algorithms

MDPI

Article

Compact Models for Some Cluster Problems on Node-Colored Graphs

Roberto Montemanni [1,*], Derek H. Smith [2], Pongchanun Luangpaiboon [3] and Pasura Aungkulanon [4]

[1] Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 42122 Reggio Emilia, Italy
[2] Faculty of Computing, Engineering and Science, University of South Wales, Pontypridd CF37 1DL, UK; derek.smith@southwales.ac.uk
[3] Faculty of Engineering, Thammasat School of Engineering, Thammasat University, Pathum Thani 12120, Thailand; lpongch@engr.tu.ac.th
[4] Department of Materials Handling and Logistics Engineering, Faculty of Engineering, King Mongkut's University of Technology North Bangkok, Bangkok 10800, Thailand; pasura.a@eng.kmutnb.ac.th
* Correspondence: roberto.montemanni@unimore.it; Tel.: +39-0522-522-126

Abstract

Three optimization problems based on node-colored undirected graphs are the subject of the present study. These problems model real-world applications in several domains, such as cybersecurity, bioinformatics, and social networks, although they have a similar abstract representation. In all of the problems, the goal is to partition the graph into colorful connected components, which means that in each of the connected components, a color can appear in at most one node. The problems are optimized according to different objective functions, leading to different optimal partitions. We propose a compact Mixed Integer Linear Programming formulation for each of the three problems. These models are based on spanning trees, represented through multi-commodity flows. The compact nature of the new linear models is easier to handle than the approaches that previously appeared in the literature. These were based on models with an exponential number of constraints, which, therefore, required complex solving techniques based on the dynamic generation of constraints within a branch-and-cut framework. Computational experiments carried out on the standard benchmark instances for the problems show the potential of the new compact methods, which, once fed into modern state-of-the-art solvers, are able to obtain results better than the previous algorithmic approaches. As an outcome of the experimental campaign, a dozen instances of the different problems considered are closed for the first time.

Keywords: colored graphs; minimum orthogonal partition; maximum edges in transitive closure; minimum colorful components; compact mixed integer linear programming models

Academic Editor: Ming-Feng Ge

Received: 10 November 2025
Revised: 26 November 2025
Accepted: 27 November 2025
Published: 29 November 2025

Citation: Montemanni, R.; Smith, D.H.; Luangpaiboon, P.; Aungkulanon, P. Compact Models for Some Cluster Problems on Node-Colored Graphs. *Algorithms* **2025**, *18*, 759. https://doi.org/10.3390/a18120759

1. Introduction

Connected components are a crucial tool in graph theory to model and solve real-world problems in several application domains. A connected component is a subgraph such that a path exists between each pair of nodes in the component. In this work, they are considered in the context of colored graphs, where each node is assigned a color, and typically, *colorful* connected components are relevant. Given a node-colored graph $G = (V, E)$, a connected component of G is *colorful* if no two nodes within the component share the same color. Starting from this common background, this paper focuses on three optimization problems

involving colorful components: the *Minimum Orthogonal Partition* (MOP), the *Maximum Edges in Transitive Closure* (MEC) problem, and the *Minimum Colorful Components* (MCC) problem. These problems were originally introduced in the context of orthology gene identification in bioinformatics [1–3], with colors associated with genes from different genomes connected through pairwise homology relations. In this context, the aim is to remove redundant and spurious homologies, and this is technically achieved by looking for connected components that satisfy the orthogonality property. In all three problems, the goal is, therefore, to remove edges so that the resulting graph is partitioned into colorful, connected components. The objective of the three problems is, however, different and reflects alternative desiderata in the biological context: the MOP problem minimizes the number of deleted edges, the MEC problem maximizes the number of edges in the transitive closure of the resulting graph, and the MCC problem minimizes the total number of colorful components. Beyond bioinformatics, the MOP, MEC, and MCC problems have applications in other domains, such as social network analysis, cybersecurity, computer network design, and blockchains (see [4] for full details).

The MOP problem was first introduced by Ref. [5]. As noted by Ref. [2], it can be viewed as a special case of the $\mathcal{NP}$-hard *Minimum Multi-Cut* problem, which asks for the minimum number of edges to be removed in order to disconnect given pairs of nodes (which are pairs of nodes with the same color in the case of MOP). The MOP problem can also be formulated as an instance of the *Multi-Multiway Cut* problem [6], which is defined over sets of nodes (that are the nodes with the same color in MOP). The MOP problem has been shown to be solvable in polynomial time when the number of colors is at most two, whereas it is $\mathcal{NP}$-hard in general [2]. Fixed-parameter algorithms have also been proposed in the same work for special cases, showing that the problem is fixed-parameter tractable with respect to the number of colors and the number of edge deletions. Again, concerning parametrization, Ref. [7] studied the problem parameterized by the size of a node cover. Ref. [5] proposed an approximation algorithm for edge-weighted graphs. Several heuristic approaches were explored in Refs. [1,2]. In Bruckner et al. [8], two more heuristic approaches were proposed together with some reformulations in terms of implicit hitting set problems and a clique partitioning problem. The implicit hitting set framework [9,10] allows constraints to be generated dynamically and added iteratively as needed. The clique-partition-based formulation [11] instead transforms the problem into finding a partition of the graph into cliques, ensuring that the colorful property is maintained. In spite of having a polynomial number of constraints—the implicit hitting set formulation has exponentially many constraints—the number of constraints is considered too large by the authors, and a row generation scheme is employed.

The MEC problem—based on transitive closures—is motivated by the observation that relations between genes exhibit transitivity in several bioinformatics applications studying orthologies. The authors of [1] conjectured the MEC problem to be $\mathcal{NP}$-hard, and presented a rudimentary heuristic approach. Some other complexity and approximation results can be found in [3,12], showing that MEC is $\mathcal{APX}$-hard when there are at least three colors and $\mathcal{NP}$-hard to approximate within a given factor. More results related to parametrized approximation complexity can be found in Ref. [13].

The MCC problem was first introduced in [3], where the authors provided several general and special-case complexity and approximation results. Further approximation and complexity results can be found in [13], where the authors also show that the MCC problem is equivalent to the *Minimum Multi-Cut* problem [14].

The reference and inspiration for the present manuscript is the work by Archetti et al. [4], where the authors analyze MOP, MEC, and MCC within a common framework. They present Linear and Non-Linear Integer Programming formulations characterized by an exponential

number of constraints, several theoretical results on the structure of the problems, and an exact branch-and-cut algorithm incorporating several valid inequalities, variable bounding, warm-start strategies, and preprocessing routines. To the best of their knowledge, no previous exact global algorithms existed for solving these problems. The resulting framework represents the current state-of-the-art for solving the three problems considered.

In the present manuscript, we introduce compact Mixed Integer Linear Programming models for the problems considered, based on a common representation of the colorful-components subproblem based on a multi-commodity spanning tree formulation. The advantage of compact models is that they do not require complex algorithmic approaches to be effective in practice, but they can instead be fed directly into black-box solvers, making the whole solving process extremely simple and more accessible to a wider community. Traditionally, purpose-designed algorithms that dynamically generate constraints from an exponential set perform better than compact models. However, recent developments in black-box solvers such as Google OR-Tools CP-SAT 9.14 [15], which use a portfolio of solvers running in parallel and exchanging information, are reviving interest in the latter, simpler models. Examples of applications where compact models attacked with modern solvers are able to provide state-of-the-art solutions can be found in [16–18].

The remainder of this paper is structured as follows. Section 2 formally defines the three problems that are the object of the present study. Section 3 presents compact models based on a common multi-commodity-flow spanning-tree formulation [19] for the three problems. Section 4 discusses the outcome of an extended computational campaign where the new models are compared with the methods presented in [4] on the instances available in the literature. Section 5 concludes the study and proposes some extensions for future work.

2. Definition of the Problems

In the context of the problems addressed in this manuscript, a node-colored graph $G = (V, E, C)$ is provided as input, with C being a set of colors associated with the nodes in V, such that $col(i)$ is the color of node $i \in V$. Given a connected component $S \subseteq G$ with node set V_S, S is defined as *colorful* if $c_i \neq c_j \; \forall i, j \in V_S$, which means all the nodes of S have different colors. An example of a colored-graph is provided in Figure 1.

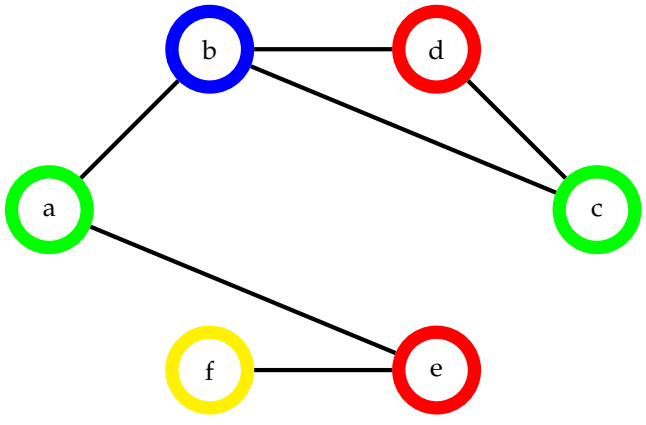

Figure 1. Example of a colored graph. The different colors assigned to nodes are instrumental in defining the different problems treated in this study.

For the three problems treated in this work, the set of feasible solutions is represented by the possible partitions of G into colorful components. However, they differ in terms of the objective. For ease of reading, let us remind the reader of the definition of the transitive closure of a graph. The transitive closure of an undirected graph G is a graph $H = (V, E^H)$,

where $E^H = \{\{i,j\} : \text{ there is a path connecting } i \text{ and } j \text{ in } G\}$. Notice that in the graph H, each connected component forms a clique.

We are now ready to provide a formal definition for each one of the three problems.

Definition 1 (Minimum Orthogonal Partition Problem—MOP)**.** *Given a node-colored graph $G = (V, E, C)$, the MOP problem consists of finding the smallest subset of edges $E' \subseteq E$ to remove from the graph G such that in the resulting graph $G' = (V, E \setminus E')$, all the connected components are colorful.*

An example of an optimal solution for the MOP problem defined on the graph of Figure 1 can be found in Figure 2.

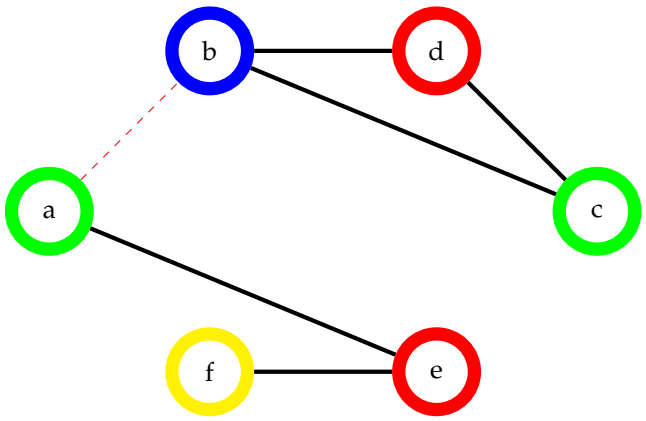

Figure 2. A solution of MOP for the colored graph introduced in Figure 1. The red dashed line identified the only edge left out while creating the connected components, which implied a solution cost of 1.

Definition 2 (Maximum Edges in Transitive Closure Problem—MEC)**.** *Given a node-colored graph $G = (V, E, C)$, the MEC problem consists of finding the subset of edges $E' \subseteq E$ to remove from the graph G, such that in the resulting graph $G' = (V, E \setminus E')$, all the connected components are colorful, and the number of edges in their transitive closure is maximized.*

An example of an optimal solution for the MEC problem defined on the graph of Figure 1 can be found in Figure 3.

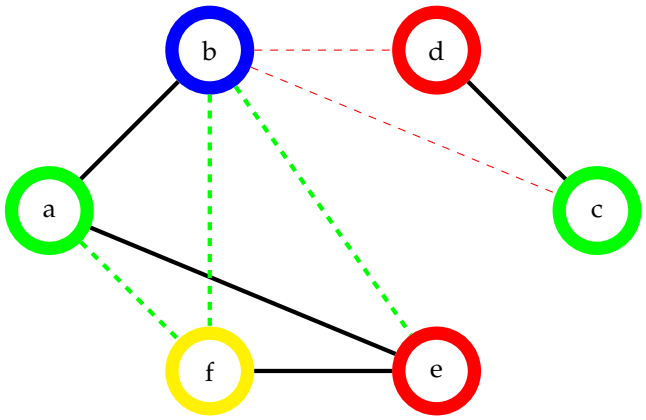

Figure 3. A solution of MEC for the colored graph introduced in Figure 1. The red dashed line identifies the original edges left out while creating the connected components. The dashed green edges are those of the transitive closure of the connected components, which have to be maximized together with the original edges (in black) left in the connected components.

Definition 3 (Minimum Colorful Components Problem—MCC)**.** *Given a node-colored graph $G = (V, E, C)$, the MCC problem consists of finding the subset of edges $E' \subseteq E$ to remove from the*

graph G, such that the resulting graph $G' = (V, E \setminus E')$ *consists of the smallest possible number of colorful components.*

An example of an optimal solution for the MEC problem defined on the graph of Figure 1 can be found in Figure 4.

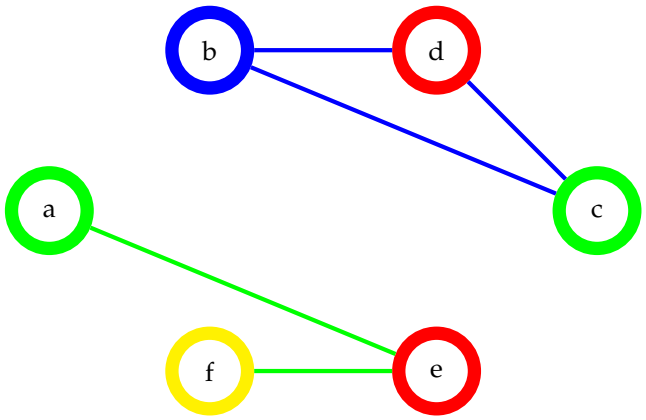

Figure 4. A solution of MCC for the colored graph introduced in Figure 1. The colors of the edges distinguish the different connected components. The target is to generate the smallest possible number of connected components.

The three problems are characterized by the same structure, which is based on identifying a set of colorful, connected components. This property will be used in Section 3, where some compact models for the problems will be devised.

3. Compact Mixed Integer Linear Programming Models

We propose compact MILPs for the three problems introduced in Section 2. These models are characterized by a polynomial number of variables and constraints. The previous approaches available in the literature [4] were based on models with an exponential number of constraints and on the dynamic generation of these constraints within a branch-and-cut framework, leading to a complex overall solution approach characterized by a substantial implementation effort. We instead propose self-contained models that can be fed into black-box solvers, which take care of the whole solution process.

The models we propose all rely on a spanning tree used to identify colorful, connected components and are calculated on a slightly modified input graph. We first introduce the set of constraints for such a spanning tree in Section 3.1. Then the models for the three problems are defined in detail in Sections 3.2–3.4.

3.1. Backbone Spanning Tree: A Compact Formulation to Model Colored Clustering

In this work, we propose compact models based on spanning tree formulations for MOP, MEC, and MCC. In the model, each connected component is identified by a representative node that is part of it. The idea of exploiting spanning tree concepts in the context of clustering is well established [20], and we want to show that it can be competitive also in the context of Mixed Integer Linear Programming models for the problems under investigation in this manuscript.

In order to formally define our model, we need to create a directed graph G^D, that will be instrumental in modeling connected components problems in terms of an arborescence problem. The node set of G^D is that of G with the addition of a new node; we will refer to it as r—the root of the original graph—which is connected by a directed arc (r, i) to each node $i \in V$. The node r will act as a root of the arborescence used to model the connected components. In fact, each connected component will be connected to r. Moreover, for

each edge $\{i, j\}$ of the graph G, we create two directed arcs (i, j) and (j, i). Formally, $G^D = (V^D, A)$, where $V^D = V \cup \{r\}$ and $A = \{(i, j) : \{i, j\} \in E\} \cup \{(r, i) : i \in V\}$. Notice that the undirected edge $\{i, j\} \in E$ is represented as a set containing nodes i and j and in the definition will generate two directed arcs (i, j) and (j, i), represented as ordered pairs in our notation. In the model we propose, connected components are identified by *representative nodes*, which are those connected to the root r through an arc (r, i) in an arborescence shaping the solution. An example of a support arborescence is provided in Figure 5.

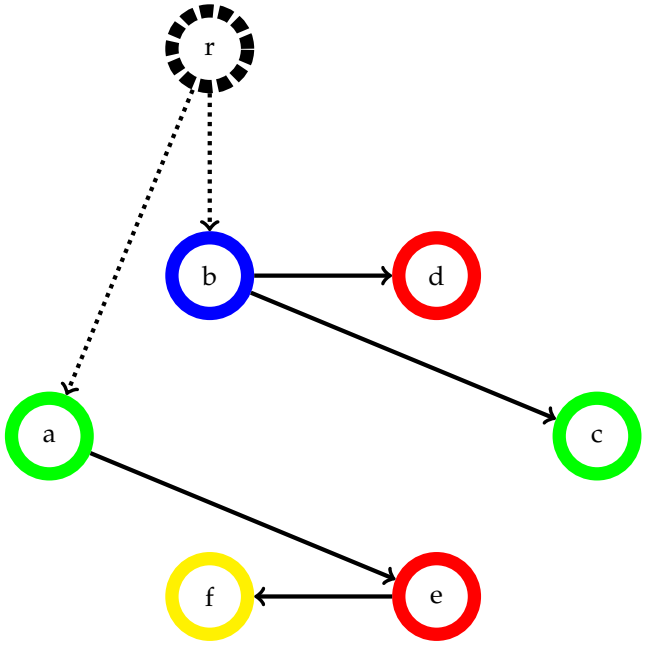

Figure 5. Backbone arborescence for the MCC solution depicted in Figure 2 relative to the example of Figure 1. The dotted node r is the artificial root of the tree, and the dotted arcs are the artificial arcs used by the solution among those added to connect r to the other nodes. Notice that the nodes a and b are in this solution the representatives of the two connected components.

Given the new graph G^D, we can model an arborescence rooted in r via a multi-commodity flow formulation, as discussed, for example, in [19]. The branches of the spanning tree on the original graph G corresponding to an arborescence on G^D will identify the connected components at the basis of the solutions of the three problems considered in this study. To model the arborescence problem, a binary variable x_{ij} is introduced for each $i, j \in V$; it takes the value 1 if the arc node i is part of the connected component with representative j, and 0 otherwise. A binary variable w_{ij} is introduced for each $\{i, j\} \in E$; it takes the value 1 if the edge $\{i, j\}$ is part of the tree used to represent the connected components, and 0 otherwise. On top of w variables representing the spanning tree, a set of integer variables y is defined to model a multi-commodity-flow arborescence representing the backbone of the spanning tree, such that $y_{ij} = \alpha > 0$ if arc (i, j) is part of the arborescence and carries α units of commodity. Given these variables, the resulting model is as follows.

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V \tag{1}$$

$$\sum_{i \in V} y_{ri} = |V| \tag{2}$$

$$y_{ri} + \sum_{(j,i) \in A} y_{ji} - \sum_{(i,j) \in A} y_{ij} = 1 \qquad i \in V \tag{3}$$

$$w_{ij} \leq y_{ij} \qquad (i,j) \in A \tag{4}$$

$$|V| w_{ij} \geq y_{ij} \qquad (i,j) \in A \tag{5}$$

$$w_{ij} = 1 \Rightarrow x_{ik} = x_{jk} \qquad (i,j) \in A, k \in V \tag{6}$$

$$|V| x_{ii} \geq y_{ri} \qquad i \in V \tag{7}$$

$$\sum_{i \in V : col(i) = c} x_{ij} \leq 1 \qquad c \in C, j \in V \tag{8}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V \tag{9}$$

$$y_{ij} \in \mathcal{N}_0^+ \qquad (i,j) \in A \tag{10}$$

$$w_{ij} \in \{0,1\} \qquad \{i,j\} \in V \tag{11}$$

Constraints (1) state that each node has to be assigned exactly to one cluster. Constraints (2) and (3) define an arborescence rooted in node r over the graph G'. They are based on a classic multi-commodity-flow model (see [19]). Constraints (2) specify that $|V|$ units of commodity—one for each vertex of V—must be sent out of the root r, while equalities (3) are conservation constraints and also impose that one unit of commodity is consumed by each vertex. The inequalities (4) and (5) connect w variables to y variables, making sure that an edge of the tree represented by w variables can be active only when one of the two corresponding arcs of the arborescence described by y variables carries some commodity. The indicator constraints (6) state that if $w_{ij} = 1$, then nodes i and j have to be assigned to the same cluster identified by a representative k. The constraints (7) regulate the selection of the representative of the clusters, which correspond to the nodes connected to the root r in the arborescence described by the y variables. Constraints (8) address colorfulness and stipulate that each color can appear at most once in each active cluster (identified through its representative). Finally, constraints (9)–(11) describe the domains of the variables.

The model above, describing a spanning tree via an arborescence, and, in turn, identifying connected components that fulfill constraints on the colors of the nodes they contain, will be the backbone of the models for the MOP, MEC, and MCC problems, as described in the next sections. The most relevant characteristic of the model is that it is based on a number of variables and constraints that are both polynomial in the number of nodes of the original graph G. The previous models discussed in the literature for the MOP, MEC, and MCC problems were instead characterized by an exponential number of constraints [4], and required a complex branch-and-cut framework to practically solve them. Conversely, the models we propose can be attacked directly by black-box solvers.

3.2. A Compact Model for the MOP

Making use of the model for the backbone spanning tree described in Section 3.1, the MOP can be modeled as follows. A new binary variable z_{ij} is introduced for each $\{i,j\} \in E$ and takes the value 1 if i and j are not part of the same connected component; 0 otherwise. The resulting model is as follows.

$$(CMOP)\ \min \sum_{\{i,j\}\in E} z_{ij} \tag{12}$$

$$\text{s.t. } (1)-(11)$$

$$z_{ij} \geq x_{ik} - x_{jk} \qquad \{i,j\} \in E, k \in V \tag{13}$$

$$z_{ij} \geq x_{jk} - x_{ik} \qquad \{i,j\} \in E, k \in V \tag{14}$$

$$z_{ij} \in \{0,1\} \qquad \{i,j\} \in E \tag{15}$$

The objective function minimizes the number of edges of the original graph G that are across different connected components in the solution. Inequalities (13) and (14) force each z_{ij} variable to take the value 1 if the nodes of an edge $i,j \in E$ are not in the same connected component. Constraints (15) finally define the domain of the z variables.

3.3. A Compact Model for the MEC

A new binary variable u_{ij} is introduced for each $i,j \in V$ and takes the value 1 if i and j are not part of the same connected component; 0 otherwise. Notice that with respect to the z variables of the model for the MOP problem, which were defined only on the edge set E, here the u variables are defined for each pair of nodes in V. The resulting model is as follows.

$$(CMEC)\ \min \sum_{i,j\in V} u_{ij} \tag{16}$$

$$\text{s.t. } (1)-(11)$$

$$u_{ij} \geq x_{ik} - x_{jk} \qquad i,j,k \in V \tag{17}$$

$$u_{ij} \geq x_{jk} - x_{ik} \qquad i,j,k \in V \tag{18}$$

$$u_{ij} \in \{0,1\} \qquad i,j \in V \tag{19}$$

The objective function (16) minimizes the number of pairs or nodes that are in different connected components in the solution. This optimization is trivially equivalent to the MEC objective function described in Section 2, which aims at maximizing the edges within the transitive closures of the connected components. In fact, in (16) we minimize the complement of the edges maximized in Definition 2, and the objective function value $v(MEC)$ of a solution calculated according to Definition 2 can always be obtained as follows:

$$v(MEC) = \frac{|V|(|V|-1)}{2} - \sum_{i,j\in V} u_{ij} \tag{20}$$

Inequalities (17) and (18) force each z_{ij} variable to take the value 1 if nodes i and j are not in the same connected component. Constraints (19) finally define the domain of the u variables.

3.4. A Compact Model for the MCC

Making use of the model for the backbone spanning tree described in Section 3.1, the MCC can be modeled as follows.

$$(CMCC)\ \min \sum_{i\in V} y_{ri} \tag{21}$$

$$\text{s.t. } (1)-(11)$$

Given the objective of the problem, which is to minimize the number of colorful connected components, it is enough to add an objective function to the constraints used to define the

backbone structure. In detail, in (21), we minimize the number of outgoing arcs from the root r in the solution.

4. Experimental Results

This section is devoted to the presentation and discussion of the experimental campaign we carried out on the MOP, MEC, and MCC problems by the models discussed in Section 3. The benchmark instances adopted—which are those commonly used in the literature—are introduced in Section 4.1, while the experimental comparison between the new models and the methods discussed in the recent work [4] can be found in Section 4.2.

4.1. Benchmark Instances

To assess the effectiveness of the compact models discussed in Section 3, the instances already adopted in [4] for the problems treated in this work were used. The interested reader can download such instances from [4], where all the solving codes of the methods proposed by the authors are also available. The instances track back to [2], where the authors generated the multiple alignment dataset of the *BAliBASE 3.0* benchmark set [21]. For the instances consisting of multiple connected components, the authors of [4] solved the problem separately for each of them and restricted the analysis to all graphs having between 10 and 210 nodes. The resulting dataset consists of 409 instances, with a broad range of difficulty levels.

4.2. Computational Experiments

The models discussed in Section 3 have been solved with both Gurobi version 12.0.3 [22] and Google OR-Tools CP-SAT 9.14 [15]. The experiments have been carried out on a computer equipped with an Intel Core i7-12700F processor running at 2.1 GHz and 32 GB of RAM, using 20 threads. The results will be compared with those reported in [4], obtained on a computer equipped with an Intel Core i7-3770 processor running at 3.4 GHz and 16 GB of RAM using one thread only. For all the methods, a maximum of 10 GB of RAM is allowed for the calculations. The computation times of our approaches are multiplied factor of 14.67, which, according to [23], makes the comparison fair by taking into account both the differences in the CPUs and the number of threads allowed for the computation. In the rest of the paper, we will, therefore, refer to times normalized to the Intel Core i7-3770 processor.

For attacking the models described in Section 3, we decided not to use any of the preprocessing techniques or valid inequalities discussed in [4], since preliminary experiments suggested that both Gurobi and CP-SAT were not receiving any observable benefits from them.

Notice that the choice of using the maximum number of available threads for solving the models described in Section 3 is suggested by the use of the CP-SAT solver, which takes great advantage of parallel computation, but would not be competitive if run longer on one thread only. The Gurobi solver, on the other hand, is not expected to benefit greatly from the parallel environment.

In the tables reported in the next sections, several variations of the methods reported in [4], with runs of a maximum 3600 s on a single core, are listed. We use the names as they appear in the paper and in its online compendium, and we refer the interested reader there for full details about the differences. When relevant, we also report statistics about the best bounds obtained by all the methods reported in [4] (Overall Best Bounds). Concerning the methods discussed in Section 3, the results achieved by solving them using Gurobi and CP-SAT with a maximum computation time of 3600 wall seconds are reported, in order to have a fair comparison with the results reported in [4]. A longer run of CP-SAT with

a maximum of 36,000 s, aiming at understanding the full potential of the models, is also reported when relevant.

For each method considered to solve the different problems, we report the aggregated results for the 409 instances considered in terms of:

- LB: The average lower bounds obtained in the given computation time;
- UB: The average upper bounds obtained in the given computation time;
- Gap %: The average optimality gap obtained in the given computation time;
- # Solved: The number of instances solved to optimality in the given computation time;
- Sec: The average computation time in seconds. For the new methods introduced in this work, we also report in brackets the unnormalized wall time registered for each instance on the Intel Core i7-12700F adopted for the experiments.

The detailed disaggregated results can be found in the online repository available at https://github.com/Monte14/MOP_MEC_MCC/ (accessed on 25 November 2025).

4.2.1. Minimum Orthogonal Partition Problem

The results for the MOP problem are summarized in Table 1. All the instances could be solved to optimality by some of the methods proposed in [4]; therefore, the challenge is mainly in the computation time required to close all the instances. When attacking the model CMOP presented in Section 3.2 with Gurobi, we see that already 396 out of the 409 instances can be closed within the given time, showing that reasoning in terms of compact models makes sense for this problem. CP-SAT performs remarkably better than Gurobi on the model CMOP, being able to solve all 409 instances to optimality within the given time. CP-SAT is likely to take advantage of the interaction of linear programming solvers, SAT solvers, and metaheuristic algorithms that run in parallel during the solving effort, while Gurobi relies only on enhanced linear programming solvers (and, eventually, metaheuristics). For this problem, a variety of collaborative solvers seem to be more effective than a self-standing sophisticated linear programming solver.

Table 1. Experimental results for the Minimum Orthogonal Partition (MOP) problem. Statistics are over 409 instances.

Method	LB	UB	Gap %	# Solved	Sec
"MOP (D)" [4]	17.53	20.01	1.87%	385	280.63
"MOP (A)" [4]	17.91	20.12	1.53%	384	255.14
"MOP" [4]	18.72	19.32	0.24%	408	2.94
"MOP + (15)" [4]	18.72	19.32	0.24%	408	3.22
"MOP + (14a)" [4]	17.92	19.74	1.08%	394	158.13
"MOP + (14b)" [4]	18.72	19.33	0.25%	406	29.61
"MOP + Alg.3" [4]	19.16	19.16	0.00%	409	4.81
"MOP + Alg.5" [4]	18.72	19.32	0.24%	408	3.41
"MOP + Alg.3 + Alg.5" [4]	19.16	19.16	0.00%	409	5.46
CMOP-Gurobi	19.15	19.59	0.35%	396	132.48 (9.02)
CMOP-CP-SAT	19.16	19.16	0.00%	409	3.33 (0.23)

In the chart reported in Figure 6, we plot for each of the methods considered in the experiments reported in Table 1 a line representing its performance in terms of instances solved over time. In detail, the number of instances closed to optimality is reported on the x-axis, while the computation time in seconds is reported on the y-axis. Therefore, the best performing methods are those with a line sticking as much as possible to the x-axis and reaching out as far as possible on the right. From the figure, we can see how several methods have similar favorable lines reaching 409, while others are clearly dominated, with Gurobi operating on model CMOP showing average performance. In terms of normalized

computation times, Gurobi shows average performance, while CP-SAT appears to be the fastest method. When considering the wall times reported in brackets, we can see that CP-SAT is able to solve each instance in negligible time.

In order to clarify the situation among the best methods, in Figure 7, we zoom in on the relevant area of the plot of Figure 6. We can observe how the blue line of CP-SAT is close to the fastest methods when solving up to 408 instances; however, it is the fastest one to close the whole set of benchmarks.

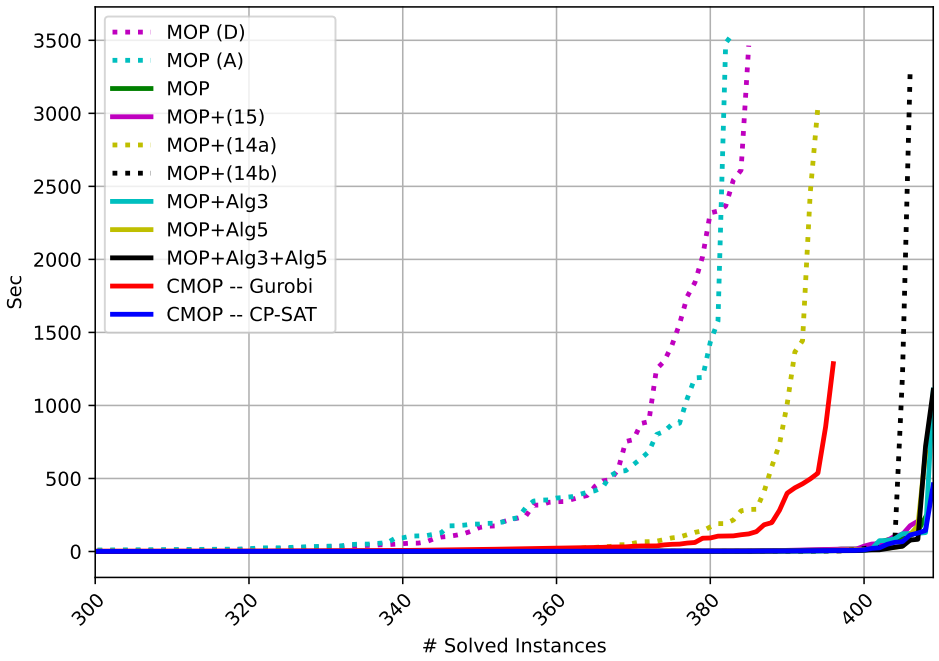

Figure 6. Minimum Orthogonal Partition problem. Times vs. number of instances solved to optimality.

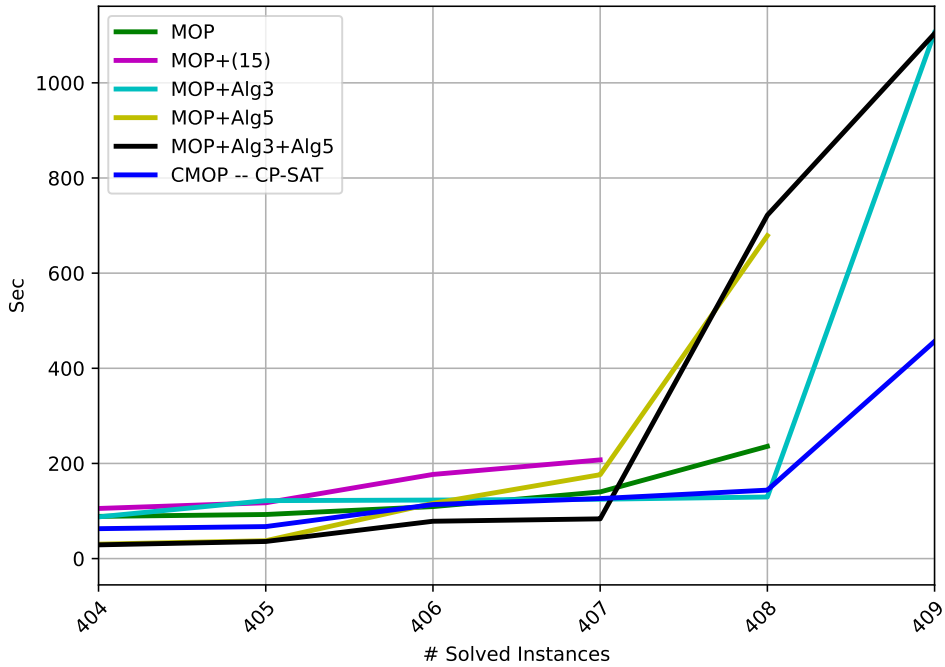

Figure 7. Minimum Orthogonal Partition problem. Times vs. number of instances solved to optimality, zoomed on the right end of Figure 6.

4.2.2. Maximum Edges in Transitive Closure Problem

The results for the MEC problem are summarized in Table 2. Solving the model CMEC presented in Section 3.3 with Gurobi, the number of solved instances and computation times position the method in the range of the most basic methods presented in [4], while CP-SAT outperforms the previous methods both in terms of solving time and number of instances solved. CP-SAT remains superior even when compared to the overall best bounds previously appeared in the literature. A further 3 instances can be closed if we let the solver run for 36,000 normalized seconds, reaching a total of 404 instances closed. These results are achieved with very short wall times.

Table 2. Experimental results for the Maximum Edges in Transitive Closure (MEC) problem. Statistics over 409 instances.

Method	LB	UB	Gap %	# Solved	Sec
"MEC (D)" [4]	23.66	236.74	21.74%	257	1504.49
"MEC" [4]	28.89	258.72	12.48%	321	840.90
"MEC + (15) + (16)" [4]	28.64	254.71	13.64%	317	891.95
"MEC + Alg.1" [4]	32.39	190.19	2.63%	372	429.24
"MEC + (14a)" [4]	25.81	211.21	5.71%	356	498.73
"MEC + (14b)" [4]	28.72	189.32	2.38%	397	148.19
"MEC + (14b) + Alg.1" [4]	33.17	191.36	2.05%	394	164.17
Overall Best Bounds [4]	33.24	172.29	1.90%	398	-
CMEC-Gurobi	27.99	223.01	13.21%	328	796.14 (54.18)
CMEC-CP-SAT	33.04	110.99	1.12%	401	90.78 (6.18)
CMEC-CP-SAT 36,000 s	33.63	33.79	0.05%	404	501.61 (34.14)

A chart analogous to that reported in Figure 6 is presented in Figure 8 for the results on the MEC problem. From the chart, the dominance of the method involving the run of CP-SAT on the model CMEC is clearly dominant, both in terms of computation times and the number of instances closed.

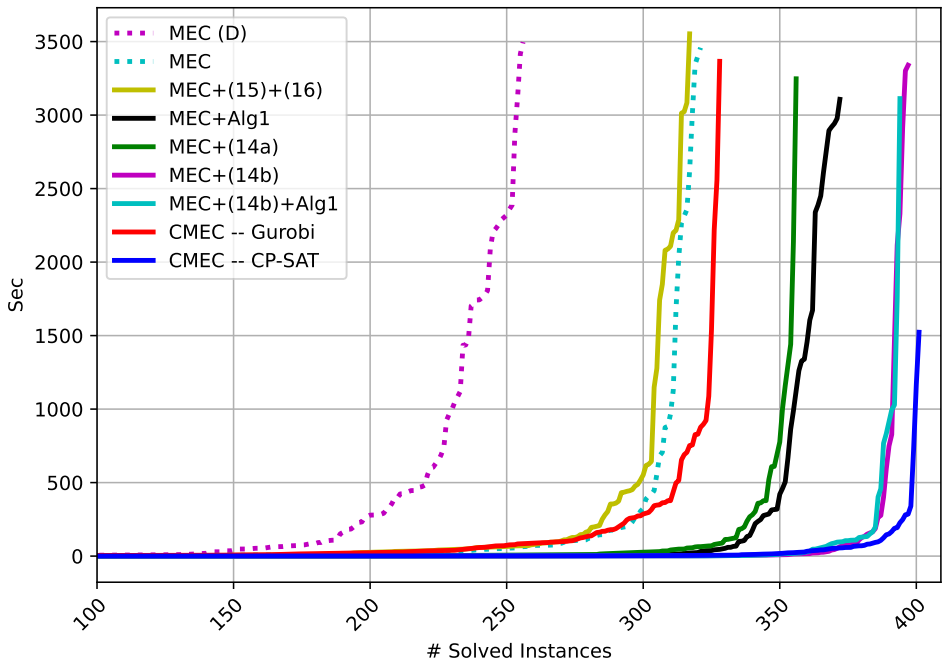

Figure 8. Maximum Edges in Transitive Closure problem. Times vs. number of instances solved to optimality.

An analysis of the disaggregated results reported in the online compendium finally suggests that when the CP-SAT solver was run on model CMEC for 3600 normalized seconds, it was able to improve the best-known results for 8 lower bounds and 10 upper bounds, closing for the first time three instances. When the method was run for 36,000 s, these figures increased, respectively, to 10, 11, and 6.

4.2.3. Minimum Colorful Components Problem

The results for the MCC problem are summarized in Table 3. The model CMCC presented in Section 3.4, while solved with Gurobi, guarantees poor performances, emerging as the worst method of them all. When the model CMCC is attacked with the CP-SAT solver, the maximum number of instances closed among the different methods is achieved (400), notwithstanding one of the methods discussed in Ref. [4]—namely MCC + (14b) + Alg.2—was able to have a lower average percentage gap. This indicates that CP-SAT might have some robustness issues, with particularly poor performances on some of the instances. It is also interesting to observe that the overall best bounds reported in [4] allowed the closure of 400 instances, suggesting that the different methods perform differently on different instances. Finally, wall times remain very favorable for CP-SAT.

Table 3. Experimental results for the Minimum Colorful Components (MCC) problem. Statistics over 409 instances.

Method	LB	UB	Gap %	# Solved	Sec
"MCC (D)" [4]	4.70	8.14	5.08%	362	454.94
"MCC" [4]	4.71	6.98	2.12%	387	229.93
"MCC + (14a)" [4]	4.72	6.82	2.38%	385	237.91
"MCC + (14b)" [4]	4.90	6.13	1.18%	399	101.80
"MCC + Alg.2" [4]	4.87	5.55	0.92%	390	174.07
"MCC + (15) + (17)" [4]	4.72	7.09	2.42%	384	227.95
"MCC + (14b) + Alg.2" [4]	5.02	5.19	0.43%	397	113.57
Overall Best Bounds [4]	5.03	5.15	0.28%	401	-
CMCC-Gurobi	4.04	5.09	6.70%	356	552.63 (37.61)
CMCC-CP-SAT	4.90	5.08	0.63%	400	98.78 (6.72)
CMCC-CP-SAT 36,000 s	4.99	5.08	0.24%	404	538.96 (36.68)

A chart similar to those seen in the previous sections is presented in Figure 9 for the results on the MCC problem. The chart suggests that CP-SAT performs similarly to the best methods (in terms of time to close instances) up to circa 390 instances, when its advantage over the competitors is clear. Although the average optimality gap is not considered in the figure, and—as observed before—the method defers somewhat to MCC + (14b) + Alg.2 under this viewpoint.

The disaggregated results reported in the online compendium indicate that when the CP-SAT solver was run on model CMEC for 3600 normalized seconds, it was able to improve the best-known results for four lower bounds and four upper bounds, closing for the first time six instances. When the method was run for 36,000 s, these figures do not change, although the optimality gap is reduced. Remarkably, by combining the results of the methods presented in [4] with the new ones in this manuscript, a total of 407 instances have been closed to optimality, with a certain choice of methods able to prove optimality.

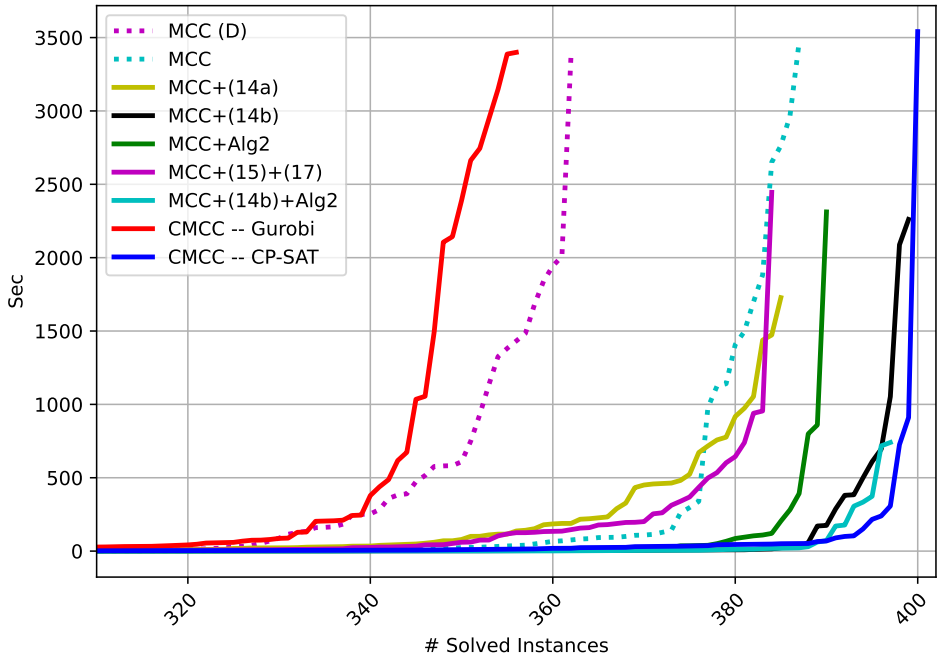

Figure 9. Minimum Colorful Components problem. Times vs. number of instances solved to optimality.

4.3. Discussion

The results presented in the previous sections mainly suggest that the new compact models in Section 3 can achieve state-of-the-art results (and sometimes improve the best-known results) when solved with modern solvers such as the CP-SAT solver. The advantage is likely due to the exploitation of the intrinsic parallel environment guaranteed by modern processors, making compact models very attractive. More complex algorithmic methods, typically based on metaheuristics or branch-and-bound, are less prone to benefit from these new hardware architectures, due to the centralized nature of their solving process.

5. Conclusions

In this work, some new compact formulations for three optimization problems defined on node-colored undirected graphs have been introduced, namely the Minimum Orthogonal Partition, the Maximum Edges in Transitive Closure, and the Minimum Colorful Components problems. All the models rely on a common structure, based on a spanning tree, to represent connected components. Despite their common structure, each problem targets a different objective function while sharing the same goal, namely, partitioning the graph into colorful connected components.

The proposed compact formulations, based on spanning-tree representations through multi-commodity-flow models, offer a different approach compared with previous solving methods, which mainly rely on exponentially large models and complex branch-and-cut strategies.

Extensive computational experiments on standard benchmark instances confirm the effectiveness of the new compact formulations, in particular when solved with modern state-of-the-art black-box solvers. A dozen benchmark instances across the three problems have been solved to optimality for the first time, further highlighting the practical potential of the proposed approach.

Future work should be in the direction of investigating compact models for other optimization problems on colored graphs.

Author Contributions: Conceptualization, R.M., D.H.S., P.L. and P.A.; methodology, R.M.; software, R.M.; validation, R.M., D.H.S., P.L. and P.A.; formal analysis, R.M., D.H.S., P.L. and P.A.; investigation, R.M., D.H.S., P.L. and P.A.; resources, R.M.; data curation, R.M.; writing—original draft preparation, R.M. and D.H.S.; writing—review and editing, R.M., D.H.S., P.L. and P.A.; visualization, R.M., D.H.S., P.L. and P.A. 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 for the experiments are available from the online repository [4].

Acknowledgments: The authors have reviewed and edited the output and take full responsibility for the content of this publication.

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

References

1. Zheng, C.; Swenson, K.; Lyons, E.; Sankoff, D. OMG! Orthologs in Multiple Genomes–Competing Graph-Theoretical Formulations. In *Algorithms in Bioinformatics, Proceedings of the WABI 2011, Saarbrücken, Germany, 5–7 September 2011* ; Przytycka, T.M., Sagot, M.F., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 364–375.
2. Bruckner, S.; Hüffner, F.; Komusiewicz, C.; Niedermeier, R.; Thiel, S.; Uhlmann, J. Partitioning into colorful components by minimum edge deletions. In *Combinatorial Pattern Matching, Proceedings of the 23rd Annual Symposium, CPM 2012, Helsinki, Finland, 3–5 July 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 56–69.
3. Adamaszek, A.; Popa, A. Algorithmic and hardness results for the colorful components problems. In *LATIN 2014: Theoretical Informatics, Proceedings of the 11th Latin American Symposium, Montevideo, Uruguay, 31 March–4 April 2014*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 683–694.
4. Archetti, C.; Cerulli, M.; Sorgente, C. Branch-and-cut algorithms for colorful components problems. *INFORMS J. Comput.* **2025**. [CrossRef]
5. He, G.; Liu, J.; Zhao, C. Approximation algorithms for some graph partitioning problems. *J. Graph Algorithms Appl.* **2000**, *4*, 1–11. [CrossRef]
6. Avidor, A.; Langberg, M. The multi-multiway cut problem. *Theor. Comput. Sci.* **2007**, *377*, 35–42. [CrossRef]
7. Misra, N. On the parameterized complexity of colorful components and related problems. In *International Workshop on Combinatorial Algorithms, Proceedings of the IWOCA 2018, Singapore, 16–19 July 2018*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 237–249.
8. Bruckner, S.; Hüffner, F.; Komusiewicz, C.; Niedermeier, R.; Wernicke, S. Correcting interlanguage links in Wikipedia: A colorful components approach. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 1359–1368.
9. Chandrasekaran, K.; Karp, R.; Moreno-Centeno, E.; Vempala, S. Algorithms for Implicit Hitting Set Problems. In Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, USA, 23–25 January 2011; pp. 614–629.
10. Moreno-Centeno, E.; Karp, R. The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment. *Oper. Res.* **2013**, *61*, 453–468. [CrossRef]
11. Grötschel, M.; Wakabayashi, Y. A cutting plane algorithm for a clustering problem. *Math. Program.* **1989**, *45*, 59–96. [CrossRef]
12. Adamaszek, A.; Blin, G.; Popa, A. Approximation and hardness results for the maximum edges in transitive closure problem. In *Combinatorial Algorithms*; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 13–23.
13. Dondi, R.; Sikora, F. Complexity and approximation for colorful component problems. *J. Comput. Syst. Sci.* **2018**, *94*, 62–75.
14. Hu, T.C. Multi-commodity network flows. *Oper. Res.* **1963**, *11*, 344–360. [CrossRef]
15. Perron, L.; Didier, F. Google OR-Tools-CP-SAT. 2025. Available online: https://developers.google.com/optimization/cp/cp_solver/ (accessed on 10 October 2025).
16. Montemanni, R.; Dell'Amico, M. Solving the parallel drone scheduling traveling salesman problem via constraint programming. *Algorithms* **2023**, *16*, 40. [CrossRef]
17. Montemanni, R.; Dell'Amico, M.; Corsini, A. Parallel drone scheduling vehicle routing problems with collective drones. *Comput. Oper. Res.* **2024**, *163*, 106514. [CrossRef]
18. Montemanni, R.; Ceschia, S.; Schaerf, A. A compact model for the home healthcare routing and scheduling problem. *EURO J. Comput. Optim.* **2025**, *13*, 100101. [CrossRef]
19. Magnanti, T.L.; Wolsey, L. Optimal trees. In *Handbook in Operations Research and Management Science: Networks Models*; Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L., Eds.; Elsevier: Amsterdam, The Netherlands, 1995; pp. 503–615.

20. Gagolewski, M.; Cena, A.; Bartoszuk, M.; Brzozowski, Ł. Clustering with Minimum Spanning Trees: How Good Can It Be? *J. Classif.* **2025**, *42*, 90–112. [CrossRef]
21. Thompson, J.D.; Koehl, P.; Ripp, R.; Poch, O. Balibase 3.0: Latest developments of the multiple sequence alignment benchmark. *Proteins Struct. Funct. Bioinform.* **2005**, *61*, 127–136. [CrossRef] [PubMed]
22. Gurobi Optimization, LLC. *Gurobi 12.0.3*; Gurobi Optimization, LLC.: Beaverton, OR, USA, 2025.
23. PassMark Software Pty Ltd. *CPU Benchmarks*; PassMark Software Pty Ltd.: Sydney, Australia, 2025.