* aims at finding a minimum-cost arborescence rooted at r, with the additional constraint that vertex i can be an ancestor of vertex j only when $i \in A_j$ (Guttmann-Beck and Hassin [10]). The *Minimum Spanning Tree Problem with Conflict Pairs* (Carrabs and Gaudioso [11]) is a variation of the classic *Minimum-Spanning Tree problem* (Kruskal [12]), characterized by an undirected graph and a set S

containing conflicting pairs of conflicting edges. The objective is to retrieve a minimum-cost spanning tree with at most one edge from the pair in S. The *Capacitated Minimum Spanning Tree problem*, introduced in Gouveia and Lopez [13], is another variation, characterized by non-negative integer node demands q_j for each node $j \in V \setminus \{r\}$ and a budget Q for the sum of the weights in any root–leaf path. Further related problems can be found in Frieze and Tkocz [14], Fertin et al. [15], Eswaran and Tarjan [16], Li et al. [17], Kawatra and Bricker [18], Galbiati et al. [19], Bérczi et al. [20], Bang-Jensen [21], Yingshu et al. [22], Carrabs et al. [23], Darmann et al. [24] and Viana and Campêlo [25]. The *Sequential Ordering Problem*, introduced in Escudero [26], is relevant to the present study and can be described as follows. Given a weighted graph and a set of precedence constraints between vertex pairs and start and end vertices, the goal is to find a Minimum-Cost Hamiltonian Path that respects the precedence constraints. Solving algorithms can be found in Moon et al. [27], Balas et al. [28], Hernádvölgyi [29], Escudero et al. [30], Gambardella and Dorigo [31], Karan and Skorin-Kapov [32], Ascheuer et al. [33], Ascheuer et al. [34], Montemanni et al. [35] and Fiala Timlin and Pulleyblank [36]. The *Precedence-Constrained Minimum-Cost Arborescence* is an extension of the MCA problem first introduced in Dell'Amico et al. [37]. Precedence constraints have the following meaning. A precedence set R containing pairs of vertices is given. For each $(s, t) \in R$, if both vertices s and t are on a same path of the arborescence, then vertex s has to be visited before vertex t. The optimization seeks to find an arborescence of minimum total cost such that all the precedence constraints are satisfied. Several models for the problem, all based on Mixed Integer Linear Programming (MILP), were proposed. We refer the interested reader to Chou et al. [38].

The *Precedence-Constrained Minimum-Cost Arborescence problem with Waiting Times* (PCMCA-WT)—which is the object of the current paper—was first introduced in Chou et al. [38], where the problem was shown to be $\mathcal{NP}$-hard through a reduction to the *Rectilinear Steiner Arborescence* problem (Shi and Su [39]), and different MILP models were proposed. The problem is about retrieving an arborescence where traveling times are present among vertices and temporal precedences relative to the time of visit have to be fulfilled among pairs of vertices. Waiting times at vertices are allowed to enforce such precedences, but these waiting times are accounted for in the objective function together with travel times. The optimization is to minimize such an objective function.

The organization of the paper is as follows. The PCMCA-WT is formally defined in Section 2. Section 3 describes a new family of compact models for the problem (characterized by a polynomial number of variables and constraints). Section 4 discusses the results of a vast experimental campaign where the new models are compared to those previously disclosed in the literature. Some conclusions are the content of Section 5.

The contributions of the paper can be summarized as follows:

- New models for the PCMCA-WT that are polynomial in size and are characterized by a substantially smaller memory footprint compared to the known models are introduced. This result is achieved by exploiting some theoretical properties emerging from the current study and previously unobserved.
- The new models are solved both with MILP and Constraint Programming (CP) solvers. The experimental results substantially improve the state of the art for the instances commonly adopted in the literature. Out of the 88 open instances from the literature, improved lower bounds are provided for 71 instances and improved upper bounds are provided for 80 instances. Finally, seven instances are closed for the first time.

2. Problem Description

The PCMCA-WT can be described according to the following definitions. A directed graph $G = (V, A, R)$ is given, with $V = \{1, \ldots, n\}$ being a set of vertices and $A \subseteq V \times V$ a set of arcs, with a non-negative cost c_{ij} associated with every arc $(i, j) \in A$. It represents the traversing time for that arc. The set $R \subset V \times V$ contains precedence relationships. Let d_j be the time step at which the flow enters vertex $j \in V$, with $d_r = 0$. For any $(s, t) \in R$, we impose $d_t \geq d_s$. This implies that the flow cannot enter vertex t before entering vertex s, but it can wait at any vertex before servicing it. We define w_j as the waiting time at vertex $j \in V$, with $w_r = 0$. The waiting time at a vertex j that is visited after another vertex i in the current solution is defined as $w_j = d_j - (d_i + c_{ij})$. Given a vertex $r \in V$ being the root of the arborescence, the target of the optimization is to retrieve an arborescence T (with root r) that provides the lowest possible sum of the total cost plus the total waiting time.

An example of PCMCA-WT is provided in Figure 1. In the instance (top part of the figure), the precedence relationship $(1,3) \in R$ is represented as a dashed arrow, while in the bottom part of the figure, an optimal solution for the given instance is shown. The corresponding values of d_i and w_i are exposed near each vertex. The cost of the solution is 8 (obtained by adding the cost of the traversed arcs and the waiting times payed at vertices). In the example, observe the waiting time of 1 unit at vertex 3 ($d_1 = 4$, $d_3 = 3$ and $(1,3) \in R$).

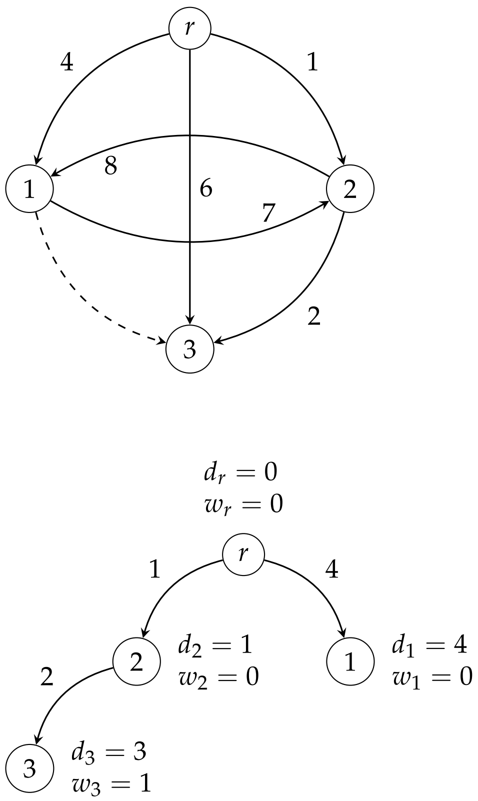

Figure 1. Instance of the PCMCA-WT problem and relative solution. The instance is depicted in the top of the figure, with its arc costs. The precedence relationship $(1,3) \in R$ is depicted through a dashed arrow. The bottom part of the figure depicts the optimal arborescence of cost 8.

3. New Compact Models

An MILP model for the PCMCA-WT that is polynomial in size was recently proposed in Chou et al. [38]. The model is based on a multicommodity flow formulation [40] that extends the flow conservation constraints in order to satisfy the precedence relationships between vertex pairs. However, the model suffers from computational limitations caused by the large number of variables and constraints, both in the order of $O(n^3)$. In this section, we derive two new models for the PCMCA-WT that are also polynomial in size. Compared to the previous models, the new ones use less variables and constraints for the description of the precedence relationships among pairs of vertices. This solves the memory issues that were encountered when using the multicommodity flow model originally introduced in [38], making compact models competitive against other solutions.

3.1. The Complete Model

We first define V^R as the set of vertices of V involved in at least one precedence relation as a head. Formally, $V^R = \{t \in V \mid \exists s \in V : (s,t) \in R\}$. Let x_{ij} be a binary variable modeling if arc $(i,j) \in A$ is visited: $x_{ij} = 1$ if $(i,j) \in T$, and 0 otherwise. Let y_i be an integer variable used to identify the position of vertex $i \in V$ along the only path of the arborescence connecting the root r to vertex i itself. Let u_j^t be a binary variable with indices representing vertex $j \in V$ and vertex $t \in V^R$. This variable will be used to model precedences. Let d_j be a continuous variable containing entering time of the flow at vertex $j \in V$. Finally, let w_j be a continuous variable modeling the time waited at vertex j before letting the flow enter the node itself.

An MILP model for the PCMCA-WT is as follows.

$$\text{minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{i \in V} w_i \tag{1}$$

$$\text{subject to:} \sum_{(i,j)\in A} x_{ij} = 1 \qquad \forall j \in V \backslash \{r\} \tag{2}$$

$$y_i - y_j + 1 \leq n(1 - x_{ij}) \qquad \forall (i,j) \in A : j \neq r \tag{3}$$

$$u_s^t = 0 \qquad \forall (s,t) \in R \tag{4}$$

$$u_t^t = 1 \qquad \forall t \in V^R \tag{5}$$

$$u_j^t - u_i^t - x_{ij} \geq -1 \qquad \forall t \in V^R, (i,j) \in A \tag{6}$$

$$d_r = 0 \tag{7}$$

$$w_r = 0 \tag{8}$$

$$d_j \geq d_i - M + (M + c_{ij})x_{ij} \qquad \forall (i,j) \in A \tag{9}$$

$$w_j \geq d_j - d_i - M + (M - c_{ij})x_{ij} \qquad \forall (i,j) \in A \tag{10}$$

$$d_t \geq d_s \qquad \forall (s,t) \in R \tag{11}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{12}$$

$$y_i \in \{0,1,\ldots,n-1\} \qquad \forall i \in V \tag{13}$$

$$u_j^t \in \{0,1\} \qquad \forall t \in V^R, j \in V \tag{14}$$

$$d_i \geq 0 \qquad \forall i \in V \tag{15}$$

$$w_i \geq 0 \qquad \forall i \in V \tag{16}$$

The objective function (1) minimizes the sum of the total travel and waiting times, as described in Section 2. The set of constraints (2) enforces that each vertex apart from the root needs to have one incoming arc. Constraints (3) model subtour elimination and dictate that any feasible solution cannot contain any cycles. The set of constraints (2) and (3) work together in order to enforce only solutions in the form of an arborescence rooted at vertex r. Constraints (4), (5) and (6) regulate precedence constraints among the vertices visited in a same branch of the tree. The logic behind these constraints will be explained in detail in Section 3.1.1, entirely devoted to this purpose. Constraints (7) and (8) initialize the distance traveled and the waiting time at the root r to 0. Constraints (9), activated once an arc $(i,j) \in A$ is selected, force the arrival time at vertex j to be not lower than the arrival time vertex i plus the travel time c_{ij}. Note that here and in the following set of constraints, M is an arbitrarily large constant. Constraints (10) push the waiting time at vertex j to be no smaller than the the service time at vertex j minus the service time at vertex i plus c_{ij}. Constraints (11) impose that the service time at vertex t cannot be smaller than the service time at vertex s for all $(s,t) \in R$. This set of constraints, in conjunction with the previous ones, regulates the value of the waiting times. Finally, constraints (12)–(16) define the domain of the variables.

The value of the large constant M, appearing in constraints (9) and (10), is an approximation by excess of the optimal cost of the problem. In our case, the solution cost of solving the instance as a Sequential Ordering Problem [34] using a nearest neighbor algorithm [41] is taken as the value of M. This is a valid upper bound for the optimal cost of a PCMCA-WT instance, being a valid solution for the Sequential Ordering Problem a simple directed path that includes all the vertices of the graph, with the constraint that t never precede s for all $(s,t) \in R$. This implies that $d_t \geq d_s$ for all $(s,t) \in R$.

The model proposed in this section has a considerably smaller memory footprint compared to the polynomial-size model proposed for the PCMCA-WT in [38]. In detail, the number of variables is reduced from $O(n^3)$ to $O(n^2)$. The number of constraints remains instead in the order of $O(n^3)$, although now the hidden multiplicative factor depends on the number of vertices involved in at least one precedence as a head, instead of all the vertices. This makes the number of constraints much smaller. All together, these improvements reduce substantially the memory footprint of the model, with great advantages for practical tractability.

3.1.1. The New Precedence Constraints

The approach proposed in this work to deal with precedences is similar to the idea originally introduced in Dell'Amico et al. [42] for the *Precedence-Constrained Minimum-Cost Arborescence* (PCMCA) problem. Constraints (4) and (5) impose the values of u_s^t and u_t^t to be 0 and 1, respectively, for all $(s,t) \in R$, and $t \in V : \exists(s,t) \in R$. On the other hand, the set of constraints (6) enforces that $u_j^t \geq u_i^t$ whenever arc $(i,j) \in A$ is selected to be part of the solution. These concepts together forbid any violation of precedence constraints along a same path of the solution arborescence T.

Figure 2 shows an example of how a precedence-violating path is detected using the set of constraints (6). In the figure, the range/value of variable u_j^t is written on the left of each vertex, and black arcs show the arcs that are part of the solution, while the red arcs show a precedence relationship $(s,t) \in R$. In the figure, constraints (6) enforce that u_1^t and u_2^t have to be greater than or equal to 1. However, once $u_s^t \geq 1$ is imposed through this logic, the constraint (4) relative to variable u_s^t is violated, rendering therefore the solution infeasible.

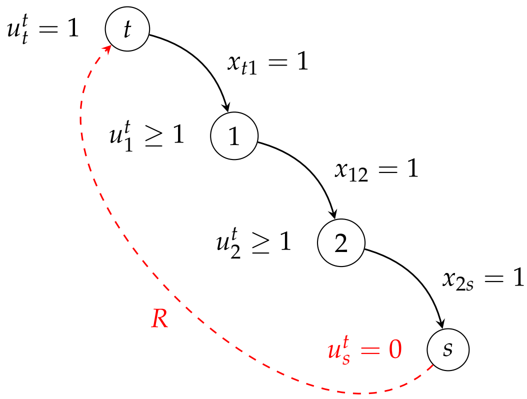

Figure 2. Example of how a precedence-violating path is detected using constraints (4)–(6).

3.2. The Reduced Model

It can be observed that by removing the set of constraints (4)–(6) from the model described in Section 3.1, the set of constraints (11) in general still enforces the precedence relationships between s and t with $(s,t) \in R$, apart from the following special case. A directed path that visits t before visiting s implies that $d_t \leq d_s$, which violates the set of constraints (11). However, the set of constraints (11) might fail to enforce a precedence relationship between s and t if a zero-cost path in G that reaches s from t exists. Therefore, variables u_j^t and constraints (4)–(6) need to be defined only for those t for which there exist at least an s for which $(s,t) \in R$ and a zero-cost path that connects t to s is available in G.

The identification of such pairs $(s,t) \in R$ for which a zero-cost path exists has, however, to be performed in a preprocessing phase, adding some complexity to the overall process. The procedure we devised for the retrieval of such pairs will be detailed in Section 3.2.1.

The reduced set of constraints can be described formally as follows. Let $G_0 = (V_0, A_0, R)$ be the graph obtained from G by considering only arcs with cost zero and the relevant nodes. Therefore, $A_0 = \{(i,j) \in A \mid c_{ij} = 0\}$ and $V_0 = \{i \in V \mid \exists j \in V : (i,j) \in A_0 \text{ or } (j,i) \in A_0\}$ (a node is considered relevant for the residual precedence constraints if it has at least an outgoing or incoming zero-cost arc). Let $SP_{ij} \subseteq A_0$ be a shortest path that starts from i to j in G, and let $c(SP_{ij}) = \sum_{(k,l) \in SP_{ij}} c_{kl}$ be its cost. For each $t \in V^R$, let $V_0(t) = \{s \in V_0 \mid \exists\,(s,t) \in R \text{ with } c(P_{ts}) = 0\}$ be the set of vertices involved in a precedence constraint with t as a head, and such that a zero-cost path from t to s exists. Finally, let $V_0^R = \{t \in V^R \mid V_0(t) \neq \emptyset\}$ be the set of vertices involved as a head in at least one precedence constraint for which an inverse zero-cost path exists.

The *Reduced* model can be obtained from the *Complete* model described in Section 3.1 by substituting constraints (4)–(6) and (14) with the following specialized version of them, characterized by a reduced domain:

$$u_s^t = 0 \qquad \forall t \in V_0^R, s \in V_0(t) \tag{17}$$

$$u_t^t = 1 \qquad \forall t \in V_0^R \tag{18}$$

$$u_j^t - u_i^t - x_{ij} \geq -1 \qquad \forall t \in V_0^R, (i,j) \in A_0 \tag{19}$$

$$u_j^t \geq 0 \qquad \forall t \in V_0^R, j \in V_0 \tag{20}$$

Notice that the domain of the u variables is reduced according to (20).

Compared to the model introduced in Section 3.1, and given that zero-cost paths are rare, the *Reduced* model uses less variables and constraints (although the theoretical complexity remains unchanged), thus further reducing the memory footprint. However, it might be characterized by a weaker linear relaxation due to the elimination of redundancy in the constraints, on top of having the burden of a preprocessing phase.

3.2.1. Selecting the Precedence Constraints involving a Zero-Cost Path

Zero-cost paths in G_0 that start from t and end in s for some $(s,t) \in R$ can be retrieved by running the procedure described in Algorithm 1.

Algorithm 1 Retrieve the Relevant Zero-Cost Paths from an Instance

```
1:  Compute the shortest path SP_ij for each pair of vertices i, j of G_0
2:  V_0^R = ∅
3:  for all t ∈ V_0 do
4:      V_0(t) = ∅
5:      for all s ∈ V_0 : (s,t) ∈ R do
6:          if C(SP_ij) = 0 then
7:              V_0(t) = V_0(t) ∪ {s}
8:          end if
9:      end for
10:     if V_0(t) ≠ ∅ then
11:         V_0^R = V_0^R ∪ {t}
12:     end if
13: end for
```

Line 1 can be implemented by running the algorithm of Floyd-Warshall [43] to retrieve the shortest path between each pair of vertices of a graph. The algorithm has a computational complexity of $O(n^3)$. Lines 2–13 scan the results to populate the sets $V_0(t)$, containing vertices involved in relevant zero-cost paths for each vertex $t \in V_0$, and the set V_0^R, with a total computational complexity of $O(n^2)$. Therefore, the overall computational complexity of the procedure remains polynomial, in the order of $O(n^3)$. This guarantees negligible computation times for the graphs considered in the study and in real applications of the problem, for which n does not exceed 700.

3.3. Solving the New Models via Constraint Programming

Modern Constraint Programming solvers such as Google OR-Tools CP-SAT [44]—the one adopted for the present work—are able to solve compact MILP models efficiently [45]. In particular, these solvers are very effective in treating logical inferences that can be expressed effectively without the use of big-M coefficients (that weaken linear relaxations and consequently worsen solving times) in their syntax. The two MILP models discussed in Sections 3.1 and 3.2 use such a technique to describe the nonlinear relation between the variable x_{ij} and the set of variables $\{y_i, u_j^t, d_j, w_j\}$ in order to turn the constraints off whenever the value of x_{ij} is equal to zero. In this section, we will therefore manipulate the models previously introduced in order to transform the constraints involving big-Ms into logical inferences. Notice that this operation mentioned above is not strictly required, since CP-SAT is able to deal with big-M constraints natively, but according to some preliminary tests, using logical inferences enhances the performance of the solver. Conversely, MILP solvers such as CPLEX [46]—the one adopted for the present work—that are also able to treat logical inferences without big-M constraints present a strong degradation of the performances when big-M constraints are removed. For this reason, in the experiments reported in Section 4, we will use big-M constraints for the MILP solver and logical inferences for the CP solver.

Finally, it can be observed that CP solvers only accept integer-valued variables, which means that the value of the c_{ij}s should be discretized before being passed to the model in case they are not integer. See Montemanni and Dell'Amico [45] for a deeper traction.

In detail, the *Complete* model can be adapted by modifying constraints (3), (6), (9) and (10), which are substituted by the following ones:

$$x_{ij} \implies y_j = y_i + 1 \qquad \forall (i,j) \in A : j \neq r \tag{21}$$

$$x_{ij} \implies u_j^t \geq u_i^t \qquad \forall t \in V^R, (i,j) \in A \tag{22}$$

$$x_{ij} \implies d_j = d_i + w_j + c_{ij} \qquad \forall (i,j) \in A \tag{23}$$

Constraints (21) implement subtour elimination. The nonlinear relationship $y_j = (y_i + 1)x_{ij}$ is modeled by setting the value of y_j to $y_i + 1$ if $x_{ij} = 1$ (true). Technically, the logical implication is implemented through the *OnlyEnforceIf* construct of the CP-SAT solver [44]. Constraints (22) are the precedence-enforcing constraints that set the value of u_j^t to be greater than or equal to u_j^t if $x_{ij} = 1$ and model the nonlinear relationship $u_j^t \geq u_i^t x_{ij}$. Constraints (23) set the value of d_j to $d_i + w_j + c_{ij}$ if $x_{ij} = 1$. The set of constraints (23) deals therefore with the nonlinear relationship $(d_j - d_i - w_j - c_{ij})x_{ij} = 0$. Notice that the two constraints (9) and (10) are now combined in a single set of constraints.

Analogously, it is possible to obtain a version of the *Reduced* model based on logical inferences by changing constraints (3) to (21) and substituting constraints (19), (9) and (10) with the following new ones:

$$x_{ij} \implies u_j^t \geq u_i^t \qquad \forall t \in V_0^R, (i,j) \in A_0 \tag{24}$$

$$x_{ij} \implies d_j = d_i + w_j + c_{ij} \qquad \forall (i,j) \in A_0 \tag{25}$$

Notice that constraints (24) and (25) are the versions of (22) and (23) specialized to the reduced graph G_0 for what concerns zero-cost precedences and that constraints (25) cover again both the sets (9) and (10).

4. Computational Experiments

The experimental settings and conditions are described in Section 4.1 together with the instances adopted for the tests. The detailed results are instead presented and commented on in Section 4.2.

4.1. Experimental Settings

The computational experiments we present in order to position the proposed models within the existing literature are based on the benchmark instances of TSPLIB [47], SOPLIB [48] and COMPILERS [49]. All these datasets had originally been proposed for the Sequential Ordering Problem, and they are commonly adopted in the PCMCA-WT literature so far, see [38]. In total, the benchmark sets considered contain 116 instances with sizes ranging between 9 and 700 vertices (and an average of 248 vertices). Currently, the benchmark set has a total of 88 open instances (i.e., without a known optimal solution).

The MILP solver adopted is CPLEX v12.8 [46] with standard settings. The CP solver used is OR-Tools v9.5 [44] CP-SAT, also run with standard settings. The computation time of both solvers is limited to 1 h, while the preprocessing time required for the *Reduced* methods is not accounted for, being in the order of fractions of a second for all the instances considered.

The best-known solutions used as reference are those appearing in [38]. The value reported is for each instance the best of the results achieved by the three models introduced there. This biases the comparison in favor of the old methods, since—according to the *No Free Lunch* Theorem [50]—taking the best of different approaches might give a substantial advantage. Among the three ideas disclosed in [38], the one improved in the present work had shown some potential, however, it was the weakest of the set due to severe scalability issues (now solved, see Section 3). On the other hand, the results of [38] were obtained on an Intel i7-8550U processor running at 1.8 GHz and with 8 GB of RAM, while the new experiments are obtained on Intel Xeon Platinum 8375C running at 2.9 GHz and using up to 16 GB of RAM. This gives a hardware advantage to the new models, somehow balancing back the comparison. Notice that the newly proposed models are a direct improvement aiming at overcoming the crucial scalability of one of the three models of [38], making computational fairness considerations less central, in our view.

4.2. Results

An aggregated summary of the results is presented in Table 1. The average optimality gap across all the instances for which all the models were able to find a feasible/optimal solution is reported under *Average optimality gap*. The average solution time among all the instances that were solved to optimality by all the models can be found under *Average solution time*. The detailed results of each model can be found instead in Tables 2–4, where the following data are reported for each instance. The name and size can be found in the columns with these names. The best-known bounds found in [38] are reported in the column *Best-Known* [38], where *LB* shows the best lower bound found, and *UB* the best-known solution. For each model, the following columns are displayed. The lower and upper bound can be found in the column with these names. The optimality gap, computed as $\frac{UB-LB}{UB}$, is reported in the column *Gap*. The solution time in seconds, which is reported only for those instances that are closed in the given time, can be found in the column *Time*. Entries in bold indicate new best-known lower or upper bounds. Finally, the name of the instances for which optimality is proven for the first time in this work is highlighted in bold across the tables.

Comparing the average optimality gap of each model, it can be observed that the MILP solver run on the *Complete* model has an optimality gap of 0.206 on average (0.418 when all the instances solved by the model are considered) but fails to solve two instances, as it runs out of memory. The MILP solver on the *Reduced* model has an optimality gap of 0.153 on average (with a 25.7% improvement over the previous model) and an optimality gap of 0.340 on average (an 18.7% improvement) across all the instances. The MILP solver on the *Reduced* model also runs out of memory on one instance. When considering the CP solver, the *Complete* model shows an optimality gap of 0.159 on average (with a 22.8% improvement), with an optimality gap of 0.157 on average across all the instances. However, the largest instances, with size larger than 200 or with a very dense precedence graph, are not solved, since the model runs out of memory due to the large number of constraints (19). When solving the *Reduced* model, the CP solver achieves an optimality gap of 0.122 on average (with a 40.7% improvement), with an optimality gap of 0.286 averaged across all the instances.

Table 1. Summary of the results achieved with each solver/model combination.

	MILP Solver		**CP Solver**	
	Complete Model	**Reduced Model**	**Complete Model**	**Reduced Model**
Average optimality gap	0.206	0.153	0.159	0.122
Average solution time	690.8	270.8	166.4	36.4
New best-known lower bounds	2	24	13	32
New best-known upper bounds	1	15	10	54
New optimal solutions	0	0	7	7

In terms of solution time, and comparing the instances that are optimally solved by all models (27 instances), using the MILP solver takes the *Complete* model to a solution time of 690.8 s on average, while the *Reduced* takes 270.8 s on average (with a 60.8% improvement). When the CP solver is used, the *Complete* model has a solution time of 166.4 s on average (with a 75.9% improvement), while solving the *Reduced* model takes 36.4 s on average (with a 94.7% improvement). Cross-comparing a same model when treated by the two different solvers, it emerges that generally the CP models outperform the MILP models on instances with medium to high density precedence graphs.

In terms of solution costs, the *Reduced* model solved by an MILP solver finds new best-known lower bounds for 24 out of 88 instances (27.3%) compared to the 2 retrieved by the *Complete* model solved by the same solver. Furthermore, the *Reduced* model finds new best-known upper bounds for 15 (17.1%) compared to the 1 only found by the *Complete* model. This indicates that the strength of the linear relaxation of the model is not drastically affected after removing a subset of the variables and constraints from the model. Furthermore, this shows that the *Reduced* model is generally easier to solve by the MILP solver adopted, and therefore the solver is able to find new bounds more frequently compared to solving the *Complete* model. When considering the CP solver, the *Reduced* model finds new best-known lower bounds for 32 (36.4%), while the *Complete* model finds new lower bounds for 13 (14.8%). For new best-known upper bounds, solving the *Reduced* model leads to new 54 new bests (61.4%), while solving the *Complete* model leads to 10 new bests (11.4%). This indicates that the *Reduced* model is generally more effective to solve by the CP solver when compared to the *Complete* model. Moreover, the use of the CP solver led to seven newly proven optimal solutions. In general, using the MILP solver seems to produce better lower bounds, while the CP solver is better at finding lower cost solutions.

In summary, the computational results show that the *Reduced* model generally outperforms the *Complete* model independently of the solver adopted. This is due to the fact that the *Reduced* model has a substantially smaller number of variables and constraints, giving an advantage to the solvers. Moreover, the CP solver performs better than the MILP solver in terms of the quality of the solutions, the average solution time and the average optimality gap. Furthermore, the CP solver finds new best-known lower/upper bounds for some instances.

Table 2. Computational results for TSPLIB instances.

			MILP Solver								CP Solver							
Instance			Complete Model				Reduced Model				Complete Model				Reduced Model			
Name	**Size**	**Best-Known [38]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**
br17.10	18	[35, 44]	39	44	0.114	-	40	44	0.091	-	**44**	**44**	0.000	46.629	**44**	**44**	0.000	56.628
br17.12	18	[35, 44]	41	44	0.068	-	41	44	0.068	-	**44**	**44**	0.000	22.604	**44**	**44**	0.000	44.550
ESC07	9	1906	1906	1906	0.000	0.028	1906	1906	0.000	0.070	1906	1906	0.000	0.023	1906	1906	0.000	0.025
ESC11	13	2174	2174	2174	0.000	0.125	2174	2174	0.000	0.114	2174	2174	0.000	0.107	2174	2174	0.000	0.077
ESC12	14	1138	1138	1138	0.000	0.035	1138	1138	0.000	0.030	1138	1138	0.000	0.034	1138	1138	0.000	0.037
ESC25	27	1158	1158	1158	0.000	6.185	1158	1158	0.000	1.945	1158	1158	0.000	0.910	1158	1158	0.000	0.833
ESC47	49	747	747	747	0.000	59.760	747	747	0.000	22.153	747	747	0.000	3.886	747	747	0.000	2.708
ESC63	65	56	56	56	0.000	24.600	56	56	0.000	57.347	56	56	0.000	1.517	56	56	0.000	2.465
ESC78	80	1196	1196	1196	0.000	2410.483	1196	1196	0.000	257.609	1196	1196	0.000	100.511	1196	1196	0.000	18.971
ft53.1	54	4089	4089	4089	0.000	1764.235	4089	4089	0.000	2023.553	4089	4089	0.000	215.480	4089	4089	0.000	291.099
ft53.2	54	[4135, 4284]	4112	4317	0.047	-	**4161**	4334	0.040	-	4102	4284	0.042	-	4103	4284	0.042	-
ft53.3	54	[4623, 5457]	4746	5425	0.125	-	**4799**	**5279**	0.091	-	4493	60	0.161	-	4508	5484	0.178	-
ft53.4	54	[5657, 6439]	5922	**6420**	0.078	-	**5923**	**6420**	0.077	-	5338	6502	0.179	-	5357	**6420**	0.166	-
ft70.1	71	[33,128, 33,298]	32,777	33,308	0.016	-	32,827	33,308	0.014	-	32,669	33,472	0.024	-	33,101	33,298	0.006	-
ft70.2	71	[33,357, 34,450]	33,057	33,977	0.027	-	33,089	33,916	0.024	-	32,938	**33,670**	0.022	-	32,897	**33,670**	0.023	-
ft70.3	71	[33,914, 42,732]	34,152	38,546	0.114	-	**34,423**	38,351	0.102	-	33,825	36,939	0.084	-	33,813	**36,932**	0.084	-
ft70.4	71	[36,517, 40,404]	36,737	39,145	0.062	-	**36,850**	38,771	0.050	-	33,825	**36,939**	0.084	-	35,664	39,843	0.105	-
rbg048a	50	[261, 264]	260	265	0.019	-	259	264	0.019	-	**263**	**263**	0.000	9.442	**263**	**263**	0.000	25.294
rbg050c	52	225	225	225	0.000	863.662	225	225	0.000	36.673	225	225	0.000	2.575	225	225	0.000	1.234
rbg109	111	[354, 414]	354	426	0.169	-	**366**	407	0.101	-	357	488	0.268	-	359	**401**	0.105	-
rbg150a	152	[447, 541]	447	511	0.125	-	461	**509**	0.094	-	**463**	591	0.217	-	461	517	0.108	-
rbg174a	176	[446, 580]	452	601	0.248	-	**463**	**553**	0.163	-	457	571	0.200	-	461	572	0.194	-
rbg253a	255	[477, 773]	523	1252	0.582	-	**532**	**718**	0.259	-	-	-	-	-	527	722	0.270	-
rbg323a	325	[926, 4035]	981	10,111	0.903	-	974	2466	0.605	-	-	-	-	-	**1009**	**1891**	0.466	-
rbg341a	343	[681, 3800]	764	9313	0.918	-	761	2907	0.738	-	-	-	-	-	**780**	**1457**	0.465	-
rbg358a	360	[706, 3296]	950	11,528	0.918	-	755	2453	0.692	-	-	-	-	-	**788**	**1150**	0.315	-
rbg378a	380	[649, 2759]	672	10,242	0.934	-	648	2191	0.704	-	-	-	-	-	**678**	**1126**	0.398	-
kro124p.1	101	[32,858, 35,231]	32,651	37,120	0.120	-	32,630	36,099	0.096	-	32,504	34,100	0.047	-	32,561	**33,962**	0.041	-
kro124p.2	101	[33,190, 37,956]	32,886	42,573	0.228	-	33,006	39,931	0.173	-	32,764	37,074	0.116	-	32,799	**35,860**	0.085	-
kro124p.3	101	[34,217, 53,988]	33,813	54,183	0.376	-	34,005	46,764	0.273	-	33,561	43,910	0.236	-	33,488	**42,416**	0.210	-
kro124p.4	101	[39,413, 55,187]	39,969	58,944	0.322	-	39,333	53,456	0.264	-	38,433	50,910	0.245	-	37,676	**49,590**	0.240	-
p43.1	44	[2827, 4470]	2660	4085	0.349	-	2656	3955	0.328	-	**2860**	**3955**	0.277	-	2851	3990	0.285	-
p43.2	44	[2826, 4275]	991	4450	0.777	-	2705	4210	0.357	-	2856	**4160**	0.313	-	**2870**	4180	0.313	-

Table 2. *Cont.*

Instance			MILP Solver								CP Solver							
			Complete Model				Reduced Model				Complete Model				Reduced Model			
Name	**Size**	**Best-Known [38]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**
p43.3	44	[2864, 5375]	1067	5015	0.787	-	1383	4440	0.689	-	**2966**	4450	0.333	-	2897	**4255**	0.319	-
p43.4	44	[3101, 4900]	2995	5035	0.405	-	**3125**	4605	0.321	-	3090	**4495**	0.313	-	3094	4620	0.330	-
prob.100	100	[674, 1008]	668	2125	0.686	-	**677**	741	0.086	-	666	784	0.151	-	667	**738**	0.096	-
prob.42	42	171	171	171	0.000	396.458	171	171	0.000	230.506	171	171	0.000	79.667	171	171	0.000	34.245
ry48p.1	49	[13,371, 13,722]	13,114	14,272	0.081	-	13,200	**13,670**	0.034	-	13,036	**13,670**	0.046	-	13,061	**13,670**	0.045	-
ry48p.2	49	[13,508, 14,659]	13,299	14,415	0.077	-	13,336	**14,305**	0.068	-	13,216	**14,305**	0.076	-	13,185	**14,305**	0.078	-
ry48p.3	49	[14,371, 16,326]	13,882	16,193	0.143	-	13,994	15,840	0.117	-	13,764	15,546	0.115	-	13,728	**15,477**	0.113	-
ry48p.4	49	[17,339, 19,649]	17,162	19,744	0.131	-	17,180	19,583	0.123	-	16,550	19,837	0.166	-	16,483	**19,495**	0.155	-
Average					0.158	552.557			0.167	263.000			0.103	37.184			0.128	36.782

Table 3. Computational results for SOPLIB instances.

Instance			MILP Solver								CP Solver							
			Complete Model				Reduced Model				Complete Model				Reduced Model			
Name	**Size**	**Best-Known [38]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**
R.200.100.1	200	29	29	29	0.000	18.394	29	29	0.000	6.017	29	29	0.000	28.271	29	29	0.000	31.403
R.200.100.15	200	[505, 1271]	497	1431	0.653	-	525	1033	0.492	-	381	1864	0.796	-	**589**	**979**	0.398	-
R.200.100.30	200	[669, 2011]	686	3252	0.789	-	774	**1761**	0.560	-	451	3001	0.850	-	**838**	1871	0.552	-
R.200.100.60	200	[8070, 18,761]	8760	17,004	0.485	-	**8861**	16,930	0.477	-	6018	31,561	0.809	-	8440	**16,197**	0.479	
R.200.1000.1	200	887	887	887	0.000	1288.092	887	887	0.000	15.635	887	887	0.000	649.979	887	887	0.000	26.0915
R.200.1000.15	200	[6665, 16,496]	6769	16,336	0.586	-	6895	**12,601**	0.453	-	5318	25,196	0.789	-	**7231**	12,812	0.436	-
R.200.1000.30	200	[9340, 30,351]	9937	23,226	0.572	-	**10,512**	**22,781**	0.539	-	7381	38,410	0.808	-	10,120	23,249	0.565	-
R.200.1000.60	200	[10,508, 23,748]	11,399	21,706	0.475	-	**12,042**	21,993	0.452	-	6666	28,522	0.766	-	10,665	**19,934**	0.465	
R.300.100.1	300	13	13	13	0.000	37.352	13	13	0.000	35.012	13	13	0.000	205.731	13	13	0.000	56.4263
R.300.100.15	300	[625, 12,903]	660	6958	0.905	-	669	2259	0.704	-	-	-	-	-	**811**	**2056**	0.606	-
R.300.100.30	300	[948, 3767]	1008	6790	0.852	-	1102	3163	0.652	-	-	-	-	-	**1157**	**2590**	0.553	-
R.300.100.60	300	[824, 3005]	919	4732	0.806	-	949	1954	0.514	-	-	-	-	-	**991**	**1865**	0.469	-
R.300.1000.1	300	715	715	715	0.000	3187.049	715	715	0.000	64.683	715	715	0.000	257.074	715	715	0.000	71.6789
R.300.1000.15	300	[7213, 112,424]	7607	110,366	0.931	-	7832	**24,047**	0.674	-	-	-	-	-	**8768**	29,423	0.702	-
R.300.1000.30	300	[10,385, 40,457]	11,179	53,835	0.792	-	12,071	40,863	0.705	-	-	-	-	-	**12,269**	**31,618**	0.612	-
R.300.1000.60	300	[9413, 30,655]	10,180	38,212	0.734	-	10,275	25,323	0.594	-	-	-	-	-	**10,408**	**21,623**	0.519	-

Table 3. *Cont.*

Instance			MILP Solver								CP Solver							
			Complete Model				Reduced Model				Complete Model				Reduced Model			
Name	**Size**	**Best-Known [38]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**
R.400.100.1	400	6	6	376	0.984	-	6	6	0.000	995.137	6	6	0.000	726.057	6	6	0.000	97.3851
R.400.100.15	400	[729, 47,117]	781	35,044	0.978	-	856	22,767	0.962	-	-	-	-	-	**963**	**3591**	0.732	-
R.400.100.30	400	[780, 7243]	911	39,022	0.977	-	1010	26,438	0.962	-	-	-	-	-	**1084**	**3061**	0.646	-
R.400.100.60	400	[731, 5545]	837	3309	0.747	-	861	2652	0.675	-	-	-	-	-	**966**	**2069**	0.533	-
R.400.1000.1	400	780	780	780	0.000	161.021	780	780	0.000	124.990	780	780	0.000	208.525	780	780	0.000	90.9555
R.400.1000.15	400	[7760, 501,543]	8357	85,878	0.903	-	9083	85,878	0.894	-	-	-	-	-	**9976**	**35,160**	0.716	-
R.400.1000.30	400	[10,076, 95,523]	11,030	127,290	0.913	-	11,783	127,290	0.907	-	-	-	-	-	**12,337**	**57,272**	0.785	-
R.400.1000.60	400	[8103, 55,950]	9360	65,615	0.857	-	9877	36,662	0.731	-	-	-	-	-	**9954**	**22,376**	0.555	-
R.500.100.1	500	3	3	3	0.000	2157.743	3	3	0.000	1881.297	3	3	0.000	2333.235	3	3	0.000	112.086
R.500.100.15	500	[924, 11,452]	964	11,452	0.916	-	1018	11,452	0.911	-	-	-	-	-	**1250**	**5508**	0.773	-
R.500.100.30	500	[773, 12,225]	849	16,963	0.950	-	976	14,273	0.932	-	-	-	-	-	**1099**	**4841**	0.773	-
R.500.100.60	500	[669, 8427]	840	49,105	0.983	-	840	6357	0.868	-	-	-	-	-	**931**	**2723**	0.658	-
R.500.1000.1	500	297	297	297	0.000	97.473	297	297	0.000	85.459	297	297	0.000	85.281	297	297	0.000	77.4382
R.500.1000.15	500	[8420, 107,776]	8949	107,776	0.917	-	9461	107,776	0.912	-	-	-	-	-	**10,628**	**45,356**	0.766	
R.500.1000.30	500	[10,431, 181,835]	11,799	156,359	0.925	-	**12,694**	156,359	0.919	-	-	-	-	-	12,576	**57,330**	0.781	
R.500.1000.60	500	[7094, 33,260]	**8233**	112,466	0.927	-	8192	45,696	0.821	-	-	-	-	-	6559	**20,465**	0.680	-
R.600.100.1	600	[1, 379]	1	55	0.982	-	1	55	0.982	-	**1**	**1**	0.000	2710.470	**1**	**1**	0.000	2182.18
R.600.100.15	600	[670, 5949]	714	5931	0.880	-	845	4044	0.791	-	-	-	-	-	**938**	**2443**	0.616	-
R.600.100.30	600	[873, 12,875]	945	18,932	0.950	-	**1099**	18,932	0.942	-	-	-	-	-	740	**6467**	0.886	-
R.600.100.60	600	[751, 7893]	**838**	26,732	0.969	-	778	25,214	0.969	-	-	-	-	-	538	**2494**	0.784	-
R.600.1000.1	600	322	322	322	0.000	352.202	322	322	0.000	140.645	322	322	0.000	127.397	322	322	0.000	103.378
R.600.1000.15	600	[10,181, 121,877]	10,753	121,877	0.912	-	**10,915**	121,877	0.910	-	-	-	-	-	9401	**65,039**	0.855	-
R.600.1000.30	600	[10,151, 151,010]	11,352	190,145	0.940	-	**12,431**	190,145	0.935	-	-	-	-	-	9356	**48,775**	0.808	-
R.600.1000.60	600	[7604, 87,770]	7962	256,464	0.969	-	**8162**	75,269	0.892	-	-	-	-	-	6908	**42,652**	0.838	-
R.700.100.1	700	2	-	-	-	-	-	-	-	-	2	2	0.000	1649.486	2	2	0.000	619.22
R.700.100.15	700	[799, 6561]	815	14,478	0.944	-	**972**	5718	0.830	-	-	-	-	-	655	**2759**	0.763	-
R.700.100.30	700	[762, 20,281]	896	6960	0.871	-	**983**	4218	0.767	-	-	-	-	-	588	**2531**	0.768	-
R.700.100.60	700	[516, 9030]	538	7033	0.924	-	**555**	1854	0.701	-	-	-	-	-	383	**1598**	0.760	-
R.700.1000.1	700	[611, 621]	611	616	0.008	-	611	616	0.008	-	**611**	**611**	0.000	592.107	**611**	**611**	0.000	368.139
R.700.1000.15	700	[4636, 147,321]	4375	147,321	0.970	-	**5136**	7145	0.281	-	-	-	-	-	2787	**6315**	0.559	-
R.700.1000.30	700	[4303, 50,000]	4477	32,742	0.863	-	**4827**	6981	0.309	-	-	-	-	-	2658	**6115**	0.565	-
R.700.1000.60	700	[2857, 15,579]	2942	8534	0.655	-	**2997**	5842	0.487	-	-	-	-	-	1913	**5357**	0.643	-
Average					0.689	912.416			0.577	372.097			0.268	797.801			0.492	319.698

Table 4. Computational results for COMPILERS instances.

			MILP Solver								CP Solver							
Instance			Complete Model				Reduced Model				Complete Model				Reduced Model			
Name	**Size**	**Best-Known [38]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**	**LB**	**UB**	**Gap**	**Time [s]**
gsm.153.124	126	[246, 313]	257	312	0.176	-	269	**311**	0.135	-	278	317	0.123	-	**280**	**311**	0.100	-
gsm.444.350	353	[2103, 2873]	2294	4878	0.530	-	2405	4856	0.505	-	-	-	-	-	**2456**	4310	0.430	-
gsm.462.77	79	[396, 488]	402	478	0.159	-	402	477	0.157	-	**419**	474	0.116	-	418	**465**	0.101	-
jpeg.1483.25	27	87	87	87	0.000	26.041	87	87	0.000	18.556	87	87	0.000	1.194	87	87	0.000	1.071
jpeg.3184.107	109	[489, 684]	506	656	0.229	-	510	715	0.287	-	**518**	718	0.279	-	517	692	0.253	-
jpeg.3195.85	87	[22, 25]	17	25	0.320	-	17	25	0.320	-	**23**	25	0.080	-	22	25	0.120	-
jpeg.3198.93	95	[172, 204]	180	188	0.043	-	180	**188**	0.043	-	**181**	**188**	0.037	-	**181**	**188**	0.037	-
jpeg.3203.135	137	[600, 750]	602	980	0.386	-	618	751	0.177	-	**629**	913	0.311	-	626	750	0.165	-
jpeg.3740.15	17	33	33	33	0.000	1.523	33	33	0.000	0.839	33	33	0.000	0.157	33	33	0.000	0.095
jpeg.4154.36	38	90	90	90	0.000	556.798	90	90	0.000	60.924	90	90	0.000	1.272	90	90	0.000	1.764
jpeg.4753.54	56	164	164	164	0.000	2753.752	164	164	0.000	1790.269	164	164	0.000	15.342	164	164	0.000	16.877
susan.248.197	199	[736, 1184]	792	1978	0.600	-	802	1370	0.415	-	**805**	1361	0.409	-	780	1320	0.409	-
susan.260.158	160	[564, 876]	568	937	0.394	-	573	938	0.389	-	596	991	0.399	-	**598**	897	0.333	-
susan.343.182	184	[591, 862]	617	798	0.227	-	622	**776**	0.198	-	**636**	1043	0.390	-	632	792	0.202	-
typeset.10192.123	125	[280, 415]	274	429	0.361	-	282	**379**	0.256	-	293	385	0.239	-	292	387	0.245	-
typeset.10835.26	28	[99, 112]	99	111	0.108	-	100	112	0.107	-	**110**	**111**	0.009	-	109	**111**	0.018	-
typeset.12395.43	45	[143, 146]	140	146	0.041	-	141	146	0.034	-	**146**	**146**	0.000	2181.942	**146**	**146**	0.000	2780.121
typeset.15087.23	25	97	97	97	0.000	60.502	97	97	0.000	29.118	97	97	0.000	0.477	97	97	0.000	0.318
typeset.15577.36	38	125	125	125	0.000	286.210	125	125	0.000	43.164	125	125	0.000	2.116	125	125	0.000	1.713
typeset.16000.68	70	[77, 80]	66	81	0.185	-	66	80	0.175	-	**79**	80	0.013	-	71	80	0.113	-
typeset.1723.25	27	60	60	60	0.000	590.577	60	60	0.000	86.068	60	60	0.000	4.013	60	60	0.000	3.469
typeset.19972.246	248	[1325, 1929]	1422	3562	0.601	-	1452	2509	0.421	-	1519	2961	0.487	-	**1525**	2804	0.456	-
typeset.4391.240	242	[1093, 1412]	1108	2595	0.573	-	1137	2476	0.541	-	1149	2511	0.542	-	**1154**	1905	0.394	-
typeset.4597.45	47	[150, 155]	150	154	0.026	-	151	154	0.019	-	**154**	**154**	0.000	209.659	**154**	**154**	0.000	128.916
typeset.4724.433	435	[2460, 3433]	-	-	-	-	2673	6131	0.564	-	-	-	-	-	**2679**	7194	0.628	-
typeset.5797.33	35	113	113	113	0.000	851.490	113	113	0.000	28.504	113	113	0.000	0.547	113	113	0.000	0.574
typeset.5881.246	248	[1305, 1700]	1378	2258	0.390	-	1396	2426	0.425	-	**1406**	2385	0.410	-	1394	2084	0.331	-
Average					0.206	640.862			0.191	257.180			0.154	241.672			0.161	293.492

5. Conclusions

This work introduced new models for the Precedence-Constrained Minimum-Cost Arborescence Problem with Waiting-Times that are polynomial in size and are characterized by a smaller memory footprint with respect to the polynomial-sized models previously proposed in the literature. A first model is based on a new set of variables to model precedences. The number of variables and constraints are further reduced, at the price of a preprocessing phase, in a second model. The two models are solved both by Mixed Integer Linear Programs and Constraint Programming solvers. The computational results show that the model characterized by the need of preprocessing outperforms the other one. Furthermore, the Constraint Programming solver achieves the best overall results in terms of both optimality gap and solution time. However, the Mixed Integer Linear Programming solver generally finds better lower bound estimates on the instances. Finally, the models proposed were able to close 7 new instances that were previously open, to provide improved lower bounds for 71 instances, and to find improved upper bounds for 80 instances, out of a total of 88 open instances.

Future work should cover aspects such as robustness of the approaches and the addition to the models of other realistic constraints. Given the progress on the solvers, new instances should be also introduced in order to extend the study on scalability of the different models. Finally, a deeper analysis of the characteristics of the instances that mainly affect the different approaches presented should be in order.

Author Contributions: Conceptualization, J.J., R.M. and M.D.; methodology, J.J.; software, J.J.; validation, J.J. and R.M.; formal analysis, J.J., R.M. and M.D.; investigation, J.J. and R.M.; resources, R.M.; data curation, J.J.; writing—original draft preparation, J.J. and R.M.; writing—review and editing, R.M. and M.D.; visualization, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The instances used in the paper are available from the literature.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chu, Y.J.; Liu, T. On the shortest arborescence of a directed graph. *Sci. Sin.* **1965**, *14*, 1396–1400.
2. Edmonds, J. Optimum branchings. *J. Res. Natl. Bur. Stand.* **1967**, *71*, 233–240. [CrossRef]
3. Gabow, H.N.; Galil, Z.; Spencer, T.; Tarjan, R.E. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica* **1986**, *6*, 109–122. [CrossRef]
4. Chou, X.; Gambardella, L.M.; Montemanni, R. A tabu search algorithm for the probabilistic orienteering problem. *Comput. Oper. Res.* **2021**, *126*, 105107. [CrossRef]
5. Bock, F. An algorithm to construct a minimum directed spanning tree in a directed network. *Dev. Oper. Res.* **1971**, 29–44.
6. Fischetti, M.; Vigo, D. A branch-and-cut algorithm for the resource-constrained minimum-weight arborescence problem. *Netw. Int. J.* **1997**, *29*, 55–67. [CrossRef]
7. Pereira, A.H.; Mateus, G.R.; Urrutia, S. Branch-and-cut algorithms for the *p*-arborescence star problem. *Int. Trans. Oper. Res.* **2021**, *29*, 2374–2400. [CrossRef]
8. Morais, V.; Gendron, B.; Mateus, G.R. The p-arborescence star problem: Formulations and exact solution approaches. *Comput. Oper. Res.* **2019**, *102*, 91–101. [CrossRef]
9. Hakimi, S.L. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Oper. Res.* **1965**, *13*, 462–475. [CrossRef]
10. Guttmann-Beck, N.; Hassin, R. On two restricted ancestors tree problems. *Inf. Process. Lett.* **2010**, *110*, 570–575. [CrossRef]
11. Carrabs, F.; Gaudioso, M. A Lagrangian approach for the minimum spanning tree problem with conflicting edge pairs. *Networks* **2021**, *78*, 32–45. [CrossRef]
12. Kruskal, J.B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc.* **1956**, *7*, 48–50. [CrossRef]
13. Gouveia, L.; Lopes, M.J. The capacitated minimum spanning tree problem: On improved multistar constraints. *Eur. J. Oper. Res.* **2005**, *160*, 47–62. [CrossRef]
14. Frieze, A.M.; Tkocz, T. A Randomly Weighted Minimum Arborescence with a Random Cost Constraint. *Math. Oper. Res.* **2021**, *47*, 1664–1680. [CrossRef]

15. Fertin, G.; Fradin, J.; Jean, G. Algorithmic Aspects of the Maximum Colorful Arborescence Problem. In *Theory and Applications of Models of Computation*; TAMC 2017; Springer: Cham, Switzerland, 2017; pp. 216–230.
16. Eswaran, K.P.; Tarjan, R.E. Augmentation problems. *SIAM J. Comput.* **1976**, *5*, 653–665. [CrossRef]
17. Li, J.; Liu, X.; Lichen, J. The constrained arborescence augmentation problem in digraphs. In Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 13–16 December 2017; pp. 1204–1209.
18. Kawatra, R.; Bricker, D. Design of a degree-constrained minimal spanning tree with unreliable links and node outage costs. *Eur. J. Oper. Res.* **2004**, *156*, 73–82. [CrossRef]
19. Galbiati, G.; Gualandi, S.; Maffioli, F. On minimum changeover cost arborescences. *Lect. Notes Comput. Sci.* **2011**, *6630*, 112–123.
20. Bérczi, K.; Fujishige, S.; Kamiyama, N. A linear-time algorithm to find a pair of arc-disjoint spanning in-arborescence and out-arborescence in a directed acyclic graph. *Inf. Process. Lett.* **2009**, *109*, 1227–1231. [CrossRef]
21. Bang-Jensen, J. Edge-disjoint in- and out-branchings in tournaments and related path problems. *J. Comb. Theory—Ser. B* **1991**, *51*, 1–23. [CrossRef]
22. Li, Y.; Thai, M.; Wang, F.; Du, D.Z. On the construction of a strongly connected broadcast arborescence with bounded transmission delay. *IEEE Trans. Mob. Comput.* **2006**, *5*, 1460–1470.
23. Carrabs, F.; Cerulli, R.; Pentangelo, R.; Raiconi, A. Minimum spanning tree with conflicting edge pairs: A branch-and-cut approach. *Ann. Oper. Res.* **2021**, *298*, 65–78. [CrossRef]
24. Darmann, A.; Pferschy, U.; Schauer, J. Determining a Minimum Spanning Tree with Disjunctive Constraints. In *Algorithmic Decision Theory*; Springer: Berlin/Heidelberg, Germeny, 2009; pp. 414–423.
25. Viana, L.A.d.C.; Campêlo, M. Two dependency constrained spanning tree problems. *Int. Trans. Oper. Res.* **2020**, *27*, 867–898. [CrossRef]
26. Escudero, L. An inexact algorithm for the sequential ordering problem. *Eur. J. Oper. Res.* **1988**, *37*, 236–249. [CrossRef]
27. Moon, C.; Kim, J.; Choi, G.; Seo, Y. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *Eur. J. Oper. Res.* **2002**, *140*, 606–617. [CrossRef]
28. Balas, E.; Fischetti, M.; Pulleyblank, W. The precedence-constrained asymmetric traveling salesman polytope. *Math. Program.* **1995**, *68*, 241–265. [CrossRef]
29. Hernádvölgyi, I. Solving the sequential ordering problem with automatically generated lower bounds. In *Operations Research Proceedings 2003*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 355–362.
30. Escudero, L.; Guignard, M.; Malik, K. A Lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships. *Ann. Oper. Res.* **1994**, *50*, 219–237. [CrossRef]
31. Gambardella, L.; Dorigo, M. An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS J. Comput.* **2000**, *12*, 237–255. [CrossRef]
32. Karan, M.; Skorin-Kapov, N. A branch and bound algorithm for the sequential ordering problem. In Proceedings of the MIPRO, 2011 Proceedings of the 34th International Convention, Opatija, Croatia, 23–27 May 2011; pp. 452–457.
33. Ascheuer, N.; Escudero, L.; Grötschel, M.; Stoer, M. A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). *SIAM J. Optim.* **1993**, *3*, 25–42. [CrossRef]
34. Ascheuer, N.; Jünger, M.; Reinelt, G. A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Comput. Optim. Appl.* **2000**, *17*, 61–84.
35. Montemanni, R.; Smith, D.H.; Gambardella, L.M. Ant colony systems for large sequential ordering problems. In Proceedings of the IEEE Swarm Intelligence Symposium (SIS), Honolulu, HI, USA, 1–5 April 2007; pp. 60–67.
36. Fiala Timlin, M.; Pulleyblank, W. Precedence constrained routing and helicopter scheduling: Heuristic design. *Interfaces* **1992**, *22*, 100–111. [CrossRef]
37. Dell'Amico, M.; Jamal, J.; Montemanni, R. A mixed integer linear program for a precedence-constrained minimum-cost arborescence problem. In Proceedings of the 8th International Conference on Industrial Engineering and Applications (Europe), Online, 8–11 January 2021; pp. 216–221.
38. Chou, X.; Dell'Amico, M.; Jamal, J.; Montemanni, R. Precedence-Constrained Arborescences. *Eur. J. Oper. Res.* **2022**, *307*, 575–589. [CrossRef]
39. Shi, W.; Su, C. The rectilinear Steiner arborescence problem is NP-complete. *SIAM J. Comput.* **2005**, *35*, 729–740. [CrossRef]
40. Wang, I.L. Multicommodity network flows: A survey, Part I: Applications and Formulations. *Int. J. Oper. Res.* **2018**, *15*, 145–153.
41. Hurkensa, C.A.J.; Woeginger, G.J. On the nearest neighbor rule for the traveling salesman problem. *Oper. Res. Lett.* **2004**, *32*, 1–4. [CrossRef]
42. Dell'Amico, M.; Jamal, J.; Montemanni, R. Compact Models for the Precedence-Constrained Minimum-Cost Arborescence Problem. In Proceedings of the 2022 The 6th International Conference on Intelligent Traffic and Transportation (ICITT), Paris, France, 25–27 September 2022; IOS Press: Amsterdam, The Netherlands, 2023; pp. 112–126.
43. Floyd, R. Algorithm 97: Shortest Path. *Commun. ACM* **1962**, *5*, 345. [CrossRef]
44. Google. Google OR-Tools. 2023. Available online: https://developers.google.com/optimization (accessed on 20 November 2023).
45. Montemanni, R.; Dell'Amico, M. Solving the Parallel Drone Scheduling Traveling Salesman Problem via Constraint Programming. *Algorithms* **2023**, *16*, 40. [CrossRef]
46. IBM. IBM CPLEX Optimizer. 2023. Available online: https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer (accessed on 20 November 2023).

47. Reinelt, G. TSPLIB–A travelling salesman problem library. *ORSA J. Comput.* **1991**, *3*, 376–384. [CrossRef]
48. Montemanni, R.; Smith, D.H.; Rizzoli, A.E.; Gambardella, L.M. Sequential ordering problems for crane scheduling in port terminals. *Int. J. Simul. Process Model.* **2009**, *5*, 348–361. [CrossRef]
49. Shobaki, G.; Jamal, J. An exact algorithm for the sequential ordeing problem and its application to switching energy minimization in compilers. *Comput. Optim. Appl.* **2015**, *61*, 343–372. [CrossRef]
50. Wolpert, D.; Macready, W. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67. [CrossRef]