UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

EMANUEL ELIAS SILVA CASTELO

CONTRIBUTIONS TO THE k-COLOR SHORTEST PATH PROBLEM

FORTALEZA

2023

EMANUEL ELIAS SILVA CASTELO

CONTRIBUTIONS TO THE k-COLOR SHORTEST PATH PROBLEM

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Teoria da Computação.

Orientador: Prof. Dr. Rafael Castro de Andrade.
Coorientador: Prof. Dr. Rommel Dias Saraiva.

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C345c Castelo, Emanuel Elias Silva.
Contributions to the k-Color Shortest Path Problem / Emanuel Elias Silva Castelo. – 2023.
54 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2023.
Orientação: Prof. Dr. Rafael Castro de Andrade.
Coorientação: Prof. Dr. Rommel Dias Saraiva.

1. Otimização Combinatória. 2. Desigualdades Válidas. 3. Caminho Mínimo k-Rotulado. I. Título.
CDD 005

EMANUEL ELIAS SILVA CASTELO

CONTRIBUTIONS TO THE k-COLOR SHORTEST PATH PROBLEM

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Teoria da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Rafael Castro de Andrade (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Rommel Dias Saraiva (Coorientador)
Universidade de Fortaleza (UNIFOR)

Prof. Dr. Manoel Bezerra Campêlo Neto
Universidade Federal do Ceará (UFC)

Prof. Dr. Thiago Ferreira de Noronha
Universidade Federal de Minas Gerais (UFMG)

Prof. Dr. Iago Augusto de Carvalho
Universidade Federal de Alfenas (UNIFAL)

A todos que, direta ou indiretamente, apoiaram o desenvolvimento deste trabalho.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Prof. Rafael Andrade, for his constructive comments and patience during my journey in the program. From him, I feel I have learned a lot.

I must also thank my co-supervisor, Prof. Rommel Saraiva, for his patience and guidance during this research.

I also thank all the members of the ParGO research group. In particular, I would like to thank all laboratory colleagues for the moments of distractions and research related discussions.

I would also like to thank the jury for all the constructive criticism that helped improve this work.

Lastly, I would like to thank my family for their support, from whom I won't dare to try to make an exhaustive list of their importance.

This study was financed in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP).

RESUMO

Dado um digrafo $D = (V, A)$, onde todos os arcos $(i, j) \in A$ possuem um custo associado $d(i, j) \in \mathbb{R}_+$ e uma cor $c(i, j)$, um inteiro positivo k, uma fonte s, e um destino t, o Problema do Caminho Mínimo k-Rotulado é um problema **NP**-Difícil que consiste em encontrar um (s,t)-caminho de custo mínimo em D usando no máximo k cores distintas. Propomos desigualdades válidas que fortalecem a relaxação linear de uma formulação existente na literatura de Programação Linear Inteira. Propomos ainda uma nova formulação exponencial, que pode ser resolvida por meio de um algoritmo de branch-and-cut. Introduzimos instâncias mais desafiadoras para o problema e apresentamos experimentos numéricos para as benchmark e as novas instâncias. Finalmente, avaliamos diferentes combinações das desigualdades válidas. Resultados computacionais para as ideias propostas e para as abordagens existentes para o problema mostram a eficiência das novas desigualdades em lidar com as novas instâncias ambos em termos de tempo de execução e em proporcionar melhoria nas soluções relaxadas.

Palavras-chave: otimização combinatória; caminho mínimo k-rotulado; desigualdades válidas.

ABSTRACT

Given a digraph $D = (V,A)$, where all arcs $(i,j) \in A$ have an associated cost $d(i,j) \in \mathbb{R}_+$ and a color $c(i,j)$, a positive integer k, a source s, and a destination t, the k-Color Shortest Path Problem is an **NP**-Hard problem that consists in finding the shortest (s,t)-path in D while using at most k distinct colors. We propose valid inequalities for the problem that proved to strengthen the linear relaxation of an existing Integer Linear Programming formulation. An exponential set of valid inequalities defines a new formulation for the problem and is solved by using a branch-and-cut algorithm. We introduce more challenging instances of the problem and present numerical experiments for both benchmark and the new instances. Finally, we evaluate the individual and the collective use of the valid inequalities. Computational results for the proposed ideas and for existing solution approaches for the problem showed the effectiveness of the new inequalities in handling the new instances both in terms of execution times and improving their linear relaxed solutions.

Keywords: combinatorial optimization; k-color shortest path; valid inequalities.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	Set of arcs
A_h	Set of arcs of color h
A_h^P	Set of arcs of color h in the path P
$c(u,v)$	Color of the arc (u,v)
C	Set of colors
$d(u,v)$	Cost of the arc (u,v)
D	Directed graph
E	Set of edges
G	Undirected graph
k	Maximum number of colors
$N^-(v)$	Maximal subset of the in-neighborhood of vertex v where no pair of nodes belonging to it reach each other
$\mathscr{P}$	Set of all (s,t)-paths in D
$P_{s,u}$	Path from s to u
$\bar{R}$	Maximal subset of non-reachable vertices
s	Source node
t	Destination node
V	Set of vertices
$R(v)$	Set of vertices reached by node v
$\delta^+(v)$	Subset of vertices in the out-neighborhood of v
$\delta^-(v)$	Subset of vertices in the in-neighborhood of v
$\pi(u,v)$	Cost of the shortest (u,v)-path
(u,v)	Arc from u to v
$[S,T]$	The edge-cut with endpoints in S and T.

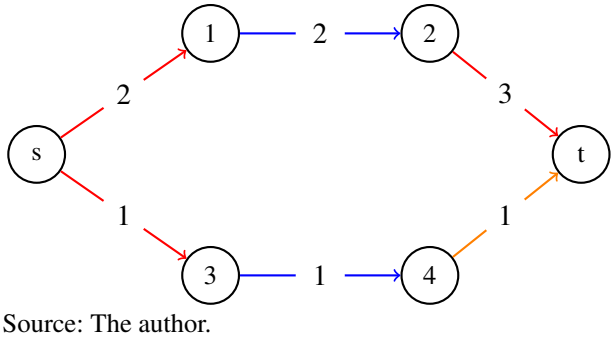
CONTENTS

1 INTRODUCTION

The shortest path problem (SPP) is one of the most popular problems in combinatorial optimization, defined in a weighted graph $G = (V, E, d)$, where V is the set of vertices, E is the set of edges, and $d : E \to \mathbb{R}$ is the edge cost function. It consists in finding the path of minimum cost that connects a source s and a destination t. Numerous efficient algorithms are known for the SPP. Assuming the graph has positive weight for every edge, the standard implementation of Dijkstra's algorithm return the shortest path, if one exists, in time $O(|V|^2)$ (DIJKSTRA, 1959). In contrast, if G has negative weights, the Bellman-Ford algorithm can be used, as long as there are no negative cycles, returning the solution in time $O(|V||E|)$ (BELLMAN, 1958). The Resource Constrained Shortest Path Problem (BEASLEY; CHRISTOFIDES, 1989), the Multiple k-Multicolor Paths (SANTOS *et al.*, 2017), and the k-Colored Shortest Path (FERONE *et al.*, 2019) are some extensions, among several others, of the standard SPP that are known to be **NP**-Hard.

In this work, we are interested in the k-Color Shortest Path Problem (k-CSPP). To describe it, let us define an arc-colored weighted digraph $D = (V, A, c, d)$, where every arc $(u, v) \in A$ has a positive cost $d(u, v)$ and a color $c(u, v)$. Given a source $s \in V$ and a destination $t \in V$ vertices, and a positive integer k, the k-CSPP consists in finding an (s, t)-path in D of minimum cost while using at most k distinct arc colors. The problem was studied independently by Ferone *et al.* (2019) and Dehouche (2020). In addition, Li *et al.* (2001) showed that finding an (s, t)-path in an edge-colored graph with at most k colors is **NP**-Complete, which implies that there does not exist a polynomial-time algorithm or approximation algorithm that allows finding a feasible solution for an arbitrary instance of the problem unless $\mathbf{P} = \mathbf{NP}$. An illustration of the problem is depicted in Fig. 1.

Figure 1 – Illustration of an instance of the k-CSPP. For $k \geq 3$ the (s, t)-path of minimum cost is given by $s \xrightarrow{red} 3 \xrightarrow{blue} 4 \xrightarrow{orange} t$ with cost 3. When $k = 2$, the optimal solution is given by $s \xrightarrow{red} 1 \xrightarrow{blue} 2 \xrightarrow{red} t$ with cost 7.



Source: The author.

Among the applications of the k-CSPP, we have finding the shortest route between two stations in a transportation network (DEHOUCHE, 2020) and finding the shortest route between a pair of nodes, in a WDM network, using at most k distinct wavelengths, mapping each wavelength in an optical link to a color in the corresponding edge-colored graph (SANTOS *et al.*, 2017).

1.1 Related Works

Optimization problems defined on edge-colored graphs can encode qualitative information using colors (or labels). They can represent different alternatives of transportation in multi-modal networks or types of connections in computer systems.

Yuan *et al.* (2005) proposed the Minimum Color Path Problem (MCPP), motivated by its applications on the design of reliable networks. The problem consist in finding an (s,t)-path that uses as few colors as possible. Assuming each color has the same failure probability, minimizing the number of colors in the path also minimizes the failure probability of that path. On their work, Yuan *et al.* (2005) introduced two heuristics for the problem, mathematical formulations, as well as an edge-disjoint paths variation considering two distinct objective functions. More recently, Kumar (2019) proposed three classes of hard instances for the problem, these being layered graphs, unit disk intersection graphs, and road networks, as well as an $O(n^{2/3})$-approximation algorithm, and two greedy heuristics.

Regarding the k-CSPP, Ferone *et al.* (2019) proposed an Integer Linear Programming (ILP) formulation and a specialized branch-and-bound (B&B) algorithm. Following their work, Ferone *et al.* (2021) introduced a dynamic programming algorithm that outperformed previous methods. More recently, Cerrone and Russo (2023) introduced a polynomial-time Dijkstra-based heuristic and a graph reduction procedure for the problem, being able to significantly reduce the size of the benchmark instances.

The Single k-Multicolor Path Problem (SMPP) is closely related to the k-CSPP and was studied by Santos *et al.* (2017). It aims at finding a single (s,t)-path of minimum cost using exactly k colors, for a given $k \in \mathbb{Z}_+$. For the input graph of the SMPP, an edge is associated with a subset of colors. As mentioned earlier, the problem has applications in finding the shortest path between two nodes in a WDM network using exactly k distinct wavelengths.

Similar problems defined over edge-colored graphs were extensively studied in the literature. The Maximum-Flow Minimum-Label Problem (MF-ML), with applications in the pu-

rification of water in the distribution process (GRANATA *et al.*, 2013), aims to find the maximum flow that uses the minimum number of distinct labels. The k-Labeled Spanning Tree Problem consists of finding a spanning tree of minimum cost with at most k labels. Finally, the Orderly Colored Longest Path Problem (OCLPP), with applications in the field of genetics (CARRABS *et al.*, 2019), aims to find the longest path in an arc-colored digraph that follows a given color ordering.

1.2 Our objectives

There still is much to be explored regarding heuristics, valid inequalities, challenging instances, and exact method approaches for the k-CSPP. Our main objectives are to propose more challenging instances, new valid inequalities for the mathematical formulation of Ferone *et al.* (2019), and new models for the problem.

Regarding the new instances, we divide them into three groups. For all groups, the instances have at least 10^{15} distinct (s,t)-paths. The arc weights are selected uniformly at random from a given interval, rendering it challenging for the heuristic of Cerrone and Russo (2023) to identify a feasible solution and for the dynamic programming of Ferone *et al.* (2021) to execute effective pruning of unproductive paths. In order to explore the trade-off between the difficulty of finding a feasible solution and its cost, the second group of new instances was created, while the third was created to penalize greedy procedures.

For us, a set of instances is said to be challenging if, on average, it takes more time for the solver to obtain their optimal solutions than for resolving the benchmark ones. With that in mind, our work will be devoted to analyzing the following hypotheses: our valid inequalities strengthen the linear relaxation of the model proposed by Ferone *et al.* (2019); the solution techniques from the literature are not efficient for our new set of instances; and to check whether the new instances are more challenging than the ones from the literature.

The remainder of this document is organized as follows. In Chapter 2 we present exact methods, heuristics, and preprocessing algorithms introduced in the literature for the k-CSPP. In Chapter 3 we explore the main contributions of this work, that being a new exponential-size model for the k-CSPP and valid inequalities for the model introduced by Ferone *et al.* (2019). In Chapter 4 we present computational results for a set of instances of the literature and for the new instances. Finally, in Chapter 5 we present the concluding remarks.

2 SOLUTION APPROACHES FROM THE LITERATURE

In this chapter, we detail the methods proposed in the literature to solve the k-CSPP.

2.1 Exact Methods

In this section, we discuss the exact methods to solve the k-CSPP. In Section 2.1.1 we detail a polynomial-size integer programming model due to Ferone *et al.* (2019). In Section 2.1.2, we detail a branch-and-bound (B&B) algorithm (FERONE *et al.*, 2019). Lastly, in Section 2.1.3, we discuss a dynamic programming algorithm introduced by Ferone *et al.* (2021) that uses pruning strategies based on the concepts of dominating and feasible paths.

2.1.1 Problem formulation

Initially, we introduce some notation. For every vertex $v \in V$, we denote its node out-neighborhood (resp. in-neighborhood) by $\delta^+(v)$ (resp. $\delta^-(v)$). Let C be the set formed by the colors of D. The first model (FFP) for the problem is due to Ferone *et al.* (2019). Let x_{uv}, for all $(u,v) \in A$, be a decision variable to represent if arc (u,v) belongs to the solution, $x_{uv} = 1$, or $x_{uv} = 0$, otherwise. Let a decision variable y_h, for all $h \in C$, be equal to 1 if color h is in the solution path and 0, otherwise. Lastly, we denote by A_h the set of arcs of color h. The formulation is a follows.

$$\text{(FFP)} \quad \min \sum_{(u,v)\in A} d_{uv} x_{uv} \tag{2.1}$$

$$\text{s.t.} \sum_{v\in\delta^-(u)} x_{vu} - \sum_{v\in\delta^+(u)} x_{uv} = \begin{cases} -1 & \text{if } u = s \\ +1 & \text{if } u = t \\ 0 & \text{otherwise} \end{cases}, \quad \forall\, u \in V, \tag{2.2}$$

$$x_{uv} \leq y_h, \quad \forall\,(u,v) \in A_h,\ \forall h \in C \tag{2.3}$$

$$\sum_{h\in C} y_h \leq k, \tag{2.4}$$

$$x_{uv} \in \{0,1\}, \quad \forall\,(u,v) \in A, \tag{2.5}$$

$$y_h \in \{0,1\}, \quad \forall\, h \in C. \tag{2.6}$$

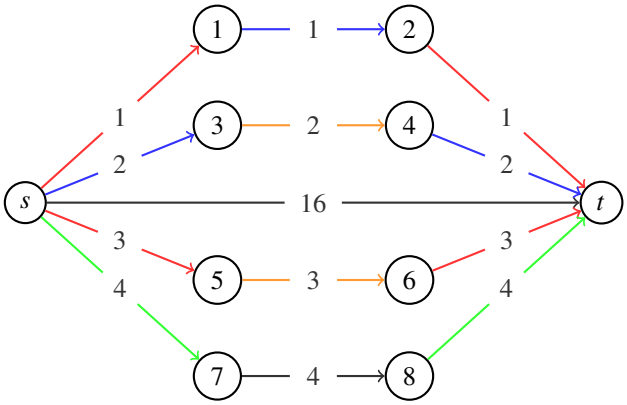
The objective function (2.1) minimizes the cost of the (s,t)-path. Flow conservation

constraints (2.2) guarantee the path connectivity. Constraints (2.3) ensure that if arc (u,v) is present in the path, then color $c(u,v)$ is also used. Constraint (2.4) imposes that at most k distinct colors are in the solution. Finally, constraints (2.5) and (2.6) are the domain of the variables. Model (FFP) has $O(|V|+|A|+1)$ constraints and $O(|A|+|C|)$ decision variables. Observe that if a given color does not belong to the solution, then all variables w.r.t. the arcs of that color can be set to zero. Because the integrality constraints (2.5) on the x variables, the integrality constraints (2.6) on y are irrelevant.

2.1.2 Branch-and-Bound Algorithm

In this section, we discuss the B&B algorithm for the k-CSPP proposed by Ferone *et al.* (2019). The idea of the B&B procedure is to solve a SPP for every node in the search tree by noticing that the relaxation of constraints (2.3) and (2.4) turns model (FFP) into the standard shortest path problem. Assuming that a node solution uses more than k colors, the procedure avoids using the arcs of some colors belonging to the solution and compute the shortest path in the resulting subgraph with the forbidden colors. Let z_i denote the solution of the i-th generated B&B node in the search tree. If $|C(z_i)| > k$, this node is partitioned into $|C(z_i)|$ subproblems, one for each color in z_i imposed not to belong to the solution. The order in which the nodes are generated is based on the absolute frequency of the colors on z_i, removing the lesser frequent colors first. The incumbent solution is then updated every time a feasible path with better solution value is found.

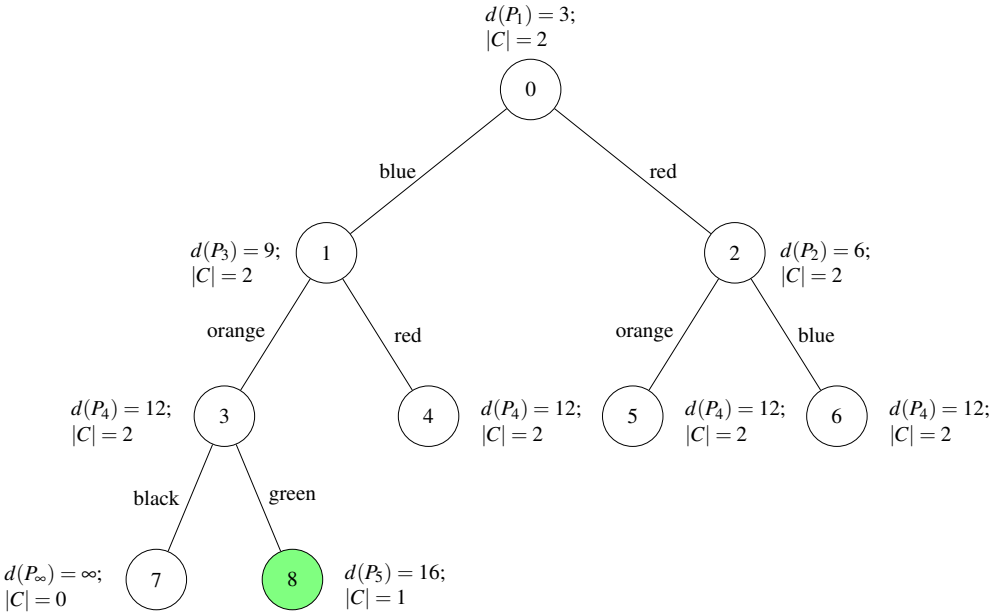
Figure 2 – Digraph instance for which the B&B procedure from literature (FERONE *et al.*, 2019) solves the same subproblem (i.e. $y_{\text{red}} = y_{\text{blue}} = 0$) twice.



Source: The author.

We observe that the procedure proposed by (FERONE *et al.*, 2019) can occasionally solve the same subproblem more than once. To illustrate, consider the digraph depicted in Fig. 2

Figure 3 – A partial B&B search tree for the algorithm proposed in the literature (FERONE *et al.*, 2019) executed on the digraph depicted in Fig. 2.



Source: The author.

and define $k = 1$. We show the corresponding partial B&B search tree in Fig. 3, obtained by using an implementation of the algorithm with a breadth-first search strategy. The cost and number of colors of the path are shown on the side of its corresponding node. We denote by P_∞ the case where no (s,t)-path exists on the subgraph. Initially, we calculate the shortest (s,t)-path of the instance shown in Fig. 2 in the root node 0 to obtain $P_1 = s \xrightarrow{red} 1 \rightarrow \xrightarrow{blue} 2 \xrightarrow{red} t$. Since $|C(P_1)| > k$, we execute the branching procedure on its colors, generating nodes 1 and 2 in the B&B tree to be solved in the subgraph without arcs of color blue and red, respectively. A feasible solution is found in node 8, shown in green. Notice that the subproblem where arcs of color blue and red are removed from the digraph is solved twice.

2.1.3 Dynamic Programming

Ferone *et al.* (2021) developed a label-setting dynamic programming algorithm to solve the k-CSPP to optimality. It relies on the concepts of feasible and dominated paths. Let $P_{s,u}$ denote an (s,u)-path. Define $P(u)$ as the set of all paths from s to u. A path $P_{s,u}$ is said to be feasible if $|C(P_{s,u})| \leq k$. Given two paths $P'_{s,u}$ and $P''_{s,u}$, path $P'_{s,u}$ is said to dominate $P''_{s,u}$ if $C(P'_{s,u}) \subseteq C(P''_{s,u})$ and $d(P'_{s,u}) \leq d(P''_{s,u})$. At least one of these conditions must be strict. The dominance condition avoids exploring unfruitful paths.

Algorithm 1: Dynamic Programming Algorithm

1: **Input:** An arc-colored digraph $D = (V, A, c, d)$, s, t, k
2: **Output:** A feasible path for the k-CSPP
3: **Initialization:** $P_s \leftarrow \{s\}$; $P(s) \leftarrow \{P_s\}$; $Q \leftarrow \{P_s\}$; $\Delta \leftarrow \infty$
4: **for** $v \in V \setminus \{s\}$ **do**
5: $\quad P(v) \leftarrow \emptyset$
6: **end for**
7: **while** $Q \neq \emptyset$ **do**
8: $\quad P \leftarrow$ EXTRACT(Q);
9: $\quad$ **if** $d(P) < \Delta$ **and** $last(P) = t$ **then**
10: $\quad\quad \Delta \leftarrow d(P)$
11: $\quad\quad best \leftarrow P$
12: $\quad$ **else**
13: $\quad\quad$ **for** $v \in N^+(last(P))$ **do**
14: $\quad\quad\quad P' \leftarrow P \cup \{v\}$
15: $\quad\quad\quad d(P') \leftarrow d(P) + d(last(P), v)$
16: $\quad\quad\quad C(P') \leftarrow C(P) \cup c(last(P), v)$
17: $\quad\quad\quad$ **if** CHECKFEASIBILITY$(C(P')) =$ TRUE **and** CHECKDOMINANCE$(v, d(P'), C(P')) =$ FALSE **then**
18: $\quad\quad\quad\quad$ Remove from $P(v)$ and Q all partial paths that are dominated by P'
19: $\quad\quad\quad\quad Q \leftarrow Q \cup P'$
20: $\quad\quad\quad$ **end if**
21: $\quad\quad$ **end for**
22: $\quad$ **end if**
23: **end while**
24: **return** *best*

For the algorithm, Ferone *et al.* (2021) proposed five extraction rules to choose the next subpath $P_{s,u}$ from the queue. For simplicity, let $last(P)$ be the last node in path P, and characterize a path by the set of nodes belonging to it. The first rule is a Dijkstra-like rule (DR), where the next path to be explored is the one of minimum cost. The authors also explored the standard First-In First-Out (FIFO) rule, where the extracted path is the one in the queue for the longest time, and Last-In First-Out (LIFO) rule, where we extract the last path inserted in the queue. The last rule is the Small-Label-First (SMF). In this rule, every time a new path P is to be added to the list of paths to a node, we check if $d(P) < d(P')$, where P' is the path currently at the head of the list. If the condition is satisfied, P is placed at the head of the list, otherwise it is placed at its tail. Lastly, in the A^* rule, we extract the path P where $d(P) + \pi(last(P), t)$ is minimum, where $\pi(last(P), t)$ corresponds to the shortest path value between the last visited node in P, say $last(P)$, and t.

A pseudocode for the dynamic programming algorithm is presented in Algorithm 1. In lines 3–6, the initial path P_s is initialized, containing only the source s, the list of paths $P(u)$

for $u \in V$, the path queue Q, and the objective value upper bound Δ. In line 7, we extract a path from Q using one of the five possible extracting rules explored by Ferone *et al.* (2021). In line 8, we extract a path from the queue or list based on one of the extraction rules defined above. In lines 9–12, we verify if P is an (s,t)-path and if its cost is less than the cost of the current best solution. If both conditions are true, then we update the upper bound Δ and the best solution, otherwise we continue the construction of an (s,t)-path. In lines 13–16, we expand for every node v in the out-neighborhood of $last(P)$, and update its cost as well as its colors. In line 17, we check if the current path is feasible and non-dominated by any other path in $P(last(P))$. To avoid wasting time, in line 18, we remove all paths dominated by P' from the queue and from the list of paths reaching $last(P')$. Finally, in line 19, we add the path P' to the queue. By assuming the A^* extraction policy, the algorithm runs in time $O((|V|-2)N(k)^2)$, where $N(k) = \sum_{l=1}^{k} \binom{|C|}{l}$.

2.2 Heuristics

Concerning existing heuristics for the k-CSPP, we have the work due to Cerrone and Russo (2023). The authors propose a constructive polynomial Dijkstra-based algorithm called Color Constrained Dijkstra Algorithm (CCDA) and a graph reduction algorithm (GRA). The latter is detailed in the next section.

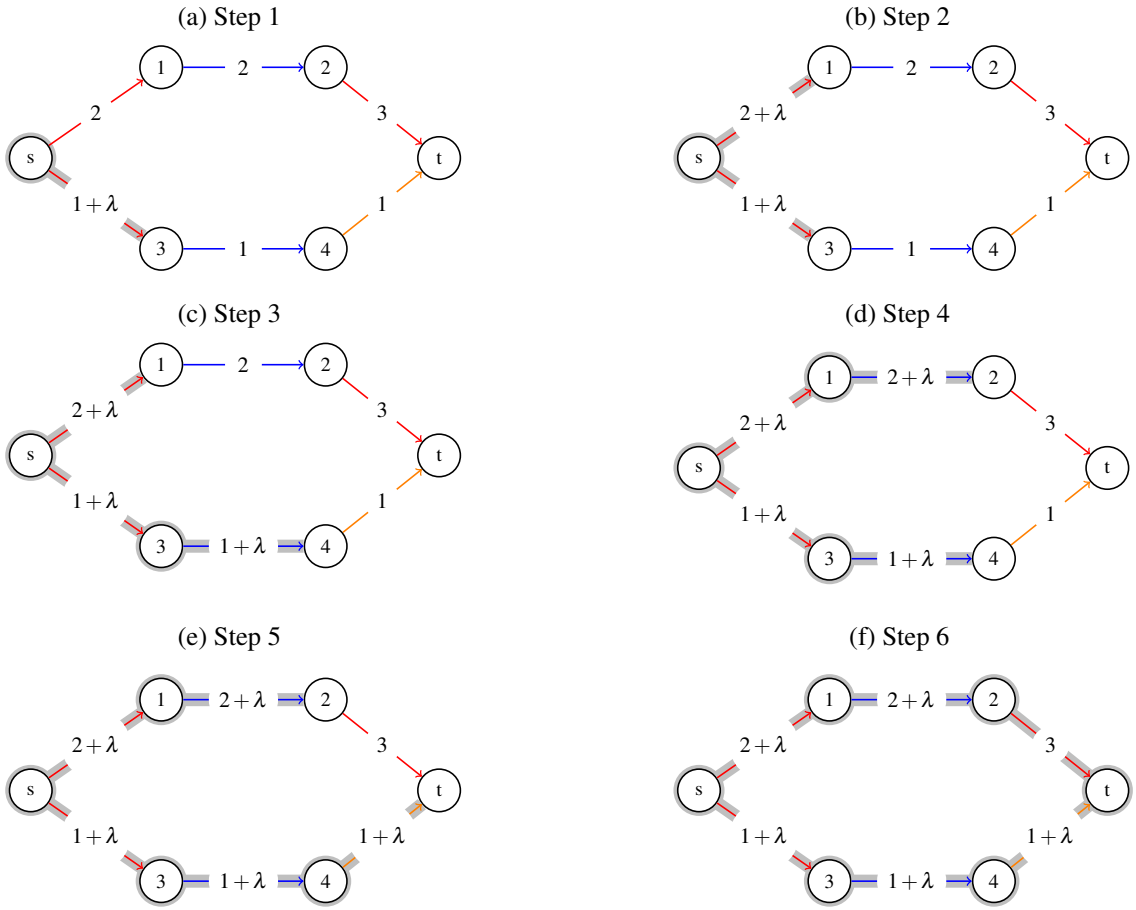
The idea behind CCDA is to iteratively construct an (s,t)-path using Dijkstra's algorithm, while imposing penalties for when new colors are used to reach a vertex. For this, the penalties are chosen from a list of values Λ previously defined (CERRONE; RUSSO, 2023). Let $w_{\min}$ denote the minimum arc cost in the digraph, w_{mean} the average arc cost and $w_{\max}$ the maximum. In the literature, Λ is defined by $\{0, w_{\min}/4, w_{\min}/2, w_{\min}, 2w_{\min}, w_{\text{mean}}/4, w_{\text{mean}}/2, w_{\text{mean}}, w_{\max}\}$ (CERRONE; RUSSO, 2023).

The penalized Dijkstra's algorithm works as usual, except the way the arc costs are defined. The CCDA considers, in increasing order, the values of penalties $\lambda \in \Lambda$, one at a time, starting from the smallest to the largest value. When updating the estimate of the path cost from s to a vertex $v \in V \setminus \{s\}$, if $c(u,v)$ is not present in the partial (s,v)-path, then one considers the penalized arc cost $d_{uv} + \lambda$. For $\lambda = 0$, the algorithm is equivalent to the standard Dijkstra algorithm. In addition to the predecessor of each vertex, it is also necessary to keep the list of colors in the (s,v)-path. The algorithm is intended to stop with a feasible path when it reaches t. The complexity of CCDA depends on the one of the classic Dijkstra's algorithm and on the number of penalty values in Λ, that being $O(|\Lambda|\mathscr{D})$ where $\mathscr{D}$ is the complexity of Dijkstra's

algorithm.

For some instances, the CCDA (CERRONE; RUSSO, 2023) stops without returning any feasible solution. The effectiveness of the heuristic depends on the interval of the penalty values defining Λ. Indeed, in Fig. 4, if we define a penalty value λ in the interval $[0,3]$ as the literature suggests (CERRONE; RUSSO, 2023), then for every penalty λ, CCDA always proposes as solution the path $s \xrightarrow{red} 3 \xrightarrow{blue} 4 \xrightarrow{orange} t$ of three colors. This should be a problem if $k = 2$. To overcome this drawback, we allow the maximum value of λ to assume values larger than the maximum arc cost present in the digraph, e.g., $2 \times w_{\max}$. For the example in Fig. 4, the value $\lambda = 6$ leads CCDA finding the optimal two-colors path $s \xrightarrow{red} 1 \xrightarrow{blue} 2 \xrightarrow{red} t$.

Figure 4 – Execution of the CCDA heuristic for a digraph instance for which it fails to obtain a feasible (s,t)-path with at most $k = 2$, if we adopt penalties from the interval $\lambda \in [0,3]$. The notation on each arc (u,v) is $d_{uv} + \lambda$, except for the arc $(2,t)$ of cost 3 because its color is the same as the one of the arc $(s,1)$.



Source: The author.

2.3 Pre-processing

Cerrone and Russo (2023) also proposed a graph reduction algorithm for the k-CSPP. The general idea behind the algorithm is to use an upper bound ub to eliminate unfruitful paths from the digraph. The algorithm removes a vertex v from the digraph if the shortest path containing it, i.e. the concatenation of the shortest (s,v)-path and (v,t)-path, is more expensive than ub. Reducing the size of an instance can be crucial to achieve lower computational times in large case scenarios. To obtain an upper bound the authors employed the heuristic detailed in the previous section.

Proposition 2.1. *(CERRONE; RUSSO, 2023) Consider a graph $G = (V,E)$, where $s,t \in V$ are respectively defined as the source and destination vertices. Let $P(u,v)$ denote a generic path and consider a vertex $v \in V$. If $\pi(s,v) + \pi(v,t) > d(P(s,t))$, then all paths that pass through v have cost higher than $d(P(s,t))$.*

We show a pseudocode of this procedure in Algorithm 2. Lines 3–4 iterates through every vertex u in the digraph that does not belong to the feasible solution used for the upper bound, except for the source and destination nodes. If u is not part of the feasible path $\bar{P}$ found by the heuristic of Cerrone and Russo (2023), and the cost of the shortest (s,t)-path containing it is greater than the current upper bound ub, we proceed with its removal. Lines 5–7 removes u and all its adjacent arcs from the input digraph D.

Algorithm 2: Graph Reduction Algorithm

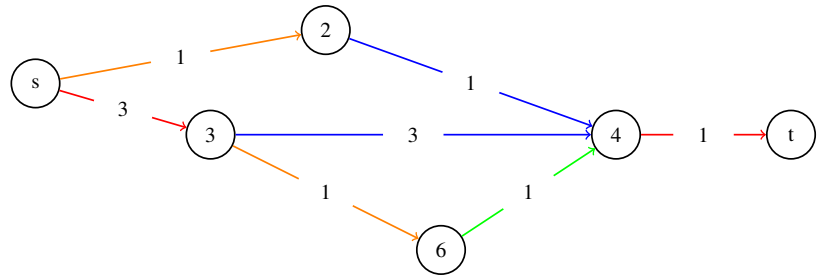
```
1: Input: An arc-colored digraph D = (V,A,c,d), a feasible path P̄, and ub.
2: Output: A reduced arc-colored digraph D.
3: for v ∈ V \ s,t do
4:    if v ∉ P̄ and π(s,v) + π(v,t) > ub then
5:       V' ← V \ {v};
6:       A' ← A \ {(v,u) ∈ A : u ∈ δ⁻(v)} ∪ {(u,v) ∈ A : u ∈ δ⁺(v)};
7:       D = (V',A',c,d);
8:    end if
9: end for
```

The procedure's efficiency heavily depends on the quality of the feasible solution found by a heuristic. It may be beneficial to spend more time in search for a feasible path and employing strategies to improve upon the solution found. This presents an obstacle for the effective use of both the heuristic introduced in the previous section and the reduction procedure.

3 VALID INEQUALITIES

In this section, we propose valid inequalities for the k-CSPP. We observe that the following inequalities are also valid for general single-path problems in arc-colored digraphs. We cawn also extend the valid inequalities proposed in this chapter to problems in edge-colored degree-constrained tree problems. For that, let α^+ (resp. α^-) be the maximum number of arcs that can leave (resp. enter) a node in the corresponding digraph. It is sufficient to multiply the right-hand side of the inequalities by the corresponding maximum degree constant. In fact, for single-path problems, we assume $\alpha^+ = \alpha^- = 1$. To begin our discussion, consider the instance depicted in Fig. 5 and assume a standard SPP integer linear programming formulation.

Figure 5 – An instance of the k-CSPP for $k = 2$.



Source: The author.

By solving the model for the instance above, we obtain the path $P_0 = \{(s,2),(2,4),(4,t)\}$ with three colors and cost 3. Because P_0 is infeasible, there is at least 1 color belonging to P_0 that is not used in a feasible solution. To remove the extra color, we isolate the arcs of same color, constructing the inequality $x_{s,2} + x_{2,4} + x_{4,t} \leq 2$. After its addition into the model, we solve it to obtain $P_1 = \{(s,3),(3,6),(6,4),(4,t)\}$ with three colors and cost 6, which is also infeasible, and consequently we construct the valid inequality $\frac{1}{2}(x_{s,3} + x_{4,t}) + x_{3,6} + x_{6,4} \leq 2$. For the model with the two newly added inequalities, the solution is $P^* = \{(s,3),(3,4),(4,t)\}$ with two colors and cost 7. From this, observe that we can estimate the value of a variable y_h in any (s,t)-path P according to $(1/|A_h^P|) \sum_{(u,v)\in A_h^P} x_{uv}$, where A_h^P denotes the set of arcs of P of color h. The set of distinct colors in P, say $C(P)$, must contain at most k elements.

We now present an exponential-size formulation for the k-CSPP. It is based on the previous inequality to cut off any infeasible path having more than k distinct arc colors.

$$\text{(PCM)} \quad \min \sum_{(u,v)\in A} d_{uv}x_{uv}$$

$$\text{s.t. } (2.2), (2.5), \quad \text{and}$$

$$\sum_{h\in C(P)} \frac{1}{|A_h^P|} \sum_{(u,v)\in A_h^P} x_{uv} \leq k, \quad \forall\, P \in \mathscr{P}, \tag{3.1}$$

where $\mathscr{P}$ stands for the set of all (s,t)-paths of D. Model (PCM) has $O(|V|!)$ constraints and $O(|A|)$ variables. Because of the number of constraints, we explore this formulation as cuts in a branch-and-cut (B&C) algorithm.

Now, we note that in any (s,t)-path, at most one arc leaves or enters a vertex $u \in V$. Consequently,

$$\sum_{v\in\delta^+(u)} x_{uv} \leq 1, \quad \forall\, u \in V \setminus \{t\}, \tag{3.2}$$

$$\sum_{v\in\delta^-(u)} x_{vu} \leq 1, \quad \forall\, u \in V \setminus \{s\}. \tag{3.3}$$

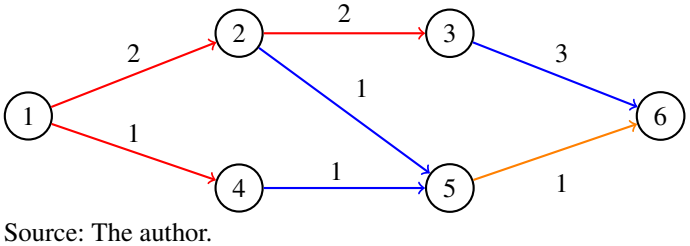
Although inequalities (3.2) and (3.3) are straightforward, from them, we propose the next set of valid inequalities regarding arcs of the same color leaving or entering a given vertex $u \in V$.

Proposition 3.1. *In any (s,t)-path, the number of arcs of color h leaving (or entering) a vertex u is limited above by y_h.*

$$\sum_{\substack{v\in\delta^+(u)\\ c(u,v)=h}} x_{uv} \leq y_h, \quad \forall\, u \in V, \forall\, h \in C, \tag{3.4}$$

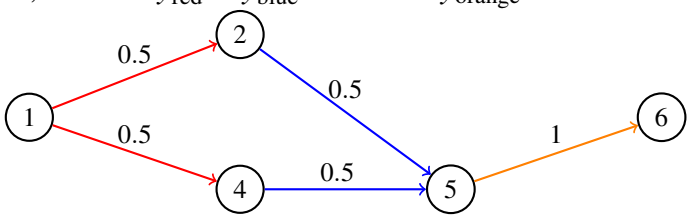
$$\sum_{\substack{v\in\delta^-(u)\\ c(v,u)=h}} x_{vu} \leq y_h, \quad \forall\, u \in V, \forall\, h \in C. \tag{3.5}$$

Figure 6 – Instance: arc-colored digraph D_1. This instance has $A_{\text{red}} = \{(s,1),(1,2),(s,3)\}$, $A_{\text{blue}} = \{(1,4),(3,4),(2,t)\}$, and $A_{\text{orange}} = \{(4,t)\}$.



Source: The author.

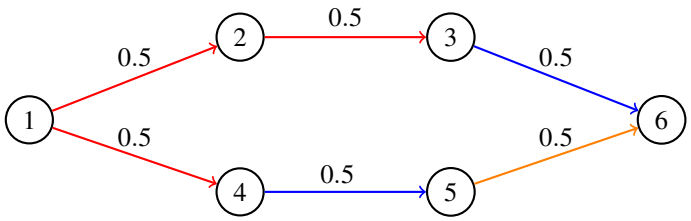
Figure 7 – Optimal linear relaxed solution of cost 3.5 for the instance depicted in Fig. 6 for $k = 2$. In this solution, we have $y_{\text{red}} = y_{\text{blue}} = 0.5$ and $y_{\text{orange}} = 1$.



Source: The author.

The proof of Proposition 3.1 is straightforward. Note that both (3.4) and (3.5) cut off fractional solutions where two or more arcs of the same color leave or enter a given node. For example, consider the instance D_1 shown in Fig. 6. Its optimal linear relaxation solution is depicted in Fig. 7. By adding inequalities $x_{1,2} + x_{1,4} \leq y_{\text{red}}$ and $x_{2,5} + x_{4,5} \leq y_{\text{blue}}$ we improve on the value of the linear relaxation objective value, resulting in the support graph depicted in Fig. 8 with cost equal to 5. We observe that for the resulting relaxed solution, $y_{\text{red}} = 1$. Constraints (3.4)–(3.5) also strengthen some constraints (2.3).

Figure 8 – Optimal linear relaxed solution for the instance depicted in Fig. 6 for $k = 2$ of cost 5 and addition of inequalities $x_{1,2} + x_{1,4} \leq y_{\text{red}}$ and $x_{2,5} + x_{4,5} \leq y_{\text{blue}}$. In this solution, we have $y_{\text{red}} = 1$ and $y_{\text{blue}} = y_{\text{orange}} = 0.5$.

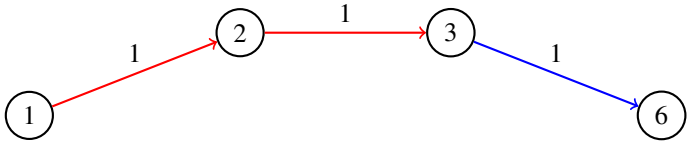


Source: The author.

We can extend the idea of Proposition 3.1 for pairs of non-reachable vertices. Let $R(u)$ be the set of vertices reachable by u in D. If for two vertices u and v we have $v \notin R(u)$ and $u \notin R(v)$, i.e., there is neither an (u,v)-path nor a (v,u)-path in the digraph, then the number of arcs of color h leaving (resp. entering) or leaving one vertex and entering another one is limited by y_h. A pair of vertices satisfying the aforementioned condition can be easily obtained by running a breadth-first search or depth-first search on the support graph, or by using a naive $O(|V|^2 + |V||A|)$ algorithm on the original digraph.

Proposition 3.2. *Consider vertices $u, v \in V$ such that $v \notin R(u)$ and $u \notin R(v)$. The number of*

Figure 9 – Optimal relaxed solution of cost 7 for the instance depicted in Fig. 6 for $k = 2$ with the addition of inequalities $x_{1,2} + x_{1,4} \leq y_{\text{red}}$, $x_{1,4} + x_{2,3} \leq y_{\text{red}}$, and $x_{2,5} + x_{4,5} + x_{3,6} \leq y_{\text{blue}}$.



Source: The author.

arcs of color h entering or leaving u and v is limited above by y_h.

$$\sum_{\substack{j \in \delta^+(v) \\ c(v,j)=h}} x_{vj} + \sum_{\substack{j \in \delta^+(u) \\ c(u,j)=h}} x_{uj} \leq y_h, \quad \forall\, u,v \in V,\ v \notin R(u),\ u \notin R(v),\ \forall\, h \in C, \tag{3.6}$$

$$\sum_{\substack{j \in \delta^+(v) \\ c(v,j)=h}} x_{vj} + \sum_{\substack{j \in \delta^-(u) \\ c(j,u)=h}} x_{ju} \leq y_h, \quad \forall\, u,v \in V,\ v \notin R(u),\ u \notin R(v),\ \forall\, h \in C, \tag{3.7}$$

$$\sum_{\substack{j \in \delta^-(v) \\ c(j,v)=h}} x_{jv} + \sum_{\substack{j \in \delta^-(u) \\ c(j,u)=h}} x_{ju} \leq y_h, \quad \forall\, u,v \in V,\ v \notin R(u),\ u \notin R(v),\ \forall\, h \in C. \tag{3.8}$$

Proof. As $v \notin R(u)$ and $u \notin R(v)$, u and v cannot appear simultaneously in a feasible solution. Thus, at most one of the arcs of color h can leave or enter both vertices, or leave one and enter the other vertex if this color belongs to the solution. □

To illustrate Proposition 3.2, consider D_1 and its optimal linear relaxation shown in Fig. 8 after the addition of inequalities $x_{1,2} + x_{1,4} \leq y_{\text{red}}$ and $x_{2,5} + x_{4,5} \leq y_{\text{blue}}$. In addition to these inequalities, derived from Proposition 3.1, we also derive the inequalities $x_{1,4} + x_{2,3} \leq y_{\text{red}}$ and $x_{2,5} + x_{4,5} + x_{3,6} \leq y_{\text{blue}}$ by noticing that vertices 3 and 5 can not belong to the same solution. The same applies to vertices 2 and 4. Their addition into the model allow us to obtain an integer solution with cost 7. We note that inequalities (3.6)–(3.8) strengthen inequalities (3.4) and (3.5).

The idea can be further extended for any set $N \subseteq V$ containing only non-reachable vertices. Since $u \notin R(v)$ and $v \notin R(u)$ for all pairs $u,v \in N$, then we can limit the total number of arcs of the same color that can leave (resp. enter) this set. A formal description is given in Proposition 3.3.

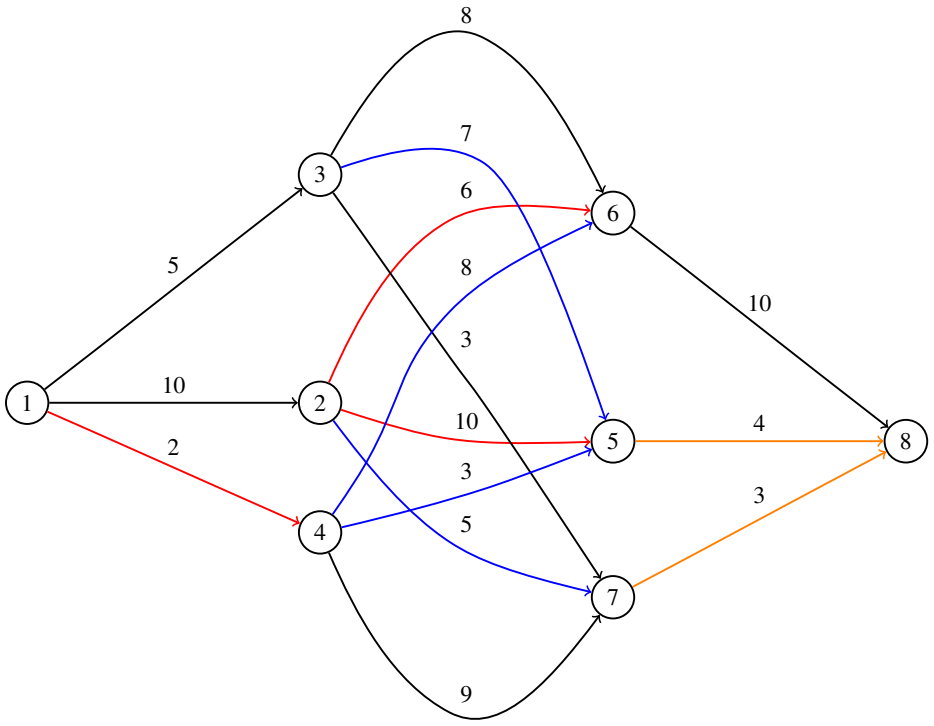
Proposition 3.3. *Let $N \subseteq V$ be such that for any two vertices u and v of N, $v \notin R(u)$ and $u \notin R(v)$. Let $Q \subseteq N$. The number of arcs of color h leaving Q and entering $N \setminus Q$ is limited above by y_h if color h belongs to the solution.*

$$\sum_{u \in Q} \sum_{\substack{j \in \delta^+(u) \\ c(u,j)=h}} x_{uj} + \sum_{v \in N \setminus Q} \sum_{\substack{j \in \delta^-(v) \\ c(j,v)=h}} x_{jv} \leq y_h, \quad \forall\, N \subseteq V,\ \forall\, Q \subseteq N,\ \forall\, h \in C. \tag{3.9}$$

Proof. The result follows from the fact that at most one of the arcs of a given color h incident to the non-reachable vertices of N can belong to the solution if this color also belongs. □

The following proposition explores the fact that if there exists an arc (u,v) such that u is not reachable by v, and there exists vertices $j \in N^+(u) \setminus \{v\}$ that do not reach v, then only one of the arcs of the same color, say h, leaving u and not reaching v as well as from v to $\delta^+(v)$ can belong to the solution.

Figure 10 – Instance: arc-colored digraph D_2. For this instance, we have $A_{\text{red}} = \{(1,4), (2,5), (2,6)\}$, $A_{\text{blue}} = \{(2,7), (3,5), (4,5)\}$, $A_{\text{orange}} = \{(5,8), (7,8)\}$, and $A_{\text{black}} = \{(1,2), (1,3), (3,6), (3,7), (4,7), (6,8)\}$.



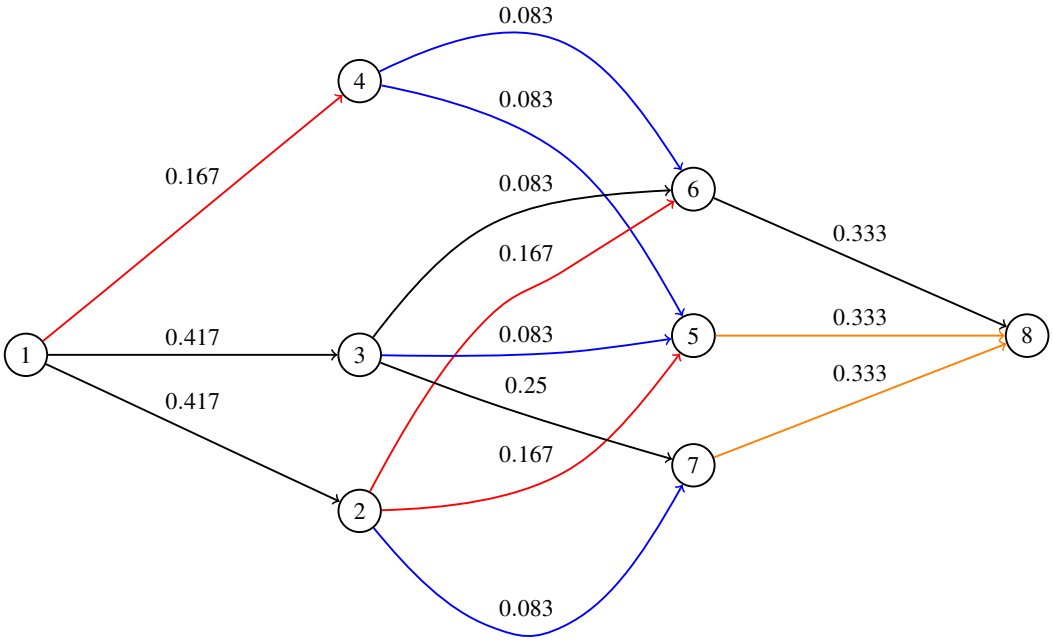
Source: The author.

Proposition 3.4. *For every arc $(u,v) \in A$ such that $u \notin R(v)$, the number of color h arcs from u to vertices in $\delta^+(u) \setminus \{v\}$ that does not reach v, and leaving v of that color, is limited above by y_h.*

$$\sum_{\substack{j \in \delta^+(u) \setminus \{v\} \\ v \notin R(j) \\ c(u,j)=h}} x_{uj} + \sum_{\substack{j \in \delta^+(v) \\ c(v,j)=h}} x_{vj} \leq y_h, \quad \forall\, (u,v) \in A, u \notin R(v), \forall\, h \in C. \tag{3.10}$$

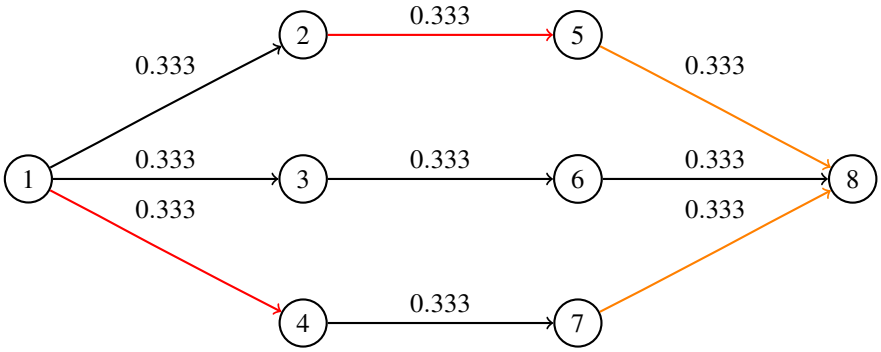
Proof. Consider an arc $(u,v) \in A$. By assumption, $u \notin R(v)$. For all $j \in \delta^+(u) \setminus \{v\}$ with $v \notin R(j)$, the arcs (u,j) of any color h cannot be used in the same solution with arcs leaving v of

Figure 11 – Optimal relaxed solution for the instance depicted in Fig. 10 of cost 18.25 for $k = 1$. This solution has $y_{\text{red}} \approx 0.167$, $y_{\text{blue}} \approx 0.083$, $y_{\text{orange}} \approx 0.333$, and $y_{\text{black}} \approx 0.417$.



Source: The author.

Figure 12 – Optimal relaxed solution for the instance depicted in Fig. 10 of cost approximately 20.33 for $k = 1$ with the addition of inequality $x_{3,7} + x_{6,8} \leq y_{\text{black}}$. This solution has $y_{\text{red}} \approx 0.333$, $y_{\text{orange}} \approx 0.333$, and $y_{\text{black}} \approx 0.333$.



Source: The author.

this color. Consequently, the sum of the corresponding x variables for these arcs is limited above by y_h. □

We give an example of Proposition 3.4 for the digraph D_2 shown in Fig. 10. Let us assume $k = 1$, the optimal linear relaxation for the instance is depicted in Fig. 11. Consider the arc $(3,6)$ and the vertices 5 and 7 from the out-neighborhood of 3. Since there is only one color in the out-neighborhood of the vertex 6, we only consider the arcs $(3,7)$ and $(6,8)$ of color black. Thus, from inequality (3.10), we derive the valid inequality $x_{3,7} + x_{6,8} \leq y_{\text{black}}$ for this instance. The addition of this inequality slightly improves the linear relaxation, as shown in Fig. 12.

We now further generalize the idea behind Proposition 3.4. Let us denote by $N_v \subseteq$

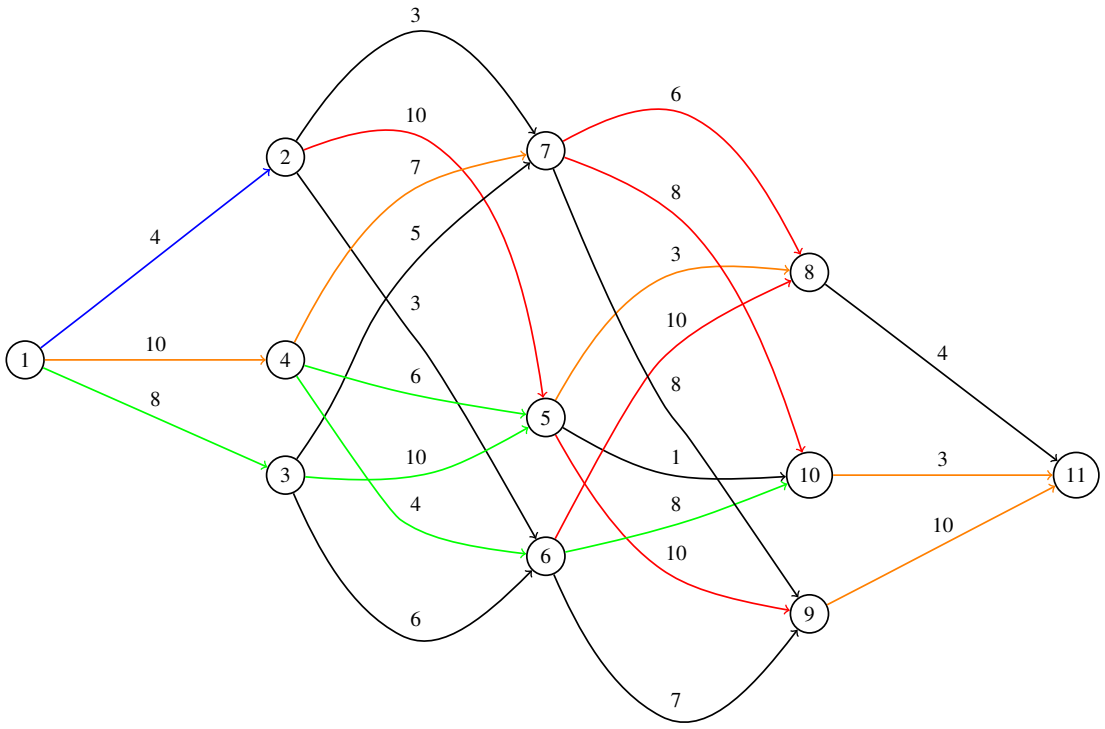
$\delta^-(v)$ a maximal set of vertices such that for all pairs of vertices $u, v \in N_v$, we have that $u \notin R(v)$ and $v \notin R(u)$.

Proposition 3.5. *If $N_v \neq \emptyset$, for some vertices $v \in V \setminus \{s,t\}$, then the number of arcs of a color $h \in C$ leaving N_v to vertices $j \in \delta^+(N_v) \setminus \{v\}$, such that $v \notin R(j)$, and those leaving v of this color is at most y_h.*

$$\sum_{u \in N_v} \sum_{\substack{j \in \delta^+(u) \\ v \notin R(j) \\ c(u,j)=h}} x_{uj} + \sum_{\substack{j \in \delta^+(v) \\ c(v,j)=h}} x_{vj} \leq y_h, \quad \forall\, v \in V \setminus \{s,t\},\ N_v \subseteq \delta^-(v),\ \forall\, h \in C \tag{3.11}$$

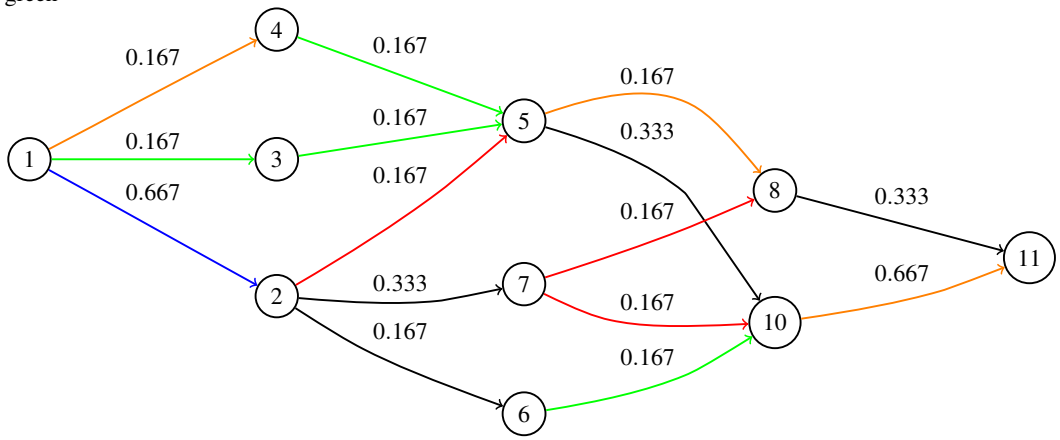
Proof. Consider a vertex $v \in V \setminus \{s,t\}$ and a color $h \in C$ for which there is at least one arc from N_v to $\delta^+(N_v) \setminus \{v\}$ of this color. By definition of N_v, at most one arc of these arcs can belong to the solution because their heads do not reach each other. The same is valid for the arcs leaving v. Moreover, neither these heads reach v nor v reach them. Consequently, at most one of all these arcs can be in the solution and, in particular, the ones of color h if it is in the solution. Thus, the result follows. □

Figure 13 – Instance: arc-colored digraph D_3. For this instance, $A_{\text{red}} = \{(2,5),(5,9),(6,8),(7,8),(7,10)\}$, $A_{\text{blue}} = \{(1,2)\}$, $A_{\text{orange}} = \{(1,4),(4,7),(5,8),(9,11),(10,11)\}$, $A_{\text{black}} = \{(2,6),(2,7),(3,6),(3,7),(5,10),(6,9),(7,9),(8,11)\}$, and $A_{\text{green}} = \{(1,3),(3,5),(4,5),(4,6),(6,10)\}$.



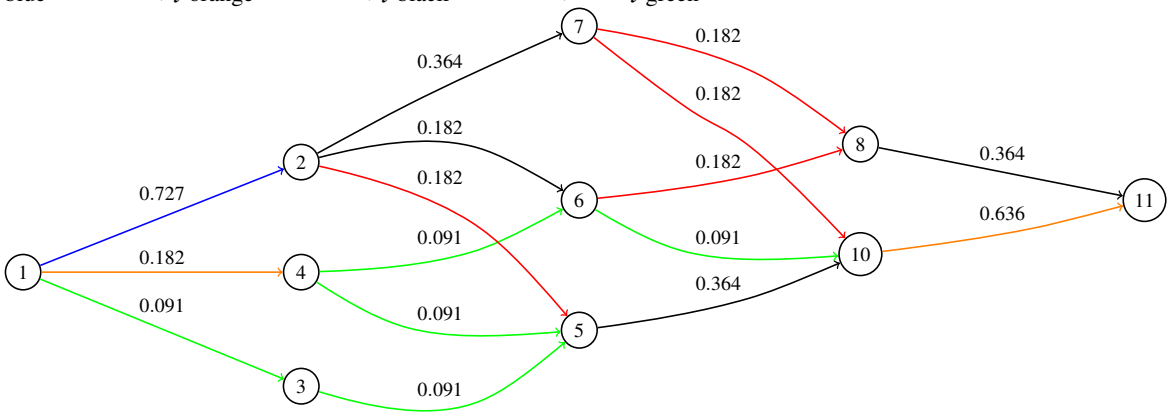
Source: The author.

Figure 14 – Optimal relaxed solution for the instance depicted in Fig. 10 for $k = 2$. The solution has cost 19.33. This solution has $y_{\text{red}} \approx 0.167$, $y_{\text{blue}} \approx 0.667$, $y_{\text{orange}} \approx 0.667$, $y_{\text{black}} \approx 0.333$, and $y_{\text{green}} \approx 0.167$.



Source: The author.

Figure 15 – Optimal relaxed solution for the instance depicted in Fig. 10 for $k = 2$ of cost 19.54 with the addition of inequality $x_{2,6} + x_{8,11} \leq y_{\text{black}} + x_{6,8}$. This solution has $y_{\text{red}} \approx 0.182$, $y_{\text{blue}} \approx 0.727$, $y_{\text{orange}} \approx 0.636$, $y_{\text{black}} \approx 0.364$, and $y_{\text{green}} \approx 0.091$.



Source: The author.

The next valid inequality is based on the fact that, if two arcs (u,w) and (v,r) of the same color belong to the solution, then there must also be a (w,v)-path or a (r,u)-path.

Proposition 3.6. *If two non-consecutive arcs of the same color belong to the solution, then this color also belongs, and we must have at least an arc between them.*

$$x_{uw} + x_{vr} \leq y_h + \sum_{\substack{j \in \delta^+(w) \\ v \in R(j)}} x_{wj} + \sum_{\substack{j \in \delta^+(r) \\ u \in R(j)}} x_{rj}, \quad \forall\, h \in C,\ \forall\, (u,w),(v,r) \in A_h. \tag{3.12}$$

Proof. If two non-consecutive arcs of color h, say $(u,w),(v,r) \in A_h$, belong to the solution, then color h must also belong. Moreover, one arc must leave w and reach v or one arc must leave r and reach u. This is true by definition of a simple path containing both arcs. Thus, the result follows. □

Consider the instance D_3 depicted on Fig. 13. Its optimal linear relaxed solution is shown in Fig. 14 for $k = 2$ and has cost approximately 19.33. Observe that the black arcs $(2,6)$ and $(8,11)$ can only be part of the same solution if the arc $(6,8)$ is also used. From this, we derive the valid inequality $x_{2,6} + x_{8,11} \leq y_{\text{black}} + x_{6,8}$, that slightly improves the linear relaxed solution value.

The following valid inequality explores the colorful cuts of Silva *et al.* (2019), where $C[S,T]$, with $s \in S$ and $t \in T$, stands for the set of colors of an (s,t)-cut of arcs $[S,T]$ of D. The idea is that at least one of the colors present in every (s,t)-cut of D must belong to the solution.

$$\sum_{h \in C[S,T]} y_h \geq 1, \quad \forall\, [S,T]\, of\, D. \tag{3.13}$$

In particular, if we consider an (s,t)-cut $[S, V \setminus S]$ of D such that $s \in S$, $t \in V \setminus S$, and $[V \setminus S, S] = \emptyset$, then the sum of the arcs of color h in $[S, V \setminus S]$ is limited above by y_h according to

$$\sum_{\substack{(u,v) \in [S, V \setminus S] \\ c(u,v) = h}} x_{uv} \leq y_h, \quad \forall\, S \subseteq V \mid s \in S, t \in V \setminus S,\ \forall\, h \in C[S, V \setminus S] \tag{3.14}$$

We can observe various occurrences of the condition for inequality (3.14) in previous instances used to exemplify the propositions above. For example, consider the instance depicted in Fig. 13 and its optimal linear relaxed solution shown in Fig. 14. By choosing the cut $\{(4,5),(3,5),(2,5),(7,8),(7,10),(6,10)\}$ we can derive the inequalities $x_{2,5} + x_{7,8} + x_{7,10} \leq y_{\text{red}}$ and $x_{4,5} + x_{3,5} + x_{6,10} \leq y_{\text{green}}$.

4 COMPUTATIONAL EXPERIMENTS

In this chapter, we present numerical experiments performed on a PC Intel Core i7-3770, 8 $\times$ 3.40 GHz, 16 GB DDR3 RAM with Ubuntu 20.4 LTS 64 bits. We use Julia 1.8.5 with JuMP package to implement the mathematical programming models in CPLEX 22.1 configured with one thread. The time limit for each instance is set to 1800 seconds for all the procedures described in this document.

For the experiments, we adopt benchmark instances (grid and random digraphs) from Ferone *et al.* (2019) and generate new classes (groups) of layered-based digraphs. Similar instances showed to be hard to handle for the MCPP (KUMAR, 2019). Each layered digraph is composed of w layers of r vertices per layer, in addition to a source s and a destination t vertices. The source s connects to every vertex of the first layer, while all the vertices in the last layer connect to the destination t. In a standard layered digraph, there is an arc from every vertex of layer i to every vertex of layer $i+1$, with $i := 1, \cdots, w-1$.

We generate 60 new instances for the k-CSPP, divided into three groups of 20 instances. All the new test-bed digraphs have $2+15\times 10$ vertices: a source, a destination, as well as $w = 15$ columns (layers) with $r = 10$ vertices each one. The groups are categorized as follows:

- Group 1 contains standard layered digraphs with $wr+2$ vertices and $2r+(w-1)r^2 = 1420$ arcs. We uniformly choose their arc costs from the integer interval $[1, 1000]$;
- Group 2 is composed of modified layered digraphs. We first generate a digraph as those of Group 1. Then, we create an arbitrary number (from the integer interval $[10, 30]$) of jump-arcs. To obtain a jump-arc (u, v), we randomly choose u and v from non-neighbor layers L_i and $L_{i'}$, respectively, with $i < i'$. We uniformly choose the cost of a jump-arc from the integer interval $[d_{\max}, d_{\max} + 30000]$, where $d_{\max}$ is the highest arc cost among non-jump-arcs. The idea behind using a few jump-arcs with high costs is to allow the heuristics to easily find a feasible (possible costly) path;
- Group 3 has digraphs as of Group 1, but initially with an empty set of arcs. For each layer L_i, $i := 1, \cdots, w-1$, we select at random 3 distinct vertices and form a directed clique Q_i with them. Then, for each pair of vertices (u, v), where $u \in Q_i$ and $v \in L_i - Q_i$, we create an arc (u, v) with probability $p = 1/2$. Finally, we add an arc from every vertex of layer L_i, excepting from the vertex with smaller label of Q_i, to every vertex of layer L_{i+1}. All arc costs of this group are chosen from the integer interval $[1, 1000]$. The digraph topology

of this group tends to force CCDA heuristic to obtain non-fruitful paths while executing its greedy search strategy.

Arc colors of our new instances are chosen based on a uniform distribution over the integer interval $[1,|C|]$, where we set $|C| = \lfloor |A|/4 \rfloor$. Finally, we define k as the minimum number of colors allowing an (s,t)-path, obtained after solving the MCPP for each instance individually. For instances of Group 2, we evaluate the MCPP in the digraph without jump-arcs. Both choices for $|C|$ and k are based on an extensive set of preliminary tuning experiments.

Given the shortest (s,t)-path having k' colors, the k-CSPP is easy for values of $k \geq k'$. To give an idea of the problem difficulty (in terms of cpu execution time) for values of $k < k'$ and distinct values of $|C|/|A|$, we depict some related experiments in Fig. 16 for four layered digraphs obtained as those of Group 1 with the same number of vertices and arcs. The axes in Fig. 16

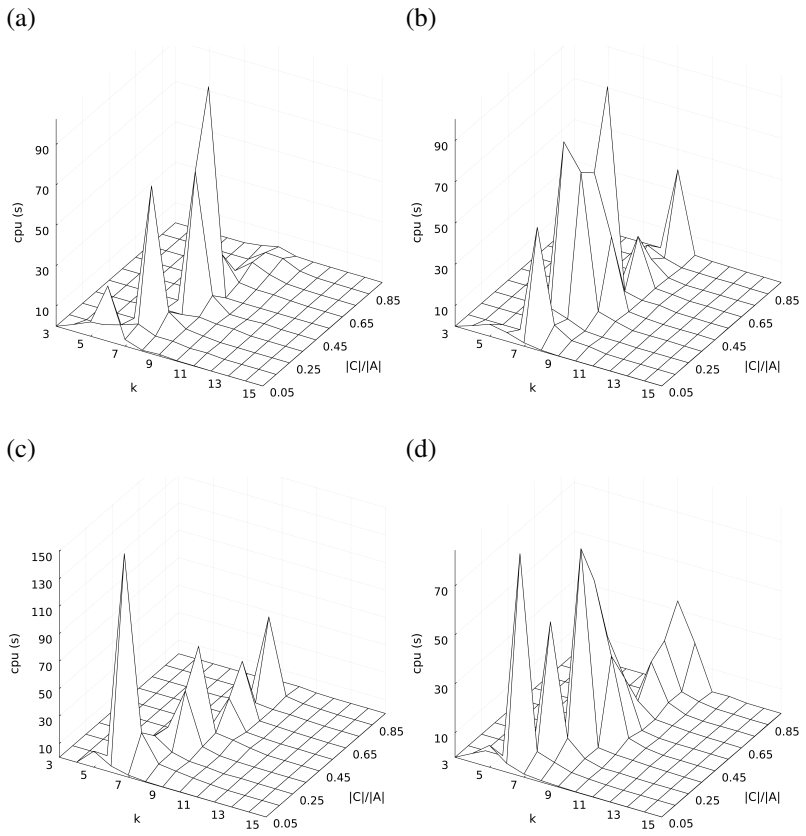


Figure 16 – Variation of execution time in function of $|C|/|A|$ and k for 4 layered digraphs.

represent the color density $|C|/|A|$, the maximum number of colors k in the solution path, and the execution time in seconds *cpu*. We observe that the instances require higher computational time when $|C|/|A| \approx 0.25$ and the values of k are equal to the minimum number of colors related to the MCPP solutions for these digraphs.

Tables 1 and 2 show the characteristics of the new instances of Groups 1, 2, and 3, and of the random and grid digraphs (FERONE *et al.*, 2019), respectively. For the legend of these tables, we identify each instance in column 'inst'. In such column, each instance receives a label from 1 to 60 in Table 1. In Table 2, the instance identifier corresponds to the original instance name as by Ferone *et al.* (2019). In Table 1, all the instances have $|V| = 152$ vertices and the number of arcs $|A|$ varies according to each group. The limit on the number of colors is k, and the known optimal solution value is opt.

We apply the graph reduction algorithm to eliminate arcs proved not to belong to the optimal solution of these instances by using the feasible solution found by the CCDA heuristic (CERRONE; RUSSO, 2023). We use the heuristic solution in the CPLEX solver as cutoff value. For the instances where we have a reduction on their size, $R(V)$ and $R(A)$ denote, respectively, the reduced set of vertices and arcs. We have to mention that the reduction algorithm was not able to remove any arc or vertex of the instances of Groups 1, 2, and 3. On the other hand, as observed by Cerrone and Russo (2023), we verify a drastic reduction on the number of vertices and arcs for the benchmark instances of Ferone *et al.* (2019).

Table 1 – Details about the instances of Groups 1, 2, and 3.

Group 1				Group 2				Group 3			
inst	$\|A\|$	k	opt	inst	$\|A\|$	k	opt	inst	$\|A\|$	k	opt
1	1420	6	9242	21	1430	7	6470	41	1704	7	7909
2	1420	7	6358	22	1447	7	6211	42	1722	7	8503
3	1420	7	5953	23	1439	7	5897	43	1692	7	7400
4	1420	7	5056	24	1435	7	5602	44	1698	7	7838
5	1420	7	6495	25	1444	7	7288	45	1720	7	7736
6	1420	7	5382	26	1431	7	6824	46	1672	8	6293
7	1420	7	6774	27	1434	7	6969	47	1676	8	4963
8	1420	7	6345	28	1434	7	6956	48	1692	8	5458
9	1420	7	4086	29	1430	7	5186	49	1690	8	4447
10	1420	7	7052	30	1445	7	4763	50	1700	8	5095
11	1420	7	6128	31	1445	7	6550	51	1652	7	6737
12	1420	7	6097	32	1435	7	7029	52	1692	8	4636
13	1420	7	5541	33	1438	7	5600	53	1688	8	5596
14	1420	7	4981	34	1442	7	6102	54	1686	8	4632
15	1420	7	6572	35	1440	7	7624	55	1680	8	5335
16	1420	7	5434	36	1441	7	5124	56	1680	8	5222
17	1420	7	7386	37	1449	7	6778	57	1660	8	4852
18	1420	7	5102	38	1438	7	5761	58	1704	8	4785
19	1420	7	5885	39	1435	7	5006	59	1662	8	6132
20	1420	7	5793	40	1440	7	6845	60	1716	8	5974

Source: the author.

Table 2 – Details about the benchmarks instances for random and grid digraphs (FERONE *et al.*, 2019).

Random							Grid						
inst	$\|V\|$	$\|A\|$	$\|R(V)\|$	$\|R(A)\|$	k	opt	inst	$\|V\|$	$\|A\|$	$\|R(V)\|$	$\|R(A)\|$	k	opt
R1-27190	75000	750000	16	16	8	242	G1-27000	10000	39600	627	1648	195	6131
R1-27191	75000	750000	35	39	6	201	G1-27001*	10000	39600	209	424	197	6233
R1-27195*	75000	750000	13	13	6	152	G1-27002	10000	39600	402	956	191	6336
R1-27197	75000	750000	12	12	6	253	G1-27003	10000	39600	423	922	196	6200
R1-27199	75000	750000	22321	99827	5	333	G1-27004	10000	39600	256	580	195	6375
R1-27200	75000	750000	120	147	6	236	G1-27005	10000	39600	248	578	197	6079
R1-27202	75000	750000	80	92	8	253	G1-27006	10000	39600	281	624	193	6109
R1-27203	75000	750000	17	20	5	255	G1-27007	10000	39600	220	462	198	6197
R1-27204	75000	750000	24808	119208	5	401	G1-27008	10000	39600	214	442	191	6193
R1-27205	75000	750000	33203	190490	6	426	G1-27009	10000	39600	247	540	196	6181
R2-27001	75000	750000	80	93	6	289	G3-27000	20000	79400	380	812	312	9808
R2-27004	75000	750000	27	29	6	246	G3-27001	20000	79400	514	1154	294	9786
R2-27005	75000	750000	23	25	6	198	G3-27002	20000	79400	489	1126	291	9652
R2-27007	75000	750000	33	38	6	231	G3-27003	20000	79400	323	672	305	9448
R2-27008	75000	750000	15	15	7	196	G3-27004	20000	79400	1145	2884	295	10149
R2-27010	75000	750000	148	180	5	246	G3-27005	20000	79400	437	1040	296	9793
R2-27012	75000	750000	26	28	7	245	G3-27006	20000	79400	688	1520	299	9654
R2-27015	75000	750000	56	65	6	238	G3-27007	20000	79400	319	664	298	9535
R2-27018	75000	750000	61	74	6	219	G3-27008	20000	79400	371	818	295	9455
-	-	-	-	-	-	-	G3-27009	20000	79400	2013	6654	296	9424

$*$ The shortest path solution is feasible for the k-CSPP.

Source: the author.

4.1 Heuristic results

We inform the CCDA results in Table 3 for random and grid digraphs (FERONE *et al.*, 2019) and the instances of Group 2. The heuristic fails to find feasible solutions for the instances of Groups 1 and 3. For the legend, in addition to the instance identifier *inst*, we have the heuristic solution cost *ub*, the execution time, in seconds, *cpu* (with a time limit of 10 seconds), and the number of colors in the path in the corresponding column.

The CCDA heuristic reaches the optimal solution values, reported in bold, for 29 out of 39 instances from the literature (FERONE *et al.*, 2019) while, for the instances of Group 2, CCDA does not reach any optimal solution value. This result shows that the heuristic, receiving as input the standard penalty list with the addition of the penalty $2 \times w_{\max}$, overall does not perform well on our new groups of instances. Indeed, we observe that even for the instances where the heuristic found feasible solutions, the found solutions are far from optimal. In addition, the solutions for instances from Group 2 are found in less than 0.1 seconds, probably due to the addition of jump arcs.

Table 3 – CCDA results for instances of Group 2 and from the literature (FERONE *et al.*, 2019).

Random				Grid				Group 2			
inst	ub	cpu	colors	inst	ub	cpu	colors	inst	ub	cpu	colors
R1-27190	**242**	1.6	6	G1-27000	6150	0.3	194	21	19579	0.4	3
R1-27191	**201**	2.1	6	G1-27001	**6233**	0.1	197	22	51145	0	4
R1-27195	**152**	1.1	6	G1-27002	**6336**	0.2	189	23	27388	0	5
R1-27197	**253**	1.2	5	G1-27003	6203	0.2	194	24	23087	0	6
R1-27199	**333**	5.1	5	G1-27004	**6375**	0.2	195	25	51876	0	5
R1-27200	**236**	5.1	6	G1-27005	**6079**	0.2	197	26	19473	0	4
R1-27202	261	4.4	7	G1-27006	6109	0.2	193	27	20394	0	6
R1-27203	**255**	4.4	5	G1-27007	**6197**	0.2	198	28	54286	0	4
R1-27204	**401**	5.5	5	G1-27008	**6193**	0.1	191	29	30249	0	6
R1-27205	**426**	6.1	6	G1-27009	6183	0.1	195	30	21169	0	4
R2-27001	**289**	5	6	G3-27000	**9808**	0.3	311	31	35305	0	5
R2-27004	**246**	2	6	G3-27001	9792	0.5	294	32	30744	0	4
R2-27005	**198**	2.6	5	G3-27002	9655	0.6	291	33	26077	0	3
R2-27007	**231**	3.1	6	G3-27003	**9448**	0.3	303	34	27487	0	3
R2-27008	**196**	2.6	6	G3-27004	062	0.5	295	35	22731	0	4
R2-27010	**246**	5	5	G3-27005	**9793**	0.5	296	36	18497	0	4
R2-27012	**245**	1.3	6	G3-27006	9661	0.3	293	37	29485	0	3
R2-27015	**238**	3.3	6	G3-27007	**9535**	0.2	298	38	27833	0	6
R2-27018	**219**	3.4	5	G3-27008	**9455**	0.3	295	39	21346	0	3
-	-	-	-	G3-27009	9521	0.5	289	40	27264	0	3

Source: the author.

4.2 Dynamic programming results

In this section, we report computational results for the dynamic programming (DP) algorithm proposed by Ferone *et al.* (2021) for reduced random and grid digraphs (FERONE *et al.*, 2019) and only for the new instances of Groups 1 and 2. This is because the DP algorithm fails to find the optimal solutions for all instances of Group 3 in the imposed time limit of 1800 seconds.

Our implementation of the DP algorithm uses the A^* extraction policy because it has the best results in the literature (FERONE *et al.*, 2021). To obtain the values $\pi(u,t)$ for all $u \in V$ we run Dijkstra's algorithm in the reverse network, starting from t. The reverse network of D is a digraph $D' = (V,A')$ for $A' = \{(v,u) \mid \forall\ (u,v) \in A\}$. The arcs in the reverse network have the same cost as the original arcs in D. The stopping criterion for the DP algorithm, in addition to the time limit, is to reach the known optimal solution for the instance.

As observed in (FERONE *et al.*, 2021), we note in Table 4 that, on average, the DP algorithm runs in negligible time for random and grid digraphs. These benchmark instances proved to be solved easily with the algorithm. On the other hand, regarding the new instances, we note that the average computational time is significantly higher than that one for instances of the literature. The DP algorithm presents an increase of 6.6% of cpu time for Group 2 in comparison

Table 4 – DP (FERONE *et al.*, 2021) results for reduced random and grid digraphs (FERONE *et al.*, 2019), and the instances of Groups 1 and 2.

Random		Grid		Group 1		Group 2	
instance	cpu	instance	cpu	instance	cpu	instance	cpu
R1-27190	0.3	G1-27000	2.8	1	85.9	21	856.6
R1-27191	0.0	G1-27001	0.0	2	1010.1	22	810.7
R1-27195	0.0	G1-27002	14.6	3	846.0	23	957.5
R1-27197	0.0	G1-27003	0.4	4	845.0	24	915.6
R1-27199	0.3	G1-27004	0.2	5	774.8	25	814.5
R1-27200	0.0	G1-27005	1.5	6	957.0	26	827.6
R1-27202	0.3	G1-27006	0.4	7	848.5	27	738.7
R1-27203	0.0	G1-27007	0.0	8	727.9	28	802.5
R1-27204	0.3	G1-27008	0.0	9	475.1	29	976.2
R1-27205	1.2	G1-27009	0.0	10	782.0	30	727.2
R2-27001	0.0	G3-27000	0.7	11	854.1	31	834.6
R2-27004	0.3	G3-27001	1.9	12	889.9	32	818.9
R2-27005	0.0	G3-27002	3.2	13	923.6	33	767.3
R2-27007	0.0	G3-27003	0.5	14	664.5	34	849.2
R2-27008	0.0	G3-27004	2.8	15	986.7	35	1118.9
R2-27010	0.0	G3-27005	8.0	16	822.6	36	729.1
R2-27012	0.0	G3-27006	1.2	17	837.3	37	1061.9
R2-27015	0.0	G3-27007	0.2	18	877.5	38	775.5
R2-27018	0.0	G3-27008	0.4	19	984.9	39	765.9
-	-	G3-27009	13.0	20	775.8	40	875.3
average	0.1	average	2.6	average	798.5	average	851.2
median	0.0	median	0.6	median	845.5	median	823.3
max	1.2	max	14.6	max	1010.1	max	1118.9
min	0.0	min	0.0	min	85.9	min	727.2

Source: the author.

with Group 1. Because the DP algorithm runs out of memory or reaches the time limit for all instances of Group 3, we do not report results for this group. As we noted for the heuristic, the results for the dynamic programming algorithm also seems to support our hypotheses. Indeed, our new groups of instances seems to be significantly more challenging.

4.3 Branch-and-Bound results

In this section, we report the results of the B&B algorithm (FERONE *et al.*, 2019). We adopt the BFS node evaluation policy in the search tree, since it presents the best numerical results (FERONE *et al.*, 2019).

We present numerical results for grid and random instances in Table 5. For each 'instance', we report the 'cpu' time and the number of generated nodes 'bb'. The algorithm did not find an optimal solution for 7 out of 19 random instances and for 11 out of 20 grid instances. We remark that the instance reduction algorithm helps to improve the number of proved optimal solutions compared to that in Ferone *et al.* (2019), where the B&B algorithm found the optimal

solution for 10 (resp. 8) random (resp. grid) instances.

Table 5 – B&B (FERONE *et al.*, 2019) results for reduced random and grid instances.

Random			Grid		
instance	cpu	bb	instance	cpu	bb
R1-27190	2.4	10	G1-27000	1860.4	68235745
R1-27191	1.3	19707	G1-27001	0.2	398
R1-27195	0.0	0	G1-27002	1803.2	66533932
R1-27197	0.0	8	G1-27003	23.3	122584
R1-27199	1800.0	69366	G1-27004	1375.2	11514256
R1-27200	1800.0	40298437	G1-27005	1800.0	60402669
R1-27202	105.8	1150279	G1-27006	1800.0	34953907
R1-27203	0.0	37	G1-27007	1.2	12000
R1-27204	1800.3	43324	G1-27008	0.0	193
R1-27205	1800.2	37596	G1-27009	1.5	13242
R2-27001	1812.9	34296891	G3-27000	1800.0	60344520
R2-27004	2.4	291	G3-27001	1800.0	58638557
R2-27005	0.0	408	G3-27002	1826.0	69363205
R2-27007	2.0	27827	G3-27003	0.5	2763
R2-27008	0.0	9	G3-27004	2029.3	65656475
R2-27010	1800.2	36995004	G3-27005	1804.2	70851721
R2-27012	0.3	81	G3-27006	13.4	47859
R2-27015	231.9	2347792	G3-27007	0.5	4200
R2-27018	1849.2	5402209	G3-27008	1800.0	37792525
-	-	-	G3-27009	1800.0	3658583
average	684.7	6352067.2	average	1076.9	30407466.7
median	2.4	27827.0	median	1800.0	23234081.5
max	1849.2	40298437.0	max	2029.3	70851721.0
min	0.0	0.0	min	0.0	193.0

Source: the author.

The numerical results in Table 5 indicate that the branch-and-bound algorithm, with a breadth-first search node evaluation policy, is not as efficient for our instances in comparison to the results reported for the instances of the literature. Indeed, the B&B algorithm was not able to find an optimal solution in the time limit of 1800 seconds for all the instances of Groups 1, 2, and 3.

4.4 Branch-and-Cut results for model (PCM)

In this section, we discuss results of a branch-and-cut (B&C) algorithm for model (PCM). Initially, we solve the (s,t)-shortest path problem for the instance. If the solution does not violate the limit k on the number of colors, then the path is optimal to the k-CSPP. Otherwise, we add the corresponding violated cut (3.1) to model (PCM) and solve it with the B&C module of the IBM CPLEX solver. We use lazy callbacks to cut paths violating inequalities (3.1). The stopping criterion is either to reach a time limit of 1800 seconds, or to obtain the optimal k-CSPP

solution.

Table 6 reports B&C results with model (PCM) for random and grid digraphs (FERONE *et al.*, 2019). The additional legend is the 'cpu' time (in seconds), the number of generated 'cuts' (3.1), and the number of CPLEX B&C nodes 'bc'.

Excepting for three instances, the remaining ones were solved to optimality. Three random instances (R1-27199, R1-27204, and R1-27205) reach the time limit of 1800 seconds (they do not benefit much of the reduction algorithm) with a CPLEX lower-bound far from their optimal solution values of 39.62%, 34.08%, and 40.97%, respectively. For the remaining 16 random instances, the model runs with an average cpu time of 1.4 seconds with an average number of cuts (3.1) of 9.4 cuts. For grid digraphs, the B&C approach solved all of them to optimality. They present average values of cpu time, cuts (3.1), and CPLEX B&C nodes, of 4.8 seconds, 369.6 cuts, and 885.5 B&C nodes, respectively. We remark that 9 instances (8 random and 1 grid) were solved at the root node of the B&C tree with no generation of cuts (3.1). The last two lines in this table refer to the average and median values for the corresponding columns, respectively.

Table 6 – B&C results with model (PCM) for random and grid benchmark instances (FERONE *et al.*, 2019).

Random				Grid			
inst	cpu	cuts	bc	inst	cpu	cuts	bc
R1-27190	1.0	1	0	G1-27000	26.5	2296	2803
R1-27191	1.5	10	4	G1-27001	0.4	10	3
R1-27195	0.1	0	0	G1-27002	7.3	674	2170
R1-27197	0.1	1	0	G1-27003	0.6	120	262
R1-27199	1800.0	2439	6508	G1-27004	1.3	150	297
R1-27200	3.7	30	39	G1-27005	2.3	401	665
R1-27202	1.9	12	12	G1-27006	0.8	174	327
R1-27203	0.2	2	0	G1-27007	0.1	9	8
R1-27204	1800.0	1759	4737	G1-27008	0.0	4	0
R1-27205	1800.0	2008	3730	G1-27009	0.1	15	35
R2-27001	3.7	13	6	G3-27000	0.7	121	166
R2-27004	0.3	2	0	G3-27001	12.5	633	1684
R2-27005	1.1	8	0	G3-27002	12.7	1211	3401
R2-27007	0.7	6	8	G3-27003	0.3	21	23
R2-27008	0.2	1	0	G3-27004	2.2	61	182
R2-27010	3.2	32	48	G3-27005	21.0	1282	5138
R2-27012	0.3	2	0	G3-27006	0.2	11	21
R2-27015	2.2	13	11	G3-27007	1.4	8	10
R2-27018	2.4	17	22	G3-27008	2.0	92	183
				G3-27009	4.0	98	332
Average	285.4	334.5	796.1	Average	4.8	369.6	885.5
Median	1.5	10	6	Median	1.3	109.0	222.5

Source: the author.

Concerning the new instances of Groups 1, 2, and 3, solving them with the B&C

approach for model (PCM) was not possible. The Ubuntu operating system aborted all executions of the CPLEX solver for these groups of instances due to memory overflow.

4.5 Results for model (FFP) with valid inequalities

In this section, we discuss the impact of the valid inequalities from Chapter 3 for model (FFP) and the validity of our hypothesis. As a reminder, we want to observe if the valid inequalities strengthen the model proposed by Ferone *et al.* (2019) and if the new instances are more challenging w.r.t the ones proposed in the literature. In the following, to verify whether $u \in \bar{R}(v)$ for a given pair of vertices u and v, we construct a $|V| \times |V|$ binary matrix T, where $T_{u,v} = 1$ if u is reached by v, by running a breadth-first search (BFS) starting from every vertex of the digraph. We generate inequalities (3.9) by starting with a unitary set $S = \{v\}$ of pairwise non-reachable vertices, one for every $v \in V \setminus \{s,t\}$. For all $u \in V \setminus \{s,t,v\}$ having a label larger than v, we add u to S if u is not reached by as well as does not reach any vertex already in S. Then, we obtain one inequality (3.9) for every subset $Q \subseteq S$ with $|Q| \in \{0,1,|S|-1,|S|\}$.

We generate inequalities (3.12) with the use of a CPLEX user cut callback, based on the support graph associated with the non-null arc variables x w.r.t. a CPLEX B&B node solution. We obtain an inequality (3.12) for every pair of arcs (u,w) and (v,r), of the same color h, of any B&B node solution violating the corresponding inequality w.r.t. this pair of arcs. In particular, for such a pair of arcs, if for all $i \in N^+(w)$ we observe that $v \notin \bar{R}(i)$, we discard the corresponding inequality (3.12) because we can show that it is weaker than a 'modified combination' of constraints (2.2) for node w.

To obtain inequalities (3.14), we first describe Karger's randomized algorithm for computing the minimum cut (MOTWANI; RAGHAVAN, 1996). The idea behind the algorithm is to contract arcs randomly chosen until a unique arc remains. The label of the resulting vertex of an arc contraction is the union of the set of labels of the arc extremities. Duplicated arcs or in both directions between a given pair of nodes are considered only one arc independently of its direction. The algorithm returns the partition of the vertices V_1 and V_2 given by the extremities of the unique remaining arc. Given such partition, we check whether the cut of arcs $[V_2,V_1]$ is empty. If so, we add the corresponding inequality (3.14) to model (FFP). Otherwise, while $[V_2,V_1] \neq \emptyset$, for every arc (u,v) in this cut, we move u from V_2 to V_1. The resulting partition is then used to generate that inequality. We generate an arbitrary number of 60 inequalities of this type for every instance by running the algorithm this number of times. Finally, we add the

polynomial sets of inequalities from Chapter 3 and constraint (3.14) directly to model (FFP).

In Tables 7 and 8, we report the results of the valid inequalities for model (FFP) on random and grid digraphs, respectively. The additional legend shows the 'inequalities' being evaluated, the optimal linear relaxation 'r', the computational time to solve the linear relaxation 'cpu_r', and the 'cpu' time to solve the ILP model. For models that implements user cut callbacks, we also report the number of 'cuts'.

Except for a single instance, the use of inequalities did not improve on the linear relaxation or computational time. In fact, we observe that instance R1-27205 is the hardest, being solved to optimality in 47 seconds by model (FFP)+(3.6)–(3.8), an increase of 11.9 seconds in comparison to model (FFP)+(3.11)(3.12)(3.14). The cpu time to solve the instances is negligible for any other random instance. The same is true for grid digraphs, except for G1-27000, which is the only instance taking more than a second to be solved by any model. This shows how much easier they get after running the graph reduction algorithm.

Tables 9, 10, and 11 show the numerical results for Groups 1, 2, and 3, respectively. Observe that the new instances requires more time to obtain the optimal integer solution in comparison to the benchmark instances. Furthermore, the linear relaxation is still far from optimal, even with the addition of the valid inequalities. We also note that the addition of inequalities (3.4) and (3.5) can sometimes increase the required cpu time to solve an instance.

Table 7 – The impact of the valid inequalities for model (FFP) with the random digraphs (FERONE *et al.*, 2019).

		instances																		
inequalities	solution	R1-27190	R1-27191	R1-27195	R1-27197	R1-27199	R1-27200	R1-27202	R1-27203	R1-27204	R1-27205	R2-27001	R2-27004	R2-27005	R2-27007	R2-27008	R2-27010	R2-27012	R2-27015	R2-27018
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
(FFP)	cpu_r	0.2	0.1	0.1	0.1	6.4	0.1	0.1	0.1	7.0	39.4	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	cpu	0.2	0.1	0.1	0.1	6.4	0.1	0.1	0.1	7.0	39.4	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.14)	cpu_r	0.1	0.1	0.1	0.1	6.7	0.1	0.1	0.1	8.1	39.6	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	cpu	0.1	0.1	0.1	0.1	6.7	0.1	0.1	0.1	8.1	39.6	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.12)	cpu_r	0.1	0.0	0.0	0.0	4.4	0.0	0.0	0.0	5.2	37.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	0.1	0.1	0.1	0.1	6.5	0.1	0.1	0.1	8.2	39.5	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.11)	cpu_r	0.1	0.1	0.1	0.1	6.0	0.1	0.1	0.1	8.0	39.2	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	cpu	0.1	0.1	0.1	0.1	6.0	0.1	0.1	0.1	8.0	39.2	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.10)	cpu_r	0.1	0.1	0.1	0.1	6.4	0.1	0.1	0.1	8.1	39.3	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	cpu	0.1	0.1	0.1	0.1	6.4	0.1	0.1	0.1	8.1	39.3	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.9)	cpu_r	0.1	0.1	0.1	0.1	5.8	0.1	0.1	0.1	9.0	37.4	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	cpu	0.1	0.1	0.1	0.1	5.8	0.1	0.1	0.1	9.0	37.4	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.6)–(3.8)	cpu_r	0.1	0.1	0.1	0.1	6.4	0.1	0.1	0.1	8.1	47.0	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	cpu	0.1	0.1	0.1	0.1	6.4	0.1	0.1	0.1	8.1	47.0	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.4)–(3.5)	cpu_r	0.1	0.1	0.1	0.1	5.9	0.1	0.1	0.1	7.9	40.6	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	cpu	0.1	0.1	0.1	0.1	5.9	0.1	0.1	0.1	7.9	40.6	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.9),(3.11)–(3.12)	cpu_r	0.0	0.0	0.0	0.0	5.1	0.0	0.0	0.0	5.1	37.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	0.1	0.1	0.1	0.1	6.4	0.1	0.1	0.1	9.0	39.7	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.11)–(3.12), (3.14)	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	0.1	0.1	0.1	0.1	6.2	0.1	0.1	0.1	7.9	35.1	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2
	r	240.5	201.0	152.0	252.0	333.0	236.0	238.5	255.0	401.0	426.0	289.0	246.0	192.7	231.0	190.0	246.0	244.0	238.0	205.0
+(3.9),(3.11)–(3.12),(3.14)	cpu_r	0.0	0.0	0.0	0.0	5.2	0.0	0.0	0.0	5.2	37.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	0.1	0.1	0.1	0.1	5.8	0.1	0.1	0.1	9.0	39.4	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2

Source: the author.

Table 8 – The impact of the valid inequalities for model (FFP) with the grid digraphs (FERONE *et al.*, 2019).

		instances																			
inequalities	solution	G1-27000	G1-27001	G1-27002	G1-27003	G1-27004	G1-27005	G1-27006	G1-27007	G1-27008	G1-27009	G3-27000	G3-27001	G3-27002	G3-27003	G3-27004	G3-27005	G3-27006	G3-27007	G3-27008	G3-27009
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
(FFP)	cpu_r	0.4	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.4
	cpu	0.4	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.4
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.14)	cpu_r	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	cpu	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.12)	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.2
	cpu	1.5	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.11)	cpu_r	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.4
	cpu	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.4
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.10)	cpu_r	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	cpu	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.9)	cpu_r	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	cpu	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.6)–(3.8)	cpu_r	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.4
	cpu	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.4
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.4)–(3.5)	cpu_r	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	cpu	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.9),(3.11)–(3.12)	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.2
	cpu	0.3	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.11)–(3.12), (3.14)	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	1.5	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.4
	r	6120.0	6233.0	6327.0	6195.5	6375.0	6077.3	6106.3	6197.0	6193.0	6181.0	9805.7	9783.0	9651.7	9446.5	10147.4	9793.0	9653.3	9535.0	9453.5	9424.0
+(3.9),(3.11)–(3.12),(3.14)	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.2
	cpu	1.4	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.5

Source: the author.

Table 9 – The impact of the valid inequalities for model (FFP) with the instances of Group 1.

inequalities	solution	instances																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
(FFP)	z	3181.8	2701.4	3049.4	2761.4	3147.2	2749.8	3016.0	2399.3	2704.1	2879.0	2713.9	2752.5	2852.8	2788.1	2933.4	2525.7	2794.0	2401.3	2314.4	2692.5
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	108.9	502.6	615.4	239.5	512.6	191.9	1213.0	493.2	9.3	1566.5	410.7	959.7	264.8	224.6	629.8	160.8	1719.6	680.5	971.8	524.4
	bb	2599.0	19084.0	34839.0	7775.0	17292.0	7968.0	67975.0	17731.0	363.0	76247.0	17214.0	55712.0	7675.0	5795.0	39503.0	5343.0	83716.0	39699.0	57550.0	27468.0
+(3.14)	z	3652.8	2963.0	3434.6	3109.0	3466.4	3139.8	3273.0	2733.3	2966.1	3161.9	3131.4	3029.3	3081.0	3054.5	3232.5	3130.5	3188.1	2590.5	2597.4	3042.5
	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	20.2	121.0	128.8	13.0	153.3	140.7	69.9	179.4	12.3	224.4	44.8	153.3	138.3	20.2	138.1	21.5	391.6	22.1	186.1	34.0
	bb	449.0	5013.0	4975.0	695.0	4402.0	5864.0	3083.0	6392.0	363.0	8642.0	2352.0	5372.0	4298.0	868.0	4945.0	1056.0	17390.0	1455.0	6648.0	1903.0
+(3.12)	z	3897.6	3043.1	3501.3	3173.4	3556.6	3215.1	3482.5	2864.4	2988.0	3385.3	3273.2	3162.8	3208.4	3192.8	3393.2	3049.1	3420.2	2642.9	2776.7	3158.6
	cpu_r	0.2	0.1	0.1	0.2	0.2	0.1	0.1	0.2	0.1	0.2	0.2	0.2	0.2	0.2	0.1	0.1	0.2	0.1	0.2	0.1
	cpu	33.6	87.3	41.2	44.9	341.3	33.3	114.8	133.3	6.1	635.3	111.3	72.9	86.9	49.2	97.2	22.7	191.9	73.9	88.6	35.8
	bb	249.0	1667.0	990.0	900.0	3639.0	736.0	1970.0	1995.0	53.0	11730.0	2078.0	1178.0	1170.0	540.0	1565.0	469.0	3017.0	1171.0	1302.0	653.0
	cuts	4795.0	15594.0	10697.0	10160.0	48039.0	8202.0	22986.0	21202.0	1278.0	116210.0	27665.0	12567.0	14465.0	8162.0	14924.0	7235.0	28670.0	13239.0	14900.0	7739.0
+(3.11)	z	3510.8	2929.7	3218.9	3044.1	3365.9	3048.7	3184.6	2579.6	2894.9	3032.6	3079.3	2951.7	3007.0	2959.9	3072.9	2964.2	3071.1	2517.9	2449.6	2943.1
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	40.0	369.7	190.9	16.0	277.1	145.5	259.2	254.5	3.8	520.0	179.7	161.3	149.1	19.1	253.1	27.4	786.4	43.1	384.2	384.3
	bb	1079.0	22911.0	7655.0	1000.0	15311.0	6089.0	11656.0	10355.0	130.0	28082.0	7229.0	6649.0	5776.0	921.0	10092.0	1506.0	37225.0	2585.0	22564.0	17139.0
+(3.10)	z	3479.7	2891.2	3186.8	3023.2	3344.6	3002.2	3156.7	2549.1	2852.1	3009.2	3026.4	2932.1	2975.8	2933.7	3005.1	2868.2	3025.8	2489.7	2444.7	2915.5
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	50.4	233.4	183.8	162.9	309.5	366.7	484.7	477.0	5.2	588.5	340.3	46.3	205.6	28.4	316.3	26.2	887.3	141.6	606.8	441.7
	bb	1299.0	9081.0	7435.0	4199.0	13405.0	15133.0	21601.0	14270.0	144.0	28114.0	21641.0	2662.0	4542.0	1279.0	10762.0	1393.0	36441.0	5587.0	25850.0	19644.0
+(3.9)	z	3661.8	2963.0	3468.5	3116.2	3466.8	3139.8	3274.0	2733.8	2973.7	3169.3	3131.4	3038.3	3090.7	3071.0	3238.7	3153.6	3193.7	2596.5	2610.9	3045.8
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1
	cpu	22.4	183.9	128.6	27.5	133.9	49.8	185.0	157.5	13.6	434.5	52.4	191.5	39.3	107.5	152.5	24.3	286.3	20.1	44.8	20.5
	bb	485.0	4812.0	4614.0	1055.0	4877.0	2112.0	6890.0	5654.0	306.0	18477.0	2564.0	6571.0	1695.0	3497.0	6119.0	1146.0	9623.0	1433.0	2112.0	1116.0
+(3.6)–(3.8)	z	3652.7	2960.8	3456.4	3108.1	3465.0	3139.8	3273.0	2733.3	2962.3	3168.0	3122.0	3038.3	3081.0	3054.5	3205.7	3122.1	3183.3	2593.2	2610.9	3042.2
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	21.3	170.6	127.9	26.3	162.1	131.5	233.9	190.3	10.9	231.6	188.2	197.3	35.2	29.3	169.1	17.6	619.3	30.1	176.2	319.4
	bb	511.0	4919.0	4728.0	953.0	4962.0	4212.0	6118.0	5805.0	268.0	11678.0	5927.0	7312.0	1927.0	1147.0	6021.0	762.0	19951.0	1404.0	5320.0	10364.0
+(3.4)–(3.5)	z	3188.7	2706.1	3167.4	2801.1	3158.4	2757.2	3016.1	2422.0	2706.9	2901.4	2747.3	2804.8	2863.8	2789.7	2965.4	2558.4	2807.9	2416.4	2340.0	2692.5
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	104.7	533.0	521.7	46.5	291.0	361.9	1738.5	876.4	24.1	1341.9	370.2	335.6	164.0	267.0	499.0	228.1	1322.3	157.9	833.5	821.1
	bb	2299.0	19407.0	13979.0	2593.0	11079.0	19142.0	70285.0	54447.0	941.0	69503.0	13915.0	14399.0	6531.0	6988.0	18168.0	7273.0	70962.0	7594.0	45182.0	36932.0
+(3.9),(3.11)–(3.12)	z	3965.2	3113.9	3594.3	3319.1	3600.5	3227.3	3537.3	2956.5	3069.7	3471.0	3333.9	3314.7	3250.0	3234.7	3461.4	3272.0	3454.6	2693.0	2861.6	3220.6
	cpu_r	0.2	0.2	0.2	0.2	0.2	0.1	0.2	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.1	0.2	0.2
	cpu	19.5	60.5	139.4	30.4	77.6	21.8	83.6	69.2	3.6	411.7	54.7	61.9	19.7	32.9	80.6	30.9	130.2	20.7	32.6	62.7
	bb	181.0	1538.0	2297.0	731.0	1781.0	618.0	1968.0	1708.0	68.0	10203.0	1251.0	1449.0	531.0	575.0	1856.0	643.0	2522.0	796.0	903.0	1211.0
	cuts	2838.0	10682.0	18062.0	7369.0	15867.0	5133.0	17615.0	13111.0	816.0	78225.0	11293.0	14313.0	4839.0	5860.0	15539.0	6808.0	20944.0	6620.0	7379.0	10827.0
+(3.11)-+(3.12), (3.14)	z	3957.2	3112.9	3555.2	3311.4	3599.5	3227.3	3537.0	2956.5	3069.5	3450.6	3330.9	3302.4	3239.9	3218.3	3440.6	3270.0	3447.3	2686.9	2834.2	3218.4
	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	20.5	65.6	56.6	18.3	50.9	23.8	52.4	64.9	2.7	84.1	58.5	35.1	24.8	15.9	72.5	33.6	137.5	23.3	51.9	22.2
	bb	205.0	1616.0	1535.0	495.0	1212.0	748.0	1471.0	1602.0	43.0	2247.0	1312.0	922.0	621.0	329.0	1747.0	766.0	2825.0	762.0	1288.0	598.0
	cuts	3211.0	11531.0	11621.0	4434.0	10363.0	5852.0	11884.0	13668.0	607.0	14433.0	12940.0	7339.0	5668.0	3597.0	14044.0	9413.0	23592.0	6323.0	10380.0	4865.0
+(3.9),(3.11)–(3.12),(3.14)	z	3965.2	3113.9	3594.3	3319.1	3600.5	3227.3	3537.3	2956.5	3069.7	3471.0	3333.9	3314.7	3250.0	3234.7	3461.4	3272.0	3454.6	2693.0	2861.6	3220.6
	cpu_r	0.3	0.5	0.2	0.3	0.5	0.2	0.6	0.5	0.2	0.6	0.5	0.4	0.3	0.2	0.5	0.2	0.5	0.2	0.6	0.3
	cpu	26.2	112.5	40.9	23.1	106.0	31.9	86.5	115.9	2.9	422.4	143.2	78.7	42.8	23.8	147.7	24.0	179.2	36.9	53.7	49.3
	bb	185.0	1788.0	866.0	370.0	1550.0	636.0	1485.0	1773.0	45.0	5329.0	1692.0	1200.0	735.0	366.0	2009.0	385.0	2425.0	755.0	935.0	706.0
	cuts	4067.0	20324.0	11385.0	6436.0	21937.0	9624.0	19232.0	23681.0	920.0	65910.0	26121.0	17579.0	11420.0	6310.0	29970.0	6195.0	32756.0	11446.0	13375.0	10286.0

Source: the author.

Table 10 – The impact of the valid inequalities for model (FFP) with the instances of Group 2.

inequalities	solution	instances																				
		21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	
(FFP)	z	3130.8	3186.2	2981.9	2682.3	3310.5	3306.3	2684.3	3450.0	2656.1	2723.8	2789.5	2822.4	2831.1	3136.3	2801.5	3074.1	2665.6	2722.1	2729.8	3082.6	
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
	cpu	461.4	274.4	191.3	274.8	607.5	708.6	846.6	443.2	254.6	343.3	390.0	725.8	230.4	507.0	1614.7	126.3	672.1	280.6	361.8	511.3	
	bb	21047.0	10130.0	7842.0	10978.0	21863.0	25157.0	30056.0	19156.0	12391.0	12640.0	17215.0	23159.0	9226.0	18924.0	44310.0	4469.0	27393.0	10629.0	15054.0	20263.0	
+(3.14)	z	3374.4	3466.1	3277.3	3048.3	3522.6	3701.8	3042.6	3802.7	2978.3	3096.8	3085.7	3272.7	3140.6	3475.1	3058.2	3316.3	2880.4	3001.9	3059.8	3623.2	
	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
	cpu	176.0	22.9	38.6	31.0	231.0	136.8	177.6	41.5	29.6	91.4	141.7	164.7	22.5	128.4	351.1	11.8	211.1	30.3	118.2	52.4	
	bb	6214.0	1216.0	2113.0	1699.0	7423.0	4832.0	7552.0	2205.0	1760.0	2728.0	5484.0	6599.0	1332.0	4557.0	13303.0	660.0	9245.0	1665.0	3926.0	2758.0	
+(3.12)	z	3652.5	3628.7	3389.5	3201.8	3696.1	3692.4	3041.6	3823.2	3121.0	3255.8	3146.1	3357.4	3300.8	3578.4	3239.8	3411.6	3035.6	3027.6	3011.4	3697.9	
	cpu_r	0.2	0.2	0.1	0.1	0.2	0.2	0.1	0.2	0.1	0.1	0.2	0.2	0.1	0.2	0.3	0.1	0.2	0.1	0.1	0.2	
	cpu	94.0	74.8	51.4	64.9	448.4	137.8	148.7	100.0	29.5	24.1	115.9	539.9	67.4	388.2	580.4	77.5	458.3	81.3	19.8	139.9	
	bb	1891.0	1164.0	1100.0	1349.0	5224.0	2442.0	2640.0	1844.0	702.0	345.0	2210.0	7791.0	1376.0	5563.0	8797.0	921.0	6493.0	1469.0	478.0	2297.0	
	cuts	15747.0	16193.0	11245.0	16474.0	75547.0	26575.0	28691.0	18865.0	7086.0	6757.0	21411.0	115740.0	14689.0	87774.0	111130.0	17208.0	90530.0	17809.0	5991.0	28138.0	
+(3.11)	z	3311.3	3417.6	3215.5	2974.7	3440.8	3539.8	2921.5	3703.4	2867.7	3039.3	2984.4	3097.7	3043.5	3364.2	2935.6	3217.8	2765.8	2840.4	3004.0	3430.9	
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
	cpu	240.1	160.4	144.5	206.1	236.8	298.3	302.8	185.5	29.7	110.2	263.3	287.5	26.2	172.5	462.3	125.7	294.1	130.5	100.3	227.5	
	bb	8921.0	5655.0	6166.0	6478.0	10138.0	11052.0	11931.0	7766.0	1718.0	3082.0	9563.0	12406.0	1611.0	6872.0	16921.0	3881.0	13183.0	4753.0	4163.0	9066.0	
+(3.10)	z	3254.5	3379.9	3186.6	2965.1	3428.5	3523.5	2917.0	3664.0	2843.6	3005.0	2938.5	3070.0	3016.7	3307.4	2896.1	3192.0	2764.9	2839.3	2881.3	3366.3	
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
	cpu	229.9	45.0	163.6	179.1	341.2	252.8	307.7	182.7	126.0	107.0	249.3	376.3	113.3	364.3	559.8	32.5	406.6	154.3	17.9	248.1	
	bb	9127.0	2266.0	6287.0	5935.0	13005.0	10899.0	12692.0	7321.0	4549.0	3312.0	10347.0	16316.0	3824.0	12543.0	20859.0	1386.0	15807.0	5875.0	1043.0	10330.0	
+(3.9)	z	3374.5	3468.9	3277.3	3048.3	3526.8	3704.6	3042.6	3803.1	2978.3	3096.8	3085.9	3280.1	3165.7	3475.1	3058.2	3316.3	2880.4	3001.9	3079.7	3632.5	
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
	cpu	163.6	20.7	107.5	27.7	318.5	50.0	197.3	58.9	24.4	14.4	177.5	215.8	15.8	42.2	259.7	11.2	206.4	114.1	120.7	144.4	
	bb	6633.0	1079.0	4242.0	1449.0	9565.0	2641.0	7998.0	3044.0	1447.0	598.0	6445.0	9332.0	899.0	2379.0	11072.0	613.0	9104.0	3804.0	4866.0	5939.0	
+(3.6)–(3.8)	z	3374.4	3458.7	3273.2	3048.3	3522.6	3693.5	3042.5	3800.1	2973.6	3086.7	3082.7	3268.5	3140.6	3475.0	3045.0	3316.0	2880.4	3001.9	3049.8	3578.3	
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
	cpu	231.9	29.3	31.5	109.9	190.4	55.4	214.0	152.4	21.0	113.2	159.7	241.9	90.4	124.1	283.5	110.6	193.7	27.4	112.4	165.4	
	bb	8205.0	1470.0	1849.0	3848.0	6995.0	2963.0	8168.0	6059.0	1351.0	3495.0	6412.0	9636.0	2866.0	4547.0	11790.0	3500.0	8567.0	1581.0	3563.0	7565.0	
+(3.4)–(3.5)	z	3141.8	3186.5	2992.8	2738.0	3311.9	3327.0	2684.6	3470.9	2663.5	2737.4	2804.6	2846.9	2856.7	3181.1	2802.3	3087.1	2667.6	2741.3	2801.6	3117.7	
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
	cpu	739.7	274.3	177.0	355.9	531.1	464.3	710.6	328.4	163.6	90.5	378.5	989.2	161.2	305.3	1038.5	192.4	671.7	236.2	231.9	599.2	
	bb	23188.0	10027.0	7971.0	14994.0	19612.0	19123.0	26272.0	13691.0	6400.0	2725.0	16746.0	29681.0	7314.0	13255.0	34393.0	5699.0	27376.0	9591.0	9347.0	20646.0	
+(3.9),(3.11)–(3.12)	z	3688.8	3715.1	3452.5	3260.8	3697.7	3793.5	3144.8	3934.8	3154.8	3310.6	3190.2	3473.1	3397.2	3653.9	3280.4	3491.5	3055.9	3082.5	3206.9	3823.5	
	cpu_r	0.2	0.2	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	
	cpu	107.7	18.5	36.7	74.0	105.9	87.4	86.9	267.7	28.3	19.6	63.6	99.9	16.8	53.0	306.1	10.6	130.7	25.9	32.9	95.0	
	bb	2193.0	490.0	1017.0	1271.0	2116.0	1924.0	2095.0	5089.0	934.0	355.0	1583.0	2001.0	477.0	1055.0	5042.0	190.0	2754.0	789.0	723.0	1867.0	
	cuts	16256.0	4406.0	8281.0	15491.0	19664.0	16241.0	17946.0	52088.0	6626.0	5258.0	11578.0	18369.0	3787.0	9971.0	47020.0	2937.0	25441.0	6245.0	7402.0	19780.0	
+(3.11)–(3.12), (3.14)	z	3688.6	3711.1	3449.6	3260.8	3696.1	3791.1	3144.6	3934.8	3154.8	3310.6	3190.2	3451.9	3364.0	3653.9	3280.4	3491.5	3054.1	3082.5	3202.8	3798.7	
	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
	cpu	283.5	19.4	50.8	68.2	88.7	48.2	93.4	47.8	29.4	26.3	88.8	99.4	13.4	92.6	172.3	34.6	100.5	47.3	18.2	106.0	
	bb	5353.0	500.0	1280.0	1222.0	1956.0	1266.0	2204.0	1261.0	817.0	418.0	2067.0	2428.0	423.0	1611.0	3382.0	594.0	2250.0	1098.0	463.0	2017.0	
	cuts	44906.0	4483.0	11006.0	14852.0	17247.0	9689.0	17572.0	9880.0	6314.0	6459.0	15898.0	22870.0	3391.0	17021.0	26476.0	9708.0	19679.0	10079.0	4267.0	21331.0	
+(3.9),(3.11)–(3.12),(3.14)	z	3688.8	3715.1	3452.5	3260.8	3697.7	3793.5	3144.8	3934.8	3154.8	3310.6	3190.2	3473.1	3397.2	3653.9	3280.4	3491.5	3055.9	3082.5	3206.9	3823.5	
	cpu_r	0.6	0.3	0.3	0.5	0.6	0.5	0.6	0.5	0.2	0.2	0.6	0.6	0.2	0.5	0.6	0.3	0.6	0.4	0.5	0.6	
	cpu	119.3	30.0	48.9	127.9	317.4	79.1	139.9	71.4	39.7	26.3	113.1	100.7	35.6	97.2	280.8	56.1	399.9	66.3	69.6	112.6	
	bb	1763.0	543.0	956.0	1436.0	4756.0	1400.0	1987.0	1234.0	843.0	371.0	1842.0	1618.0	588.0	1373.0	3295.0	674.0	5295.0	1008.0	974.0	1695.0	
	cuts	21932.0	6806.0	12105.0	27127.0	75211.0	16887.0	26657.0	14973.0	11510.0	9087.0	23044.0	22345.0	9212.0	22260.0	41546.0	15686.0	79845.0	14838.0	17705.0	25649.0	

Source: the author.

Table 11 – The impact of the valid inequalities for model (FFP) with the instances of Group 3.

inequalities	solution	instances																			
		41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
(FFP)	z	3112.0	3145.9	3526.6	2871.4	3611.3	3196.4	2870.1	3104.1	2423.5	3052.2	3392.0	2687.6	2616.2	2858.3	3056.3	2543.5	2581.1	2892.3	3226.2	3001.0
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	694.8	1126.5	1800.0	1558.5	727.1	1800.0	219.8	266.5	1004.7	412.5	249.3	349.4	1771.9	215.4	1073.0	701.3	191.3	927.4	1625.6	569.5
	bb	17709.0	50871.0	45082.0	60863.0	33727.0	120174.0	9518.0	10764.0	65201.0	13673.0	7260.0	12292.0	143871.0	5414.0	74474.0	45047.0	10802.0	47153.0	57894.0	17794.0
+(3.14)	z	3269.8	3331.0	3619.2	2962.1	3720.1	3300.0	3015.5	3256.3	2485.8	3167.2	3673.8	2843.5	2769.8	2970.7	3128.1	2706.5	2733.4	2988.2	3342.8	3018.5
	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	275.1	382.4	1524.3	599.3	1800.0	845.0	244.2	187.5	29.8	203.3	66.3	152.6	250.3	29.8	226.1	208.3	169.3	186.7	534.1	363.9
	bb	7087.0	10172.0	29539.0	32957.0	24491.0	24612.0	8169.0	7169.0	1576.0	7358.0	2629.0	6542.0	11491.0	1114.0	8729.0	8516.0	6833.0	5127.0	23847.0	12585.0
+(3.12)	z	3497.0	3710.6	3676.4	3162.8	3850.2	3455.0	3052.2	3305.1	2741.3	3383.0	3738.7	2936.3	2936.8	3049.8	3171.4	2871.9	2749.9	3111.3	3490.9	3130.0
	cpu_r	0.2	0.2	0.3	0.1	0.2	0.2	0.1	0.2	0.2	0.1	0.1	0.1	0.2	0.1	0.2	0.1	0.1	0.1	0.1	0.2
	cpu	885.6	456.9	1193.0	380.3	810.5	1047.8	80.8	496.9	31.9	87.3	92.7	274.2	457.4	34.3	768.8	305.2	77.1	43.5	526.5	656.9
	bb	14650.0	6858.0	17522.0	5741.0	13986.0	15214.0	1836.0	6584.0	602.0	2247.0	1549.0	4827.0	6903.0	604.0	23472.0	8089.0	2150.0	895.0	10653.0	24230.0
	cuts	87805.0	26981.0	123450.0	20609.0	72424.0	48648.0	4050.0	26563.0	1552.0	8269.0	6530.0	12619.0	22879.0	2941.0	109862.0	18213.0	4662.0	3480.0	26608.0	76548.0
+(3.11)	z	3130.7	3233.0	3575.5	2882.4	3614.0	3218.0	2949.8	3168.6	2462.8	3066.6	3479.9	2704.8	2649.4	2874.9	3084.2	2557.6	2649.4	2900.8	3255.2	3001.5
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	967.4	1292.9	1167.5	1800.0	1289.3	1800.0	142.9	525.0	130.1	242.3	197.8	290.3	813.4	237.2	496.9	340.8	298.7	371.6	792.4	354.6
	bb	39926.0	52432.0	41920.0	49142.0	48790.0	119345.0	5046.0	29150.0	4888.0	10423.0	6715.0	12301.0	29579.0	7475.0	14612.0	15034.0	12118.0	11554.0	79267.0	15453.0
+(3.10)	z	3157.3	3249.3	3575.5	2894.4	3611.5	3200.8	2870.1	3180.7	2451.6	3070.1	3497.8	2688.0	2635.4	2879.9	3084.2	2554.5	2600.0	2900.0	3274.4	3001.0
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	1488.1	1254.2	1113.4	1800.0	1099.6	1321.3	213.1	253.6	122.4	269.7	279.6	252.4	789.0	43.0	539.9	462.8	214.7	404.1	1241.1	1681.4
	bb	55122.0	55209.0	43012.0	43204.0	47607.0	96525.0	4775.0	8573.0	4427.0	8790.0	7420.0	7997.0	69426.0	1870.0	19547.0	19642.0	7954.0	14207.0	46457.0	101368.0
+(3.9)	z	3157.3	3254.5	3599.1	2898.7	3614.0	3234.9	3009.9	3235.3	2466.4	3080.9	3612.9	2758.5	2668.2	2926.1	3086.9	2566.5	2659.7	2967.8	3319.6	3037.3
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	776.1	1044.6	1203.1	780.4	1192.7	1338.8	283.6	262.2	43.2	157.3	236.4	395.5	481.0	25.4	658.9	290.3	189.7	537.2	998.5	1800.0
	bb	39711.0	44927.0	36094.0	38159.0	47240.0	45334.0	6235.0	7787.0	2513.0	6236.0	7294.0	17611.0	16099.0	1118.0	19966.0	13457.0	5551.0	18208.0	78375.0	123236.0
+(3.6)–(3.8)	z	3157.4	3257.5	3610.6	2898.9	3690.7	3223.9	2880.3	3187.1	2456.3	3080.9	3609.4	2747.7	2656.0	2973.8	3095.1	2569.7	2613.3	2971.8	3322.4	3037.0
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	1444.4	706.8	684.4	461.0	1105.7	1266.3	38.3	223.3	53.8	255.2	236.4	208.9	1463.4	262.8	597.3	484.0	310.9	332.9	1068.9	399.1
	bb	46692.0	16823.0	25771.0	13206.0	49178.0	85383.0	2031.0	8509.0	2440.0	14856.0	5639.0	5339.0	111814.0	7593.0	40443.0	16941.0	11625.0	6998.0	71667.0	15401.0
+(3.4)–(3.5)	z	3157.8	3145.9	3547.9	2874.0	3614.0	3225.0	2990.5	3139.9	2430.2	3052.4	3439.5	2774.1	2705.1	2861.9	3056.3	2550.2	2635.9	2892.3	3232.9	3004.6
	cpu_r	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	684.1	1772.8	321.8	1661.2	726.1	1800.0	49.5	312.7	245.7	243.1	364.4	181.5	1439.3	208.6	1280.7	409.3	196.9	380.4	1227.3	1134.3
	bb	25146.0	46772.0	9673.0	52232.0	31615.0	115724.0	2215.0	10804.0	7894.0	8556.0	8645.0	5013.0	98310.0	5618.0	81946.0	15550.0	8204.0	8554.0	84833.0	75879.0
+(3.9),(3.11)–(3.12)	z	3548.3	3723.5	3719.1	3163.0	3857.8	3484.8	3191.6	3356.7	2745.7	3383.6	3824.5	3031.1	2954.3	3080.3	3171.4	2874.2	2809.9	3115.6	3521.6	3135.5
	cpu_r	0.2	0.2	0.1	0.2	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	cpu	392.6	452.1	234.6	505.1	1061.4	653.8	187.6	219.9	33.8	221.2	61.4	63.4	366.7	52.7	437.2	114.3	77.0	270.7	330.6	338.7
	bb	5340.0	5302.0	3897.0	6362.0	15973.0	10635.0	2814.0	3960.0	882.0	3841.0	1223.0	1701.0	6217.0	1154.0	7435.0	3120.0	2192.0	3940.0	7838.0	7945.0
	cuts	25714.0	23990.0	17796.0	20901.0	101906.0	32878.0	14287.0	8260.0	2593.0	10723.0	4391.0	3926.0	23978.0	3954.0	22173.0	5866.0	5342.0	18986.0	14874.0	10726.0
+(3.11)–(3.12), (3.14)	z	3535.0	3725.6	3708.5	3163.0	3857.8	3484.8	3185.6	3356.7	2745.7	3383.6	3824.5	3031.1	2987.6	3080.3	3171.4	2890.7	2826.0	3115.6	3521.5	3135.5
	cpu_r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	cpu	582.2	284.0	585.1	379.8	313.9	774.5	516.3	335.7	28.8	59.6	87.2	60.0	320.9	47.8	693.6	274.9	246.2	19.8	765.8	320.4
	bb	10694.0	3796.0	11475.0	6245.0	4952.0	19518.0	13972.0	5383.0	745.0	1486.0	1805.0	1380.0	7275.0	1179.0	17338.0	7020.0	6059.0	616.0	14882.0	7885.0
	cuts	64657.0	12225.0	65384.0	23061.0	20357.0	124120.0	69252.0	22002.0	2011.0	4683.0	6953.0	2864.0	14613.0	2584.0	94834.0	11817.0	19290.0	1555.0	37790.0	10373.0
+(3.9),(3.11)–(3.12),(3.14)	z	3548.3	3725.6	3719.1	3163.0	3857.8	3484.8	3191.6	3356.7	2745.7	3383.6	3824.5	3031.1	3003.9	3080.3	3171.4	2890.7	2826.0	3115.6	3521.6	3135.5
	cpu_r	0.9	0.4	1.0	0.7	0.3	0.6	0.1	0.2	0.2	0.3	0.2	0.2	0.5	0.4	0.5	0.2	0.2	0.2	0.6	0.3
	cpu	881.7	574.1	1800.0	679.7	476.5	953.4	27.9	116.5	83.9	88.6	157.5	104.6	797.3	314.7	531.7	395.8	101.9	59.5	743.5	454.1
	bb	11860.0	4848.0	11947.0	6413.0	5124.0	10950.0	718.0	2226.0	1227.0	1415.0	1487.0	1671.0	9407.0	2759.0	9757.0	6312.0	1786.0	788.0	10174.0	6692.0
	cuts	150203.0	35951.0	217393.0	51568.0	40621.0	74514.0	3677.0	9860.0	11907.0	12370.0	15371.0	12477.0	54425.0	49573.0	61567.0	35811.0	11435.0	8476.0	70230.0	33674.0

Source: the author.

In Table 12, we report the summary of the impact of the proposed inequalities for model (FFP) to solve random and grid instances (FERONE *et al.*, 2019), and the new ones of Groups 1, 2, and 3. The first column 'Ineq' indicates the set of inequalities we add to model (FFP). In the first group of rows, we present, in the second column, for each set of test-bed instances, features details concerning average values of the optimal solutions in row 'opt' and the average number of 'colors' in these solutions; the average cost of the shortest paths 'sp' w.r.t. these instances and the average number of colors 'spc' in these shortest paths.

For each combination of valid inequalities we add to model (FFP), from the second to the last group of rows, we inform their impact on the average linear relaxed value z_r and the corresponding average cpu_r time; the average 'cpu' time to solve these instances and the average number of CPLEX B&B nodes 'bb'; and finally, when inequalities (3.12) are present, we also inform the average number of 'cuts' added to model (FFP) by a CPLEX user-cut callback. Average execution times equal to zero means values less than 0.05 second.

We separate results for distinct groups of inequalities by a horizontal line. For example, we report results for the set of constraints defining model (FFP) in the second group of rows. The second set of experiments is for model (FFP) with the addition of inequalities (3.14). We show in bold the best results for z_r, cpu, and bb for the new instances.

With regard to the benchmark instances (FERONE *et al.*, 2019) in Table 12, we observe that all valid inequalities cannot improve their average linear relaxed values and present small differences in both cpu_r and cpu times w.r.t. model (FFP). All these instances were solved at the root node of the B&B search tree, with their average shortest path cost being very close to the optimal linear relaxed and integer values (for random instances, *opt*=255.8 and *sp*=201.8, while for grid ones, *opt*=7936.9 and *sp*=7929.1). This also occurs for the average number of distinct colors in the shortest path solutions (for random instances, colors=5.7 and *spc*=8.1, while for grid ones, colors=246.3 and *spc*=248.5). On the other hand, for the new instances, on average, the shortest path *sp* is far from the optimal solution values opt (this is also true when comparing colors with *spc* of these instances). The differences between *sp* and *opt* is of 83.45%, 82.19%, and 81.93% for Groups 1, 2, and 3, respectively. Their average linear relaxed solution values (z_r) are also far from their average optimal values (*opt*). These instances require more average cpu times than the benchmark ones (FERONE *et al.*, 2019) despite their original dimensions. Whether we consider the individual use of a unique set of inequalities, the one (3.12) obtained the best results for these three groups for the linear relaxation and number of B&B nodes, while

Table 12 – The impact of the valid inequalities for model (FFP) for benchmark (FERONE *et al.*, 2019) and the instances of Groups 1, 2, and 3.

Ineq		Random	Grid	Group 1	Group 2	Group 3
Average features	opt	255.8	7936.9	6083.1	6229.3	5977.2
	sp	201.8	7929.1	1007.0	1109.4	1080.3
	colors	5.7	246.3	7.0	6.7	7.7
	spc	8.1	248.5	15.7	16.1	17.0
(FFP)	z_r	253.5	7934.9	2767.9	2938.4	2988.4
	cpu_r	2.9	0.1	0.1	0.1	0.1
	cpu	2.9	0.1	600.0	491.3	864.2
	bb	0.0	0.0	29577.4	18095.1	42479.2
+(3.14)	z_r	253.5	7934.9	3098.9	3261.2	3115.1
	cpu_r	2.9	0.1	0.0	0.0	0.0
	cpu	2.9	0.1	110.6	110.4	413.9
	bb	0.0	0.0	4308.3	4363.6	12027.2
+(3.12)	z_r	253.5	7934.9	3219.3	3365.5	3251.0
	cpu_r	2.5	0.0	0.2	0.2	0.2
	cpu	2.9	0.2	115.1	182.1	435.4
	bb	0.0	0.0	1853.6	2804.8	8430.6
	cuts	0.0	0.0	20436.5	36680.0	35234.7
+(3.11)	z_r	253.5	7934.9	2991.3	3155.8	3023.0
	cpu_r	2.9	0.1	0.1	0.1	0.1
	cpu	2.9	0.1	246.8	200.2	677.6
	bb	0.0	0.0	10797.7	7766.3	30258.5
+(3.10)	z_r	253.5	7934.9	2955.6	3122.0	3018.8
	cpu_r	2.9	0.1	0.1	0.1	0.1
	cpu	2.9	0.1	295.1	222.9	742.2
	bb	0.0	0.0	12224.1	8686.2	33156.6
+(3.9)	z_r	253.5	7934.9	3106.9	3264.8	3057.7
	cpu_r	2.8	0.1	0.1	0.1	0.1
	cpu	2.8	0.1	113.8	114.6	634.7
	bb	0.0	0.0	4257.9	4657.5	28757.6
+(3.6)–(3.8)	z_r	253.5	7934.9	3098.6	3255.6	3052.0
	cpu_r	3.3	0.1	0.1	0.1	0.1
	cpu	3.3	0.1	154.4	132.9	580.2
	bb	0.0	0.0	5214.5	5221.5	27917.5
+(3.4)–(3.5)	z_r	253.5	7934.9	2790.6	2958.1	3016.5
	cpu_r	3.0	0.1	0.1	0.1	0.1
	cpu	3.0	0.1	541.9	432.0	732.0
	bb	0.0	0.0	24581.0	15902.6	35159.2
+(3.9),(3.11)–(3.12)	z_r	253.5	7934.9	**3297.6**	**3440.4**	3284.6
	cpu_r	2.5	0.0	0.2	0.2	0.1
	cpu	3.0	0.1	72.2	83.3	**303.7**
	bb	0.0	0.0	1641.5	1698.3	**5088.6**
	cuts	0.0	0.0	13707.0	15739.4	18663.2
+(3.11)–(3.12),(3.14)	z_r	253.5	7934.9	3288.3	3435.6	3286.5
	cpu_r	0.0	0.0	0.0	0.0	0.0
	cpu	2.7	0.2	**45.8**	**76.4**	334.8
	bb	0.0	0.0	**1117.2**	**1630.5**	7185.3
	cuts	0.0	0.0	9288.3	14656.4	30521.2
+(3.9),(3.11)–(3.12),(3.14)	z_r	253.5	7934.9	**3297.6**	**3440.4**	**3288.8**
	cpu_r	2.5	0.0	0.4	0.5	0.4
	cpu	2.9	0.2	87.4	116.6	467.1
	bb	0.0	0.0	1261.8	1682.6	5378.1
	cuts	0.0	0.0	17448.7	24721.3	48055.2

Source: the author.

the best cpu times are provided by the set of inequalities (3.14).

Indeed, for Groups 1, 2 and 3, we observe an improvement on cpu times w.r.t. the one of model (FFP) of up to 81.57%, 77.52% and 52.10%, respectively. Concerning the combined use of valid inequalities, the models (FFP)+(3.9)(3.11)(3.12) and (FFP)+(3.9)(3.11)(3.12)(3.14) both obtain, on average, the strongest linear relaxed values for Groups 1 and 2. For the number

of B&B nodes, the best result is achieved by model (FFP)+(3.11)(3.12)(3.14) for Groups 1 and 2, while model (FFP)+(3.9)(3.11)(3.12) reached the smallest number of B&B nodes and cpu times for Group 3.

We remark that all proposed inequalities for model (FFP) improve its linear relaxation when handling the new instances. Since inequalities (3.9) dominate the ones (3.6)–(3.8), which in turn dominate inequalities (3.4)–(3.5), as well as inequalities (3.11) dominate the ones (3.10), we do not report results for models combining these dominated inequalities. With respect to execution times for instances of Groups 1 and 2, compared to those obtained by the DP algorithm, model (FFP)+(3.11)(3.12)(3.14) provides an improvement of 94.26% and 91.02%, respectively. We emphasize that the DP algorithm is unable to find optimal solutions for instances of Group 3 within the time limit, while model (FFP)+(3.11)(3.12)(3.14) obtained the optimal solution for all these instances. For grid and random instances, the DP procedure presents an increase of 96.15% and an improvement of 96.29% of execution times, respectively, in comparison to this model.

Table 13 – Statistics of the relative integrality gap (ratio) between optimal and linear relaxed solutions of variations of model (FPP) for Groups 1, 2, and 3.

	Ratios											
	Group 1				Group 2				Group 3			
Ineq	Mean	Median	Max	Min	Mean	Median	Max	Min	Mean	Median	Max	Min
(FFP)	119.8	118.3	190.5	51.1	112.6	106.5	172.1	66.7	98.9	89.0	173.0	62.1
+(3.14)	96.2	96.3	153.0	37.8	91.5	84.1	149.3	53.8	90.3	83.4	164.6	55.9
+(3.12)	88.6	90.0	137.1	36.7	85.5	79.5	135.3	46.3	82.8	78.3	147.8	50.6
+(3.11)	103.4	100.8	163.2	41.1	98.1	90.6	159.7	56.7	93.0	85.8	163.0	61.1
+(3.10)	105.9	103.7	165.6	43.3	100.2	91.8	163.3	58.5	93.4	87.3	161.7	60.8
+(3.9)	95.7	96.1	152.4	37.4	91.3	84.0	149.3	53.8	94.9	84.7	170.4	58.3
+(3.6)–(3.8)	96.2	96.5	153.0	37.9	91.8	84.3	150.4	54.3	95.3	86.2	170.4	55.8
+(3.4)–(3.5)	118.1	116.3	189.8	50.9	111.3	104.9	172.1	66.0	97.6	89.7	172.7	61.9
+(3.9),(3.11)–(3.12)	**84.1**	**83.9**	133.1	33.1	**81.6**	**76.1**	132.4	43.9	81.0	75.1	147.8	50.4
+(3.11)–(3.12),(3.14)	84.6	84.3	133.5	33.1	81.8	**76.1**	132.4	43.9	80.8	75.1	147.8	50.4
+(3.9),(3.11)–(3.12),(3.14)	**84.1**	**83.9**	133.1	33.1	**81.6**	**76.1**	132.4	43.9	**79.8**	**74.1**	147.8	50.4

Source: the author.

Table 13 reports statistics results for Groups 1,2, and 3, as 'Mean', 'Median', maximum 'Max', and minimum 'Min' ratio values w.r.t. the relative difference, in percentage, between the optimal and the linear relaxed solution value $(100(opt - z_r)/z_r)$ of the instances when solved with the use of valid inequalities for model (FPP). We highlight, in bold, the smallest mean and median values for each group.

We observe, considering the models from (FFP)+(3.14) to (FFP)+(3.4)–(3.5), that inequalities (3.12) provide the smallest mean and median ratios for all the three groups. Concerning the joint use of valid inequalities in the last three lines of this table, we note a

slight difference between the mean and median ratios of the three groups. Globally, model (FFP)+(3.9)(3.11)(3.12)(3.14) obtained the smallest mean and median ratios.

Table 14 – Statistics of results for the instances of Group 1.

Ineq.	linear relaxation				cpu				bb			
	Average	*Median*	*Max*	*Min*	*Average*	*Median*	*Max*	*Min*	*Average*	*Median*	*Max*	*Min*
(FFP)	2767.9	2756.9	3181.8	2314.4	600.0	507.6	1719.6	9.3	29577.4	18407.5	83716.0	363.0
+(3.14)	3098.9	3119.8	3652.8	2590.5	110.6	124.9	391.6	12.3	4308.3	4350.0	17390.0	363.0
+(3.12)	3219.3	3200.6	3897.6	2642.9	115.1	80.4	635.3	6.1	1853.6	1174.5	11730.0	53.0
+(3.11)	2991.3	3019.8	3510.8	2449.6	223.2	185.3	786.4	3.8	10797.7	7442.0	37225.0	130.0
+(3.10)	2955.6	2989.0	3479.7	2444.7	295.1	271.5	887.3	5.2	12224.1	9921.5	36441.0	144.0
+(3.9)	3106.9	3123.8	3661.8	2596.5	113.8	80.0	434.5	13.6	4257.9	3030.5	18477.0	306.0
+(3.6)–(3.8)	3098.6	3115.1	3652.7	2593.2	154.4	165.6	619.3	10.9	5214.5	4940.5	19951.0	268.0
+(3.4)–(3.5)	2790.6	2795.4	3188.7	2340.0	541.9	366.0	1738.5	24.1	24581.0	14189.0	70962.0	941.0
+(3.9),(3.11)–(3.12)	**3297.6**	3293.3	3965.2	2693.0	**72.2**	57.6	411.7	3.6	1641.5	1231.0	10203.0	68.0
+(3.11)–(3.12),(3.14)	3288.3	3286.2	3957.2	2686.9	45.8	43.0	137.5	2.7	**1117.2**	1067.0	2825.0	43.0
+(3.9),(3.11)–(3.12),(3.14)	**3297.6**	3293.3	3965.2	2693.0	87.4	51.5	422.4	2.9	1261.8	900.5	5329.0	45.0

Source: the author.

Tables 14–16 summarize the impact of the valid inequalities for model (FFP) with the instances of Groups 1, 2, and 3, respectively. For the legend, we inform the valid inequality *Ineq* strengthening model (FFP). We also show average (*Average*), *Median*, maximum (*Max*) and minimum (*Min*) values for the 'linear relaxation', the 'cpu' time to solve the instance, and the number of CPLEX B&B nodes 'bb'.

The first set of rows reports results for model (FFP) without the use of the valid inequalities. For the remaining groups of rows, we add to model (FFP) the inequalities informed in column *Ineq*. The best results among all employed inequalities are displayed in bold. Considering the inequalities individually, we observe that inequality (3.12) has the best overall impact, obtaining both the smallest average number of B&B nodes and the largest linear relaxed values, while inequalities (3.14) show the smallest average computational times. For Groups 1, 2 and 3, these inequalities allowed to improve the average cpu times w.r.t. model (FFP) in up to 81.57%, 77.52% and 52.10%, respectively.

Table 14 summarizes numerical results for instances of Group 1. The *Max* column for the cpu time shows that model (FFP)+(3.4)(3.5) can sometimes increase the computational time needed to solve the model in comparison to the standard (FFP).

As for the linear relaxation, the addition of inequalities (3.14) and (3.6)–(3.8) show very similar results, although inequality (3.14) requires less cpu time. Model (FFP)+(3.9)(3.11)(3.12) shows only a slight improvement of 2.43% on the linear relaxation, 59.41% on computational time and an increase of 22.05% in the number of nodes generated in the B&B search tree in comparison to the results obtained by inequality (3.12), obtaining both the strongest linear

relaxation and best computational time. In addition, model FFP+(3.11)(3.12)(3.14) obtained the smallest number of nodes in the B&B search tree.

Table 15 – Statistics of results for the instances of Group 2.

Ineq.	linear relaxation				cpu				bb			
	Average	*Median*	*Max*	*Min*	*Average*	*Median*	*Max*	*Min*	*Average*	*Median*	*Max*	*Min*
(FFP)	2938.4	2826.8	3450.0	2656.1	491.3	416.6	1614.7	126.3	18095.1	18069.5	44310.0	4469.0
+(3.14)	3261.2	3206.7	3802.7	2880.4	110.4	104.8	351.1	11.8	4363.6	3342.0	13303.0	660.0
+(3.12)	3365.5	3329.1	3823.2	3011.4	182.1	97.0	580.4	19.8	2804.8	1867.5	8797.0	345.0
+(3.11)	3155.8	3070.6	3703.4	2765.8	200.2	195.8	462.3	26.2	7766.3	7319.0	16921.0	1611.0
+(3.10)	3122.0	3043.3	3664.0	2764.9	222.9	206.3	559.8	17.9	8686.2	8224.0	20859.0	1043.0
+(3.9)	3264.8	3221.5	3803.1	2880.4	114.6	110.8	318.5	11.2	4657.5	4023.0	11072.0	598.0
+(3.6)–(3.8)	3255.6	3204.6	3800.1	2880.4	132.9	118.7	283.5	21.0	5221.5	4197.5	11790.0	1351.0
+(3.4)–(3.5)	2958.1	2851.8	3470.9	2663.5	432.0	342.2	1038.5	90.5	15902.6	14342.5	34393.0	2725.0
+(3.9),(3.11)–(3.12)	**3440.4**	3424.9	3934.8	3055.9	83.3	68.8	306.1	10.6	1698.3	1427.0	5089.0	190.0
+(3.11)–(3.12),(3.14)	3435.6	3406.8	3934.8	3054.1	**76.4**	59.5	283.5	13.4	**1630.5**	1273.0	5353.0	418.0
+(3.9),(3.11)–(3.12),(3.14)	**3440.4**	3424.9	3934.8	3055.9	116.6	88.1	399.9	26.3	1682.6	1386.5	5295.0	371.0

Source: the author.

Table 15 presents a summary of numerical results for instances of Group 2. On average, models (FFP)+(3.9)(3.11)(3.12)(3.14) and (FFP)+(3.9)(3.11)(3.12) show the strongest linear relaxation, a slight improvement of 2.23% in comparison to (FFP)+(3.12). In addition, model (FFP)+(3.11)(3.12)(3.14) shows the best cpu time, a decrease of 30.8% in comparison to (FFP)+(3.14), and smallest number of generated nodes in the B&B search tree, with an improvement of 41.87% in comparison to (FFP)+(3.12).

Table 16 – Statistics of results for the instances of Group 3.

Ineq.	linear relaxation				cpu				bb			
	Average	*Median*	*Max*	*Min*	*Average*	*Median*	*Max*	*Min*	*Average*	*Median*	*Max*	*Min*
(FFP)	2988.4	3026.6	3611.3	2423.5	864.2	714.2	1800.0	191.3	42479.2	39387.0	143871.0	5414.0
+(3.14)	3115.1	3073.3	3720.1	2485.8	413.9	235.1	1800.0	29.8	12027.2	8342.5	32957.0	1114.0
+(3.12)	3251.0	3167.1	3850.2	2741.3	435.4	418.6	1193.0	31.9	8430.6	6721.0	24230.0	602.0
+(3.11)	3023.0	3034.1	3614.0	2462.8	677.6	434.3	1800.0	130.1	30258.5	15243.5	119345.0	4888.0
+(3.10)	3018.8	3035.5	3611.5	2451.6	742.2	501.3	1800.0	43.0	33156.6	19594.5	101368.0	1870.0
+(3.9)	3057.7	3059.1	3614.0	2466.4	634.7	509.1	1800.0	25.4	28757.6	17909.5	123236.0	1118.0
+(3.6)–(3.8)	3052.0	3058.9	3690.7	2456.3	580.2	430.1	1463.4	38.3	27917.5	15128.5	111814.0	2031.0
+(3.4)–(3.5)	3016.5	3028.5	3614.0	2430.2	732.0	394.8	1800.0	49.5	35159.2	13177.0	115724.0	2215.0
+(3.9),(3.11)–(3.12)	3284.6	3181.5	3857.8	2745.7	**303.7**	252.6	1061.4	33.8	**5088.6**	3950.0	15973.0	882.0
+(3.11)–(3.12), (3.14)	3286.5	3178.5	3857.8	2745.7	334.8	317.1	774.5	19.8	7185.3	6152.0	19518.0	616.0
+(3.9),(3.11)–(3.12),(3.14)	**3288.8**	3181.5	3857.8	2745.7	467.1	424.9	1800.0	27.9	5378.1	4986.0	11947.0	718.0

Source: the author.

Table 16 shows a summary of the computational experiments for instances of Group 3. On average, model (FFP)+(3.9)(3.11)(3.12)(3.14) obtained the strongest linear relaxation, an increase of 1.16% over model (FFP)+(3.12). Furthermore, model (FFP)+(3.9)(3.11)(3.12) shows the best computational time, a decrease of 26.62% over (FFP)+(3.14), and lowest number of generated B&B nodes, an improvement of 39.64% over (FFP)+(3.12). For this group, some models failed to obtain the integer optimal solution (see column *Max* for the cpu times).

5 CONCLUSIONS

In this work, we proposed valid inequalities for the k-Color Shortest Path Problem and showed that they strengthen the existing formulation (FFP) introduced by Ferone *et al.* (2019), validating our hypothesis that our novel valid inequalities do improve the linear relaxation of the existing model.

One of the exponential-size set of valid inequalities was explored as a B&C algorithm for the k-CSPP, referred to as model (PCM). We also reproduced the instance reduction procedure of Cerrone and Russo (2023) and pointed that the Dijkstra-based heuristic CCDA can fail finding a feasible solution for the k-CSPP depending on the penalties adopted by this heuristic.

We observed that CPLEX found no obstacle in solving the reduced benchmark instances (FERONE *et al.*, 2019) at the root node of its B&B search tree with model (FFP). This is because the reduction procedure of Cerrone and Russo (2023) drastically reduces the large dimensions of almost all instances (excepting for three of them). Their linear relaxed, and optimal solution values are very close to the solution of their shortest paths. This motivated us to propose three groups of more difficult instances for the problem. The CCDA heuristic fails to find a feasible solution for Groups 1 and 3 of the new instances, and the quality of the solutions obtained for the instances of Group 2 is far from optimal. Moreover, the reduction procedure (CERRONE; RUSSO, 2023) was not able to reduce the number of arcs and vertices of these new instances. The values of the linear relaxed and optimal solution values are not close to the solution of the shortest paths for these instances.

Concerning the numerical results, for the new instances, inequalities (3.12) individually obtain the best improvement on the linear relaxation of the model (FFP) as well as on the reduction of the number of evaluated CPLEX B&B nodes. On the other hand, inequalities (3.14) obtain the smallest cpu times to solve these instances. Considering the combined use of the proposed inequalities, models (FFP)+(3.9)(3.11)(3.12) and (FFP)+(3.9)(3.11)(3.12)(3.14) obtained the best results for Groups 1 and 2, while the latter model attained slightly better results for Group 3. We emphasize that the B&B procedure (FERONE *et al.*, 2019) fails to find the optimal solution for all the new instances, while the DP algorithm (FERONE *et al.*, 2021) fails to find the optimal solutions for the instances of Group 3. Although DP algorithm finds optimal solutions for Groups 1 and 2, it requires more computational time compared to the mathematical models. We remark that despite the improvement in the linear relaxed solutions, they are still far from their optimal solutions, thus indicating that further research can be done in this direction.

The numerical results indicate that our new set of instances are indeed more challenging, i.e., requires more computational time to solve, than the ones introduced by Ferone *et al.* (2019). In addition, these results support our hypothesis that the methods proposed in the literature (FERONE *et al.*, 2019; FERONE *et al.*, 2021; CERRONE; RUSSO, 2023) are not efficient in our novel set of instances in comparison to the benchmark instances.

As future research, we intend to handle problems like the MCPP with the proposed inequalities, and investigate whether they can be further strengthened. It seems that maximal cliques of non-reachable arcs (or colors) can be used to obtain facets of the polyhedron associated with the model (FFP). We also plan on investigating the parameterized complexity of the k-CSPP. Furthermore, in order to avoid solving the same subproblem multiple times, it is feasible to store the subproblems that have already been solved, making a trade-off between memory and computational time. In addition, in order to assure the validity of our hypotheses under different scenarios, more numerical results analyzing different extraction policies for the DP algorithm and node evaluation policies for the B&B algorithm are necessary.

Lastly, we reinforce that preliminary results of this study were reported in Castelo *et al.* (2022) and in the XXI Latin Ibero-American Conference on Operations Research (ANDRADE *et al.*, 2022). The complete paper with all our findings was submitted for publication (ANDRADE *et al.*, 2023).

BIBLIOGRAPHY

ANDRADE, R.; CASTELO, E.; SARAIVA, R. D. Valid inequalities for the k-color shortest path problem. **European Journal of Operational Research**, 2023. Submitted.

ANDRADE, R.; SARAIVA, R. D.; CASTELO, E. Valid inequalities and a new integer programming model for the k-color shortest path problem. **XXI Latin Ibero-American Conference on Operations Research**, 2022. Disponível em: https://claio2022.dc.uba.ar/docs/abstract-book.pdf. Acesso em: 23 maio 2023.

BEASLEY, J. E.; CHRISTOFIDES, N. An algorithm for the resource constrained shortest path problem. **Networks**, v. 19, n. 4, p. 379–394, jul. 1989.

BELLMAN, R. On a routing problem. **Quarterly of applied mathematics**, v. 16, n. 1, p. 87–90, 1958.

CARRABS, F.; CERULLI, R.; FELICI, G.; SINGH, G. Exact approaches for the orderly colored longest path problem: Performance comparison. **Computers & Operations Research**, v. 101, p. 275–284, jan. 2019.

CASTELO, E.; ANDRADE, R.; SARAIVA, R. D. O problema do caminho mínimo k-rotulado. In: **Anais do Simpósio Brasileiro de Pesquisa Operacional**. Campinas: Galoá, 2022. Disponível em: https://proceedings.science/sbpo/sbpo-2022/trabalhos/o-problema-do-caminho-minimo-k-rotulado?lang=pt-br. Acesso em: 10 maio 2023.

CERRONE, C.; RUSSO, D. D. An exact reduction technique for the k-Colour Shortest Path Problem. **Computers & Operations Research**, v. 149, p. 106027, jan. 2023.

DEHOUCHE, N. The *k* -interchange-constrained diameter of a transit network: A connectedness indicator that accounts for travel convenience. **Transportation Letters**, v. 12, n. 3, p. 197–201, mar. 2020.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, v. 1, n. 1, p. 269–271, dez. 1959.

FERONE, D.; FESTA, P.; FUGARO, S.; PASTORE, T. A dynamic programming algorithm for solving the k-Color Shortest Path Problem. **Optimization Letters**, v. 15, n. 6, p. 1973–1992, set. 2021.

FERONE, D.; FESTA, P.; PASTORE, T. The k-Color Shortest Path Problem. In: PAOLUCCI, M.; SCIOMACHEN, A.; UBERTI, P. (Ed.). **Advances in Optimization and Decision Science for Society, Services and Enterprises**. Cham: Springer International Publishing, 2019. v. 3, p. 367–376.

GRANATA, D.; CERULLI, R.; SCUTELLÀ, M. G.; RAICONI, A. Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs. In: PARDALOS, P. M.; DU, D.-Z.; GRAHAM, R. L. (Ed.). **Handbook of Combinatorial Optimization**. New York, NY: Springer New York, 2013. p. 1913–1948.

KUMAR, N. Computing a Minimum Color Path in Edge-Colored Graphs. In: KOTSIREAS, I.; PARDALOS, P.; PARSOPOULOS, K. E.; SOURAVLIAS, D.; TSOKAS, A. (Ed.). **Analysis of Experimental Algorithms**. Cham: Springer International Publishing, 2019. v. 11544, p. 35–50.

LI, X.; ZHANG, S.; BROERSMA, H. Paths and cycles in colored graphs. **Electronic Notes in Discrete Mathematics**, v. 8, p. 128–132, 2001. ISSN 1571-0653. 1st Cologne-Twente Workshop on Graphs and Combinatorial Optimization. Disponível em: https://www.sciencedirect.com/science/article/pii/S1571065305800988.

MOTWANI, R.; RAGHAVAN, P. Randomized algorithms. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 28, n. 1, p. 33–37, 1996.

SANTOS, R. F.; ANDRIONI, A.; DRUMMOND, A. C.; XAVIER, E. C. Multicolour paths in graphs: NP-hardness, algorithms, and applications on routing in WDM networks. **Journal of Combinatorial Optimization**, v. 33, n. 2, p. 742–778, fev. 2017.

SILVA, T.; GUEYE, S.; MICHELON, P.; OCHI, L.; CABRAL, L. A polyhedral approach to the generalized minimum labeling spanning tree problem. **EURO Journal on Computational Optimization**, v. 7, n. 1, p. 47–77, mar. 2019.

YUAN, S.; VARMA, S.; JUE, J. Minimum-color path problems for reliability in mesh networks. In: **Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.** Miami, FL, USA: IEEE, 2005. v. 4, p. 2658–2669.