Computers & Operations Research 149 (2023) 106027

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

An exact reduction technique for the k-Colour Shortest Path Problem

Carmine Cerrone [a], Davide Donato Russo [b,*]

[a] Department of Economics, University of Genova, Genova, 16126, Italy
[b] Department of Biosciences and Territory, University of Molise, Pesche, 86090, Italy

ARTICLE INFO

Keywords:
Graph reduction
Dijkstra algorithm
Shortest path
Labelled graph

ABSTRACT

The k-Colour Shortest Path Problem is a variant of the classic Shortest Path Problem. This problem consists of finding a shortest path on a weighted edge-coloured graph, where the maximum number of different colours used in a feasible solution is fixed to be k. The k-CSPP has several real-world applications, particularly in network reliability. It addresses the problem of reducing the connection cost while improving the reliability of the network. In this work, we propose a heuristic approach, namely Colour-Constrained Dijkstra Algorithm (CCDA), which is able to produce effective solutions. We propose a graph reduction technique, namely the Graph Reduction Algorithm (GRA), which removes more than 90% of the nodes and edges from the input graph. Finally, using a Mixed-Integer Linear Programming (MILP) model, we present an exact approach, namely Reduced Integer Linear Programming Algorithm (RILP), that takes advantage of the heuristic CCDA and the GRA. Several tests were performed to verify the effectiveness of the proposed approaches. The computational results indicate that the produced approaches perform well, in terms of both the solution's quality and computation times.

1. Introduction

The large number of scientific publications concerning the Shortest Path Problem (SPP) and its innumerable variants is due to the capacity of this problem to model many real-life applications. An optimal shortest path, from a source node to a destination node, is a path that minimises a specified length criterion. Although the SPP can be solved in polynomial time using various algorithms, which are summarised by Madkour et al. (2017), many of its variants are known to be NP-hard. Some of the variants of this problem that fit various real-world applications are defined on an edge-coloured graph. In this work, we focus on a specific variant of the SPP defined on an edge-labelled graph. In the remainder of this paper, the terms 'colours' and 'labels' will be used interchangeably. These graphs, characterised by an enumerable characteristic associated with the edges that is usually called the colour or label, allow one to create generalised version of many problems defined on graphs. The Minimum Labelling Spanning Tree (MLST) Problem is a generalisation of the classic Minimum Spanning Tree Problem (Captivo et al., 2009), (Naji-Azimi et al., 2010) that, thanks to the use of labels, has been exploited to model problems concerning telecommunications networks. Cerulli et al. (2006) proposed various extensions of the problem and investigated several approaches, comparing their results and characteristics. Furthermore, Cerrone et al. (2019) introduced and analysed the Strong Generalised Minimum Label Spanning Tree Problem, presenting an Integer Linear Programming (ILP) formulation and three heuristic approaches. da Silva et al. (2019) provided a tighter bound on the time complexity of the MLST and proposed a new Mixed-Integer Programming based (MIP) meta-heuristic, namely multi-start local branching. Strictly related to the Minimum Labelling Spanning Tree Problem is the k-Labelled Spanning Forest Problem. Cerulli et al. (2014) defined this problem and proposed an exact approach to solve it, and Consoli et al. (2017) presented a comparison between the main meta-heuristic approaches. The Rainbow Spanning Forest Problem (RSFP) consists of finding a spanning forest of graph G such that the number of components (trees) is minimum, and each connected component contains only edges with different colours. Carrabs et al. (2018a), Carrabs et al. (2018b) proposed this problem, studied its complexity and then designed two approaches for it, one exact and one greedy. Labelled graphs have been used not only for tree and forest problems but also for path identification problems. In the Orderly Coloured Longest Path Problem (OCLPP), the aim is to find the longest possible path such that a set of constraints on the sequence of the colours is respected. Carrabs et al. (2019) performed several tests of all existing formulations, summarising their characteristics, in order to determine which of the formulations obtains better results and under what conditions. Yuan et al. (2005) developed an ILP formulation and a heuristic algorithm to minimise the number of colours for a path; indirectly, the authors succeeded in minimising the number of overlapping colours to prevent a single failure from causing multiple failures.

* Corresponding author.
E-mail addresses: carmine.cerrone@unige.it (C. Cerrone), d.russo7@studenti.unimol.it (D.D. Russo).

https://doi.org/10.1016/j.cor.2022.106027
Received 15 December 2021; Received in revised form 1 July 2022; Accepted 13 September 2022
Available online 24 September 2022

For a node-coloured graph, the All Colours Shortest Path Problem (ACSP), studied by Bilge et al. (2015), consists of finding a minimum-cost shortest path that uses all the different colours in an undirected weighted graph, where each node is assigned a colour. Carrabs et al. (2018c) proposed a variant of the problem where the source is unknown, and for this variant, they defined a mathematical formulation and a Variable Neighbourhood Search meta-heuristic. Finally, a problem family close to the colour problem family consist of resource-constrained problems. The Resource Constrained Shortest Path Problem (RCSPP) is based on the definition of an L-dimensional vector of resources R in addition to the graph G. In particular, each edge is linked to a resource attribute that needs to be addressed during path planning. A deep analysis of the RCSPP, its applications and possible approaches was performed by Irnich and Desaulniers (2005), Avella et al. (2004), Beasley and Christofides (1989) and Pugliese and Guerriero (2013). An interesting discussion of the RCSPP and the k-CSPP can be found in Ferone et al. (2019). To solve the RCSPP, Smith et al. (2012), Ferone et al. (2016), Marinakis et al. (2017) proposed an effective hybrid algorithm that combines the Particle Swarm Optimisation meta-heuristic and a Variable Neighbourhood Search approach. Tilk et al. (2017), exploiting the asymmetry in the number of forward and backward label extensions for the Shortest Path Problem with Resource Constraints (SPPRC), proposed an effective dynamic half way-point algorithm. Zhu and Wilhelm (2013) presented an implementation of a three-stage approach for the dynamic version of the RCSPP that has to be solved as a sub-problem in the branch-and-price algorithm, while Horváth and Kis (2016) investigated a Linear Programming (LP) based branch-and-bound method and introduced new cutting planes and separation procedures for the problem. Finally, Di Puglia Pugliese et al. (2019) analysed the problem in uncertain data conditions and proposed a robust formulation to obtain optimal solutions.

In this work, we focus on the k-Colour Shortest Path Problem (k-CSPP); the k-CSPP consists of finding a shortest path in a weighted edge-coloured graph, where the maximum number of different colours that can be used is fixed to be k. The first formal definition of the k-Colour Shortest Path Problem was presented by Ferone et al. (2019). They proposed a mathematical model based on a flow formulation and a Branch and Bound (B&B) Algorithm to solve the problem. After the formal definition of the problem, Ferone et al. presented a B&B approach based on the idea that by relaxing the colour constraints, the problem can be simply solved using a shortest path algorithm. After a relaxed solution is computed, if the number of colours used in the solution is higher than k, the solution is inadmissible; otherwise, the new solution can be evaluated in terms of the path length. It becomes the new incumbent solution if it is better than the previous best solution; otherwise, it is discarded. The problem of finding a path between a source node and a destination node with a fixed maximum number of colours, namely k-CSPP, was proven to be NP-hard by Broersma et al. (2005), who reduced it from the 3-SAT (3-Satisfiability) problem. In particular, the authors highlighted that any instances of the problem of finding a path with at most k colours can be related to an instance of k-CSPP, where each edge has a null cost. In their second work (Ferone et al., 2020), Ferone et al. proposed a novel dynamic programming (DP) algorithm that uses a path-labelling approach with an A^*-like technique as an exploration strategy. The authors also defined a new set of instances with fewer colours in order to analyse the behaviour of their approaches, varying the number of colours. Testing their approaches on two test sets highlighted that the DP algorithm outperformed previous approaches in terms of the number of optimum solutions and the computational time.

In this paper, we propose a heuristic approach (CCDA) to the k-CSPP that is able to identify optimal or near-optimal solutions regardless of the size of the instances; we propose an effective Graph Reduction algorithm (GRA) that is able to remove from the graph more than 90% of the nodes and edges; and we propose an exact approach (RILP) that combines the two approaches described previously and guarantees optimal solutions in a reasonable running time. Finally, we perform several tests to address the effectiveness of the proposed approaches and to analyse the characteristics of the instances used as benchmarks.

The paper is organised as follows: Section 2 contains an overview of the formulation of the problem. In Sections 3–5, the proposed approaches are investigated; Section 6 contains an analysis of the results and a comparison with state-of-the-art methods. Finally, Section 7 describes some possible future work on the k-Colour Shortest Path Problem.

2. Mathematical model

In the rest of this paper, we will use $P(n_s, n_d)$ to refer to a general path from node n_s to node n_d and $l(P(n_s, n_d))$ to refer to the length of the path $P(n_s, n_d)$. The mathematical model, based on a flow formulation, is presented here to clarify the characteristics of the problem (see the model proposed by Ferone et al. in Ferone et al. (2019)). Let G be a graph $G = (N, E)$, with $|N|$ nodes and $|E|$ edges. $C : E \longrightarrow \mathbb{N}$ and $d : E \longrightarrow \mathbb{R}_0^+$ are two functions: the first is an edge-colouring function, where the colour of the edge e is defined as a positive integer $C(e), \forall e \in E$. The second is a non-negative distance function defined for each edge $e \in E$. We define $C = \bigcup_{e\in E}\{C(e)\}$ as the set containing all the colours of G. For each colour $h \in C$, we define $E_h = \{e \in E s.t. C(e) = h\}$ as the set containing all the arcs with colours equal to h.

The k-CSPP consists of finding a shortest path $P^* = (n_s, \dots, n_i, \dots, n_d)$ from a source node n_s to a destination node n_d, with $n_s, n_d \in N$, such that the number of different colours traversed on the path does not exceed k. The flow formulation proposed by Ferone et al. considers two Boolean decision variables: the first, x_{ij}, is related to each edge of E; this variable will be equal to 1 if edge (i, j) belongs to P^*, and it will be equal to 0 otherwise. The second decision variable, y_h, is related to each possible colour such that $y_h = 1$ if at least one edge of colour h is traversed in P^*, and $y_h = 0$ otherwise. The parameter b_i defined for each $i \in N$ is:

$$b_i = \begin{cases} -1, & \text{if } i = n_s \\ +1, & \text{if } i = n_d \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the problem can be formulated as follows:

$$z = min \sum_{(i,j)\in E} d(i,j)x_{ij}, \tag{1}$$

subject to

$$\sum_{(j,i)\in E} x_{ji} - \sum_{(i,j)\in E} x_{ij} = b_i \qquad \forall i \in N, \tag{2}$$

$$x_{ij} \le y_h \qquad \forall h \in C, \forall (i,j) \in E_h, \tag{3}$$

$$\sum_{h\in C} y_h \le k, \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E, \tag{5}$$

$$y_h \in \{0,1\} \qquad \forall h \in C. \tag{6}$$

The objective function (1) minimises the total length of the solution path P^*. The constraints in (2) are the flow conservation constraints. The constraints in (3) enable the passage of the flow only on arcs whose colour is selected. Constraint (4) is used to limit the maximum number of useable colours to k. Finally, constraints (5) and (6) are used to define the Boolean domain of the decision variables.

3. Colour-Constrained Dijkstra Algorithm

The heuristic approach proposed in this work, namely Colour-Constrained Dijkstra Algorithm (CCDA), is an extension of the well-known Dijkstra Algorithm (Dijkstra et al., 1959) that, taking into account the colour function C, tries to define admissible solutions for the k-Colour Shortest Path Problem. In particular, the approach is

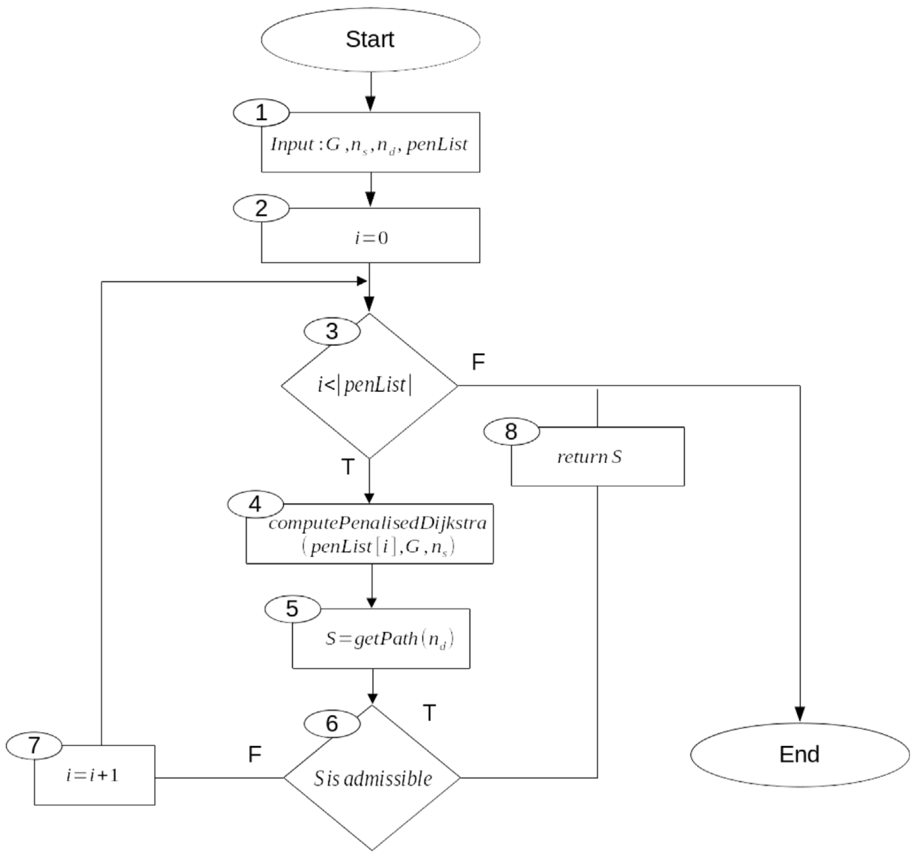


Fig. 1. Flowchart of the Colour-Constrained Dijkstra Algorithm (CCDA), described in Section 3, that extends the well-known Dijkstra Algorithm to handle the edge colours and produces solutions for the k-CSPP.

designed to compute admissible solutions by modifying dynamically, during the computation, the costs of the edges according to their colour. During the computation, the Dijkstra Algorithm updates a shortest path from a source node n_s to all the other nodes of the graph, considering, in each step, the minimum distance from the source node to the current node n_i, plus the edge cost between n_i and the next node n_j in the path sequence. Our approach modifies the cost between n_i and n_j, adding a penalisation value, namely $penalty$, if the colour of the edge (n_i, n_j) is not used in the previously computed path $P(n_s, n_i)$. Starting from a penalisation value equal to 0, the approach iterates several times, increasing the penalisation value each time. It starts from 0 to check if the canonical shortest path is also a solution for the k-CSPP. This is because the smaller the $penalty$ value, the closer the final path will be to a shortest path without penalisation, and so the better the fitness value of the solution will be. Therefore, the higher the $penalty$ value, the higher the probability of avoiding new colours and thus, the higher the probability of identifying admissible solutions. The penalty values used are those included in the list $penList$. To introduce the values contained in this list, we must define the following values for each instance:

$$minEdgeCost = min\{d_{i,j} | (i,j) \in E\},$$
$$maxEdgeCost = max\{d_{i,j} | (i,j) \in E\},$$
$$sumEdgeCost = \sum_{(i,j)\in E} d_{i,j},$$
$$meanEdgeCost = \frac{sumEdgeCost}{|E|}.$$

The penalty list used is the following:

$$penList = (0, \frac{minEdgeCost}{4}, \frac{minEdgeCost}{2}, minEdgeCost, minEdgeCost * 2, \frac{meanEdgeCost}{4}, \frac{meanEdgeCost}{2}, meanEdgeCost, maxEdgeCost).$$

As can be seen in Fig. 1, the CCDA proposed is very simple, but as described in Section 6, it is very effective. In particular, it takes as input the graph G, the source and destination nodes n_s and n_d, and the penalties list $penList$ in Step (1). The algorithm starts iterating over the various penalty values in Step (3), trying each time, in Steps (4) and (5), to compute a shortest path with a different value of the penalty for the usage of new colours. If the current penalty value allows the Penalised Dijkstra Algorithm to find an admissible solution (Step (6)), the approach ends, returning, in Step (8), the identified solution; otherwise, it reiterates with a new penalty value (Step (7)).

A key role in this approach is played by the list of used colours. The Penalised Dijkstra Algorithm, used in Step (4), differs from the well-known standard algorithm because it takes into account the colours used in the path from the source to all the other nodes; thus, instead of memorising only the previous node and the minimum distance from the source, our approach also considers the colours used in the path. For example, when the algorithm is computing a shortest path for the node n_j, with n_i as the previous node, it checks if the colour of the edge n_i, n_j was used for the path connecting n_s and n_i: if it was, then the edge is considered with its own cost; otherwise, its cost is increased by the penalty value. The penalty value is given as input to the Penalised Dijkstra Algorithm and increases for each iteration of the CCDA. In this way, according to the current penalty, it is possible to produce a shortest path that uses fewer colours compared to the original algorithm, with a small increase in the computational complexity of the algorithm.

4. Graph Reduction Algorithm

For the k-CSPP, the complexity grows as the input size increases. Unfortunately, unlike the classic SPP for which the complexity grows

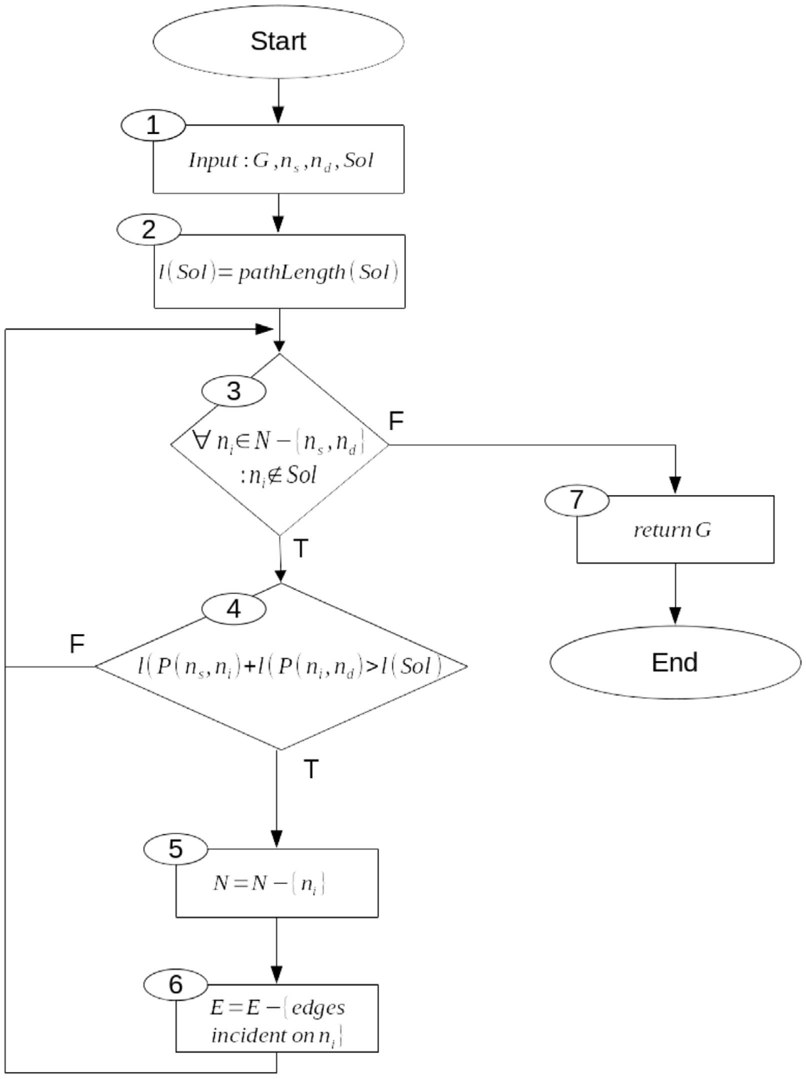

Start
1
$Input: G, n_s, n_d, Sol$
2
$l(Sol) = pathLength(Sol)$
3
$\forall n_i \in N - \{n_s, n_d\} : n_i \notin Sol$
F
T
4
$l(P(n_s, n_i)) + l(P(n_i, n_d)) > l(Sol)$
F
T
5
$N = N - \{n_i\}$
6
$E = E - \{edges\ incident\ on\ n_i\}$
7
$return\ G$
End

Fig. 2. Flowchart of the Graph Reduction Algorithm described in Section 4. This algorithm, depending on the quality of the upper bound provided as input, is able to drastically reduce the size of the instances, speeding up the computation time for further algorithms applied to the problem.

polynomially, for the k-CSPP, the complexity grows exponentially (Broersma et al., 2005). This means that reducing the size of the input is crucial for lowering the computation time and creating exact approaches that are useable in large scenarios. To solve this problem, in this work, we propose a Graph Reduction Algorithm (GRA) that is able to reduce the size of an instance in order to apply the ILP formulation and obtain optimal solutions for many instances, even in the case of large sizes.

The GRA consists of removing from the original graph G as many nodes and edges that cannot be in the optimal solution as possible. In particular, our reduction algorithm starts by computing an upper bound (UB) for the problem using a heuristic or meta-heuristic approach; we used the Colour-Constrained Dijkstra Algorithm proposed in Section 3. The algorithm removes from the original graph G all the nodes and edges that, if they are forced to be in the solution, cause a fitness value higher than the upper bound. Now, given that the k-CSPP is a minimisation problem, the lower the objective function of the UB, the higher the number of nodes and edges removed. More formally, a heuristic solution path $P^h(n_s, n_d)$, with a length of $l(P^h(n_s, n_d))$, is computed. Then, $\forall n_i \notin P^h(n_s, n_d)$, if the shortest path length from the source node n_s to n_i plus the shortest path length from n_i to the destination node n_d is greater than $l(P^h(n_s, n_d))$, then the node n_i and all its incident edges are removed from the graph G.

A formal proof that this algorithm does not remove from the graph G nodes that can belong to the optimum solution S^*, or any solution S better than the upper bound provided, follows.

Proposition 1. *Consider a graph $G = (N, E)$, where n_s, $n_d \in N$ are respectively defined as the source and destination nodes. Consider a generic path $P^h(n_s, n_d)$ and a generic node $n_i \in N$ such that $SP(n_s, n_i)$ is a shortest path from the source node n_s to n_i, and $SP(n_i, n_d)$ is a shortest path from the node n_i to the destination node n_d. The node n_i cannot belong to path P^h.*

If

$$l(SP(n_s, n_i)) + l(SP(n_i, n_d)) > l(P^h(n_s, n_d)), \tag{7}$$

then

$$\forall\ path\ P \in \{P(n_s, n_d)\ s.t.\ n_i \in P(n_s, n_d)\},\quad l(P) > l(P^h(n_s, n_d)). \tag{8}$$

Proof. In order to prove this implication, we propose a proof based on a *reductio ad absurdum*. It is possible to assume that if (7) is satisfied for the node n_i, then there exists a generic path $P^1(n_s, n_d)$ such that

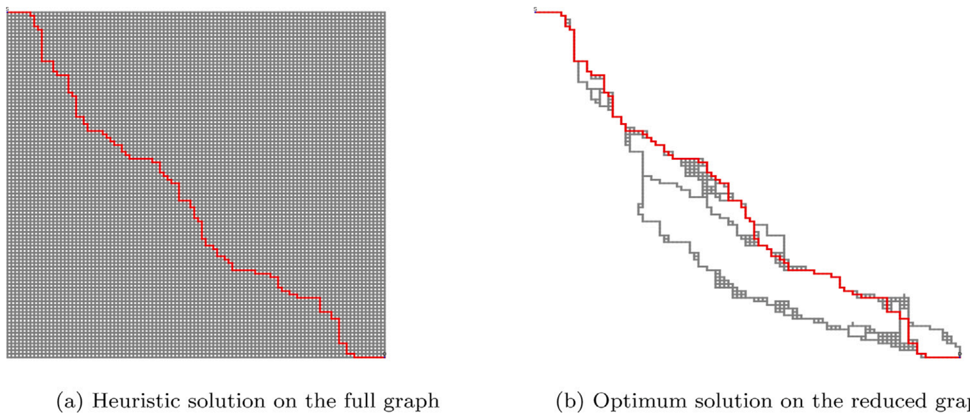

(a) Heuristic solution on the full graph
(b) Optimum solution on the reduced graph

Fig. 3. Comparison between heuristic and optimum solutions on the original graph and the graph reduced using the approach proposed in Section 4. As can be seen, the reduction process removes more than the 90% of the graph without affecting the optimal solution.

Table 1
Summary of the characteristics of grid instances belonging to set A.

Name	$\|V\|$	$\|E\|$	%Colour	$\|C\|$	k
A-G1	10000	39600	15%	5940	194.90
A-G2	10000	39600	20%	7920	195.40
A-G3	20000	79400	15%	11910	298.10
A-G4	20000	79400	20%	15880	298.90
A-G5	62500	249000	15%	37350	498.50
A-G6	62500	249000	20%	49800	499.60
A-G7	125000	498500	15%	74775	765.10
A-G8	125000	498500	20%	99700	766.30
A-G9	250000	998000	15%	149700	1005.10
A-G10	250000	998000	20%	199600	1005.30
A-G11	500000	1997000	15%	299550	1533.70
A-G12	500000	1997000	20%	399400	1535.00

Table 2
Summary of the characteristics of grid instances belonging to set B.

Name	$\|V\|$	$\|E\|$	%Colour	$\|C\|$	k
B-G1	10000	39600	1%	396	154.30
B-G2	10000	39600	2%	792	174.10
B-G3	20000	79400	1%	794	250.00
B-G4	20000	79400	2%	1588	273.70
B-G5	62500	249000	1%	2490	455.90
B-G6	62500	249000	2%	4980	480.20
B-G7	125000	498500	1%	4985	713.90
B-G8	125000	498500	2%	9970	740.40
B-G9	250000	998000	1%	9980	960.00
B-G10	250000	998000	2%	19960	985.70
B-G11	500000	1997000	1%	19970	1479.50
B-G12	500000	1997000	2%	39940	1508.70

$n_i \in P^1$ and $l(P^1) \le l(P^h(n_s, n_d))$. The path P^1 can be divided into two paths $P^2(n_s, n_i)$ and $P^3(n_i, n_d)$. Thus, we can write $l(P^h(n_s, n_d)) > l(P^1) = l(P^2(n_s, n_i)) + l(P^3(n_i, n_d)) > l(SP(n_s, n_i)) + l(SP(n_i, n_d))$, which is absurd with respect to (7). □

Fig. 2 presents the flowchart of the reduction algorithm proposed in this section. The GRA takes as input, in Step (1) , the graph G, the source and destination nodes n_s and n_d, and an initial solution Sol. It computes the length of the solution as the upper bound for the problem in Step (2) . In Step (3), for each node $n_i \in N$ such that $n_i \notin Sol$, $n_i \neq n_s$ and $n_i \neq n_d$, if the length of the shortest path between n_s and n_i, namely $l(P(n_s, n_i))$, plus the length of the shortest path between n_i and n_d, namely $l(P(n_i, n_d))$, is greater than $l(Sol)$ (Step (4)), then the node n_i is removed from the set of nodes N (Step (5)), and all the edges incident on n_i are also removed from E in Step (6). In Step (7), the algorithm returns the graph without the nodes and edges that were removed during this process.

Fig. 3 shows an example of the results produced by the reduction algorithm. Fig. 3(a) shows an initial graph with a heuristic solution highlighted in red, and Fig. 3(b) shows the reduced graph with the optimum solution highlighted in red. As can be seen in this figure, which represents the instance Grid_100x100_5940_27000, the graph is reduced by more than 90% with respect to the original size of the graph, and none of the nodes and edges of the optimal solution are removed in the reduction process.

Sometimes, after the application of GRA, a node n_i can be identified in the reduced graph that, if it is removed, causes a disconnection of the graph and the creation of two connected components. This scenario occurred quite often during our tests, as can be seen from Fig. 3(b). This scenario can be considered to improve the performance of some solution approaches. Any approach used to compute the path between the source node n_s and the destination node n_d can be applied twice in parallel, first from n_s to n_i and second from n_i to n_d. This parallelisation can drastically reduce the computation time. However, in the k-CSPP, this idea cannot be applied. This is because the colour variables must be considered for the global path and not only in the sub-graph. This approach can cause the creation of an inadmissible path that uses more than k colours. Despite this, it may be interesting to consider this scenario in further research.

Table 3
Summary of the characteristics of random instances belonging to set A.

Name	$\|V\|$	$\|E\|$	%Colour	$\|C\|$	k
A-R1	75000	750000	15%	112500	6.10
A-R2	75000	750000	20%	150000	6.00
A-R3	75000	1125000	15%	168750	5.20
A-R4	75000	1125000	20%	225000	5.20
A-R5	75000	1500000	15%	225000	5.50
A-R6	75000	1500000	20%	300000	5.50
A-R7	100000	1000000	15%	150000	5.70
A-R8	100000	1000000	20%	200000	6.30
A-R9	100000	1500000	15%	225000	5.30
A-R10	100000	1500000	20%	300000	5.30
A-R11	100000	2000000	15%	300000	5.60
A-R12	100000	2000000	20%	400000	5.60
A-R13	125000	1250000	15%	187500	6.30
A-R14	125000	1250000	20%	250000	6.30
A-R15	125000	1875000	15%	281250	5.30
A-R16	125000	1875000	20%	375000	5.30
A-R17	125000	2500000	15%	375000	5.40
A-R18	125000	2500000	20%	500000	5.40

Table 4
Summary of the characteristics of random instances belonging to set B.

Name	$\|V\|$	$\|E\|$	%Colour	$\|C\|$	k
B-R1	75000	750000	1%	7500	5.60
B-R2	75000	750000	2%	15000	5.60
B-R3	75000	1125000	1%	11250	4.80
B-R4	75000	1125000	2%	22500	4.80
B-R5	75000	1500000	1%	15000	4.20
B-R6	75000	1500000	2%	30000	4.20
B-R7	100000	1000000	1%	10000	6.10
B-R8	100000	1000000	2%	20000	6.30
B-R9	100000	1500000	1%	15000	5.00
B-R10	100000	1500000	2%	30000	5.00
B-R11	100000	2000000	1%	20000	4.70
B-R12	100000	2000000	2%	40000	4.70
B-R13	125000	1250000	1%	12500	5.90
B-R14	125000	1250000	2%	25000	5.90
B-R15	125000	1875000	1%	18750	5.30
B-R16	125000	1875000	2%	37500	5.30
B-R17	125000	2500000	1%	25000	4.10
B-R18	125000	2500000	2%	50000	4.10

5. Reduced ILP

The purpose of the exact approach proposed in this work is to drastically reduce the computational time compared to the other exact approaches proposed in the literature. To achieve this objective, we combine the effectiveness of the ILP formulation and the speediness of our heuristic algorithm. In order to compute an optimal solution, we propose the Reduced ILP (RILP) algorithm. Our algorithm starts by computing an admissible solution using the heuristic approach described in Section 3. If this solution is optimal, the algorithm stops and returns the identified optimal solution. If the heuristic solution is not optimal, it can be used as an upper bound for the reduction algorithm proposed in Section 4. We use the GRA to remove, from the graph G, all the nodes and edges that are useless in the discovery of the optimal solution. Finally, on the reduced graph, our approach executes the ILP formulation to identify the optimal solution.

Fig. 4 shows the flowchart of the RILP approach. The technique takes as input the graph G, the source node n_s and the destination node n_d, and the penalty list $penList$ (Step (1)). Then, using the proposed heuristic algorithm, a solution Sol, considered as the upper bound for the algorithm, is computed in Step (2). A shortest path on graph G is computed in Step (3), and its length is compared with the upped bound solution in Step (4). If the length of the heuristic solution is equal to the length of the shortest path, then the algorithm stops, and the obtained solution is returned (Step (9)). Otherwise, the GRA is applied on graph G in Step (5). After the reduction phase has removed from the graph all the useless nodes and edges, the ILP formulation is applied in Step (6) to obtain the optimal solution. Finally, the solution obtained in Step (7) is returned in Step (8).

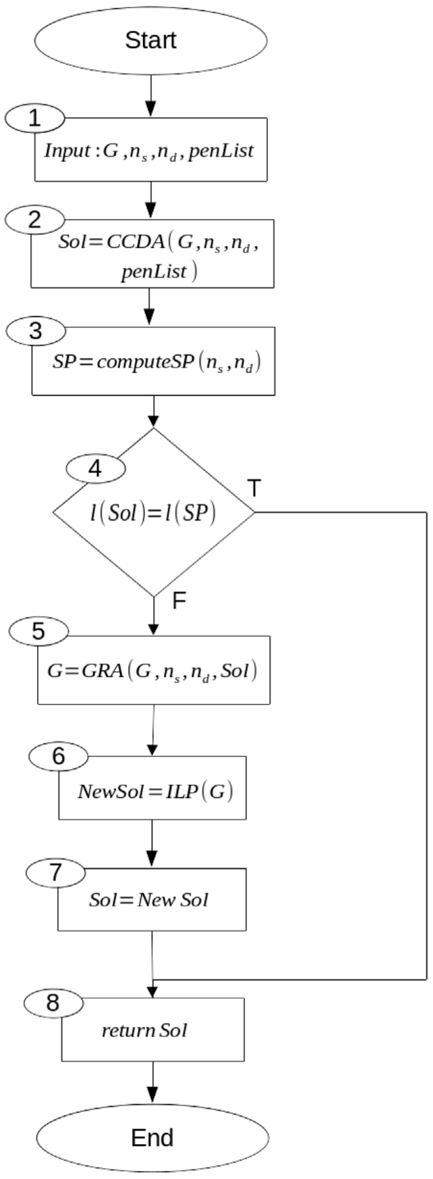

Start
1
Input : G , n_s , n_d , penList
2
Sol = CCDA(G , n_s , n_d , penList)
3
SP = computeSP(n_s , n_d)
4
l(Sol) = l(SP)
T
F
5
G = GRA(G , n_s , n_d , Sol)
6
NewSol = ILP(G)
7
Sol = New Sol
8
return Sol
End

Fig. 4. Flowchart of the exact approach described in Section 5. The algorithm, exploiting the properties of the Colour-Constrained Dijkstra Algorithm and the Graph Reduction Algorithm, guarantees optimal solutions in a reasonable amount of time.

6. Results

In order to investigate the effectiveness of the proposed approaches, in this section, we show and discuss the results of our computational experiments. To validate the solutions produced by the approaches developed in this work, we have compared our solutions with the best solutions from the literature. We used the two sets of benchmark instances proposed by Ferone et al. in Ferone et al. (2019, 2020). These sets of instances, namely set A (Tables 1 and 3) and set B (Tables 2 and 4), are characterised by grid-graph-based instances and random-graph-based instances, respectively.

These four sets of instances are made up of 10 different instances for each scenario; the scenarios are identified in the 'name' column of each table. In particular, in the first column, the name that represents the group of 10 instances is shown. Then, in order, the following attributes are reported: the number of nodes, the number of edges, the number of colours divided by the number of edges (given as a percentage), the number of colours and finally, the maximum number of colours (k) that can be used to generate a solution. Note that in each instance, each value is an integer number even if the mean value reported is real. For all of the following tables, for each row, we provide the mean values of the ten instances.

Information about the instances used to test our approaches is shown in Tables 1–4; these tables, for each scenario composed of 10 instances, report the average values.

As can be deduced from the tables, the main difference between sets A and B, for both grid and random instances, is related to the number of colours: in set A, the number of colours is equal to 15% or 20% of the number of edges, while in set B, the number of colours is equal to 1% or 2% of the number of edges. The main difference between grid instances and random instances, in addition to the graph structure, lies in the value of the parameter k. For the grid instances, this value increases proportionally with the size of the graph, while for

Table 5
Comparison of the performances of the ILP formulation, RILP and the dynamic programming technique on grid instances from set A.

Name	ILP				RILP			DP	
	ff	time	#Feas	#Opt	ff	time	#Opt	time	#Opt
A-G1	6203.40	37.66	10	10	6203.40	0.27	10	0.92	10
A-G2	6204.40	35.49	10	10	6204.40	0.24	10	60.81	9
A-G3	9336.60	232.49	10	9	9670.40	0.41	10	3.81	10
A-G4	9670.80	202.28	10	10	9670.80	0.37	10	12.69	10
A-G5	25073.56	827.00	9	–	15448.10	0.79	10	130.83	8
A-G6	25538.00	TL	8	1	15448.70	0.81	10	130.01	8
A-G7	39181.30	TL	10	–	23876.40	2.05	10	64.99	9
A-G8	40581.80	TL	10	–	23876.00	2.20	10	62.00	9
A-G9	52656.56	TL	9	–	30808.10	4.89	10	367.50	4
A-G10	54160.90	TL	10	–	30808.00	5.52	10	362.37	4
A-G11	79212.20	TL	10	–	47639.70	11.90	10	232.51	7
A-G12	82025.50	TL	10	–	47639.70	11.90	10	275.19	6
Mean		**645.73**	**9.67**	**3.33**		**3.45**	**10**	**141.97**	**7.83**

Table 6
Comparison of the performances of the ILP formulation, RILP and the dynamic programming technique on grid instances from set B.

Name	ILP				RILP			DP	
	ff	time	#Feas	#Opt	ff	time	#Opt	time	#Opt
B-G1	6196.70	18.64	10	9	6197.50	0.16	10	0.03	10
B-G2	6200.00	32.21	10	9	6199.60	0.20	10	0.07	10
B-G3	9656.70	119.63	10	9	9665.40	0.56	10	0.11	10
B-G4	9656.60	124.20	10	9	9665.30	0.37	10	0.09	10
B-G5	–	TL	–	–	15444.40	0.86	10	0.92	10
B-G6	25779.40	TL	10	1	15444.50	0.72	10	0.79	10
B-G7	–	TL	–	–	23874.60	2.15	10	1.01	10
B-G8	40506.70	TL	10	–	23875.20	2.30	10	2.41	10
B-G9	–	TL	–	–	30806.30	4.60	10	50.67	10
B-G10	54227.70	TL	10	–	30805.80	4.51	10	18.37	10
B-G11	–	TL	–	–	47638.80	13.90	10	46.33	10
B-G12	81828.50	TL	10	–	47639.00	11.35	10	69.49	10
Mean		**630.04**	**6.67**	**3.08**		**3.47**	**10**	**15.86**	**10**

the random instances, the value of the parameter k is small (lower than 10) and independent of the size of the instances.

In the remainder of this section, we show the results of our computational experiments. The tests were performed using CentOS Linux 7 with an Intel Xeon CPU E5-2650 v3 2.30 GHz and 126 GB of RAM. The approaches were implemented in Java 1.8, and the mathematical formulation was implemented using CPLEX 12.10. The time limit imposed on all the algorithms was 900 s.

6.1. Comparison results

The computational results are organised as follows. Tables 5–8 show a comparison between the exact approach proposed in this work, namely RILP, the mathematical formulation presented in Section 2, namely ILP, and the state-of-the-art dynamic programming approach proposed by Ferone et al. (2020). Results are reported for both the grid and random instances of sets A and B. Furthermore, Tables 9–12 show a comparison between our CCDA and the RILP approach to assess the effectiveness of our heuristic compared to the optimal solutions.

Tables 5 and 6 compare the ILP formulation, RILP and the dynamic programming technique. For all the approaches, these tables show the computation time (in seconds) and the number of optimum solutions identified. For ILP and RILP, the fitness function is also shown. For ILP the number of feasible solutions identified is reported. For DP and RILP, the number of feasible solutions is not shown because all of the feasible solutions are also optimum. Finally, for DP, the objective function is not shown for the reason described previously. A summary row is given at the end of both tables.

As can be seen in Tables 5 and 6, for grid instances, the ILP formulation is able to obtain optimal solutions only in small instances; therefore, it loses effectiveness as the instance size increases. In fact, for medium and large instances, it can barely obtain admissible solutions, often reaching the time-limit value. However, our exact approach also performs better than the dynamic programming technique; it is able to identify, in all cases, the optimum solution in a small amount of time, while the DP approach is not able to identify a feasible solution for the grid instances of set A, especially the largest ones. On the grid instances of set B , even though the DP approach is able to identify the optimum solution in all cases, our approach requires, on average, less time. In particular, the time required by the DP algorithm increases quickly as the instance size increases. This demonstrates RILP's effectiveness regardless of the size of the instances.

Tables 7 and 8 compare the ILP formulation, RILP and the dynamic programming technique. For all the approaches, the tables show the computation time (in seconds) and the number of optimum solutions identified. All the methods are able to identify the optimum solutions in all cases. The first column reports the average fitness value. A summary row is given at the end of the table.

For the random scenarios (Tables 7 and 8), all three approaches can always identify optimum solutions. Therefore, in these instances, the results highlight that the DP approach is the best algorithm in terms of the computation time. Even though both the RILP and DP approaches are faster than the ILP, for both set A and set B, the DP approach is the fastest method; in all cases, it requires less than 2 s. The DP approach only required more time when it found unfeasible instances. This is because this technique has an advantage in scenarios where the number

Table 7
Comparison of the performances of the ILP formulation, RILP and the dynamic programming technique. The methods were applied on random instances from set A. The scenarios in which there is an instance that does not allow admissible solutions for the given k value are shown in bold.

Name	ff Opt	ILP		RILP		DP	
		time	#Opt	time	#Opt	time	#Opt
A-R1	250.18	172.90	10	17.92	10	1.56	10
A-R2	**210.80**	**95.14**	**9**	**6.60**	**9**	**2.35**	**9**
A-R3	**187.80**	**131.23**	**9**	**6.97**	**9**	**60.24**	**9**
A-R4	**187.80**	**132.51**	**9**	**7.37**	**9**	**60.24**	**9**
A-R5	179.55	231.90	10	8.21	10	0.47	10
A-R6	179.55	248.22	10	7.98	10	0.49	10
A-R7	232.91	139.55	10	12.50	10	0.65	10
A-R8	**224.80**	**128.55**	**9**	**10.29**	**9**	**3.95**	**9**
A-R9	195.27	232.27	10	13.05	10	0.47	10
A-R10	195.27	259.18	10	13.15	10	0.50	10
A-R11	164.45	265.73	10	10.98	10	0.49	10
A-R12	164.45	269.23	10	10.85	10	0.50	10
A-R13	242.18	178.60	10	17.56	10	0.43	10
A-R14	242.18	208.26	10	18.08	10	0.43	10
A-R15	209.73	287.02	10	20.60	10	0.63	10
A-R16	209.73	336.75	10	20.59	10	0.64	10
A-R17	164.36	408.05	10	20.36	10	0.71	10
A-R18	164.36	453.29	10	19.30	10	0.72	10
Mean		**232.13**	**9.78**	**13.46**	**9.78**	**7.53**	**9.78**

Table 8
Comparison of the performances of the ILP formulation, RILP and the dynamic programming technique. The methods were applied on random instances from set B.

Name	ff Opt	ILP		RILP		DP	
		time	#Opt	time	#Opt	time	#Opt
B-R1	234.40	206.78	10	15.16	10	0.28	10
B-R2	226.90	270.65	10	18.00	10	0.29	10
B-R3	205.00	283.22	10	16.03	10	0.29	10
B-R4	221.78	294.64	10	21.38	10	0.29	10
B-R5	235.11	248.61	10	18.71	10	0.91	10
B-R6	239.22	190.36	10	13.21	10	1.01	10
B-R7	271.30	210.80	10	17.68	10	0.32	10
B-R8	225.10	240.70	10	12.86	10	0.31	10
B-R9	209.50	267.22	10	14.81	10	0.40	10
B-R10	205.44	290.80	10	17.86	10	0.40	10
B-R11	231.44	276.19	10	19.87	10	0.80	10
B-R12	235.44	233.95	10	19.65	10	0.84	10
B-R13	233.40	180.45	10	12.75	10	0.45	10
B-R14	229.40	241.53	10	15.89	10	0.47	10
B-R15	221.90	297.48	10	18.77	10	1.01	10
B-R16	201.56	291.08	10	16.49	10	1.16	10
B-R17	230.89	293.31	10	21.60	10	0.81	10
B-R18	252.67	241.53	10	19.13	10	0.82	10
Mean		**253.30**	**10**	**17.21**	**10**	**0.60**	**10**

Table 9
Analysis of the results obtained by applying the CCDA heuristic to the grid instances of set A.

Name	ff Opt	CCDA		
		gap	time	#Opt
A-G1	6203.40	0.04%	0.04	8
A-G2	6204.40	0.14%	0.04	7
A-G3	9670.40	0.13%	0.09	5
A-G4	9670.80	0.11%	0.09	6
A-G5	15448.10	0.03%	0.31	4
A-G6	15448.70	0.02%	0.32	5
A-G7	23876.40	0.02%	0.80	3
A-G8	23876.00	0.02%	0.80	3
A-G9	30808.10	0.04%	2.44	2
A-G10	30808.00	0.04%	2.49	3
A-G11	47639.70	0.02%	6.66	1
A-G12	47639.70	0.02%	5.67	1
Mean		**0.05%**	**1.65**	**4**

of colours k is small. It is interesting to note that RILP is more stable than the other methods; in fact, regardless of the size, kind, density and characteristics of the instance, it is always able to identify the optimum solution in a reasonable amount of time.

A detailed analysis of the heuristic approach proposed in Section 3 is reported in Tables 9 and 10 for grid instances, and in Tables 11 and 12 for random instances. These tables report, for the Colour-Constrained Dijkstra Algorithm (CCDA), the gap with respect to the RILP algorithm, the computation time and the number of optimum solutions identified. The number of admissible solutions identified is not shown because our approach can obtain a feasible solution for each instance. For reference, the fitness function value for the RILP algorithm is also reported. At the end of each table a summary row is provided.

For grid instances (Tables 9 and 10), the CCDA provides very good results; it is able to obtain optimal or near-optimal solutions with an average gap equal to 0.05% of the optimum for set A and 0.07% of the optimum for set B in less than two seconds. The results obtained for the random instances (Tables 11 and 12) are in line with the results obtained for the grid instances. In fact, the CCDA obtains solutions with an average gap equal to 0.59% of the optimum for set A and 0.13% of the optimum for set B. Even though this gap is worse than the gap obtained for the grid instances, the number of optimum solutions obtained is considerably higher. Furthermore, our heuristic approach is always able to obtain admissible solutions regardless of the size of the graph and the number of colours.

After a deep analysis, we determined that the reason for the excellent performance of the DP approach on random instances is related to the k value. For the grid instances, this value goes from more than 150 to more than 1500, while for random instances, this value goes from 4 to 7. Given that the DP algorithm performs better when the k value is small, it obtains better results on random scenarios. Therefore, this also demonstrates that our algorithms, including both the exact and heuristic algorithms, are stable in terms of the quality of the results and the computation time, regardless of the size of the graph, the number of colours and the k value.

Table 10
Analysis of the results obtained by applying the CCDA heuristic to the grid instances of set B.

Name	ff Opt	CCDA		
		gap	time	#Opt
B-G1	6197.50	0.11%	0.02	3
B-G2	6199.60	0.15%	0.03	3
B-G3	9665.40	0.21%	0.06	2
B-G4	9665.30	0.13%	0.07	2
B-G5	15444.40	0.07%	0.28	–
B-G6	15444.50	0.02%	0.23	2
B-G7	23874.60	0.04%	0.64	–
B-G8	23875.20	0.05%	0.79	–
B-G9	30806.30	0.02%	1.75	–
B-G10	30805.80	0.02%	1.88	1
B-G11	47638.80	0.04%	7.00	–
B-G12	47639.00	0.03%	5.68	1
Mean		**0.07%**	**1.54**	**1.17**

Table 11
Analysis of the results obtained by applying the CCDA heuristic to the random instances of set A. The scenarios in which an instance does not allow admissible solutions for the given k value are shown in bold.

Name	ff Opt	CCDA		
		gap	time	#Opt
A-R1	250.18	0.29%	5.07	9
A-R2	**210.80**	**0.00%**	**3.87**	**9**
A-R3	**187.80**	**0.00%**	**3.79**	**9**
A-R4	**187.80**	**0.00%**	**3.79**	**9**
A-R5	179.55	1.72%	4.25	7
A-R6	179.55	1.72%	4.41	7
A-R7	232.91	0.00%	7.74	10
A-R8	**224.80**	**0.44%**	**5.91**	**9**
A-R9	195.27	0.33%	7.53	7
A-R10	195.27	0.33%	7.53	7
A-R11	164.45	0.06%	4.39	9
A-R12	164.45	0.06%	4.43	9
A-R13	242.18	0.45%	11.21	7
A-R14	242.18	0.45%	11.05	7
A-R15	209.73	0.17%	12.53	9
A-R16	209.73	0.17%	12.41	9
A-R17	164.36	2.21%	10.08	8
A-R18	164.36	2.21%	10.11	8
Mean		**0.59%**	**7.23**	**8.28**

Table 12
Analysis of the results obtained by applying the CCDA heuristic to the random instances of set B.

Name	ff Opt	CCDA		
		gap	time	#Opt
B-R1	234.40	0.13%	9.90	10
B-R2	226.90	0.00%	10.88	10
B-R3	205.00	0.00%	10.05	10
B-R4	221.78	0.50%	13.62	10
B-R5	235.11	0.00%	12.52	10
B-R6	239.22	0.09%	7.68	10
B-R7	271.30	0.07%	10.98	9
B-R8	225.10	0.18%	7.20	9
B-R9	209.50	0.19%	8.32	9
B-R10	205.44	0.00%	11.21	9
B-R11	231.44	0.00%	11.42	10
B-R12	235.44	0.47%	10.70	10
B-R13	233.40	0.13%	7.05	8
B-R14	229.40	0.13%	10.04	8
B-R15	221.90	0.00%	11.04	9
B-R16	201.56	0.00%	10.21	8
B-R17	230.89	0.48%	13.74	10
B-R18	252.67	0.00%	13.03	10
Mean		**0.13%**	**10.53**	**9.39**

Table 13
Four random instances of set A that, with the original k value, do not admit solutions for the k-CSPP. The ‘Instance’ column refers to the instance list reported before. By increasing the k value to the value shown in column ‘min k’, it is possible to apply all the approaches proposed. All three approaches obtain the optimum, which is reported in column ‘ff’. The time required by each method is also reported.

Instance	k	min k	ff	ILP time	RILP time	CCDA time
1	5	6	272	229.10	32.43	4.54
2	5	6	221	550.57	8.29	5.64
3	5	6	221	846.51	111.55	5.65
4	5	6	298	483.71	9.65	7.23

Table 14
Reduction percentages and computation times of the reduction process for grid instances of both sets. At the end of the table, a summary row with mean values is given.

Set A	%Reduct	Time	Set B	%Reduct	Time
A-G1	96.88	0.13	B-G1	97.47	0.11
A-G2	95.55	0.07	B-G2	96.85	0.08
A-G3	96.66	0.10	B-G3	96.39	0.11
A-G4	96.85	0.11	B-G4	97.29	0.11
A-G5	98.93	0.32	B-G5	98.68	0.33
A-G6	98.91	0.32	B-G6	98.97	0.32
A-G7	99.15	0.91	B-G7	98.99	0.94
A-G8	99.15	0.73	B-G8	98.86	0.79
A-G9	99.21	1.79	B-G9	99.45	1.64
A-G10	99.23	1.68	B-G10	99.47	1.66
A-G11	99.53	3.89	B-G11	99.34	3.99
A-G12	99.53	3.71	B-G12	99.47	3.90
Mean	**98.30**	**1.15**	Mean	**98.43**	**1.16**

Table 15
Reduction percentages and computation times of the reduction process for random instances of both sets. At the end of the table, a summary row with mean values is given.

Set A	%Reduct	Time	Set B	%Reduct	Time
A-R1	89.25	2.67	B-R1	99.21	2.68
A-R2	99.93	2.65	B-R2	99.21	2.69
A-R3	99.95	3.17	B-R3	99.76	3.16
A-R4	99.95	3.18	B-R4	99.76	3.15
A-R5	99.69	3.48	B-R5	92.43	3.54
A-R6	99.70	3.51	B-R6	92.43	3.53
A-R7	98.05	4.35	B-R7	99.84	4.52
A-R8	99.89	4.56	B-R8	99.94	4.54
A-R9	99.88	5.18	B-R9	99.90	5.39
A-R10	99.88	5.28	B-R10	99.90	5.38
A-R11	99.92	5.80	B-R11	97.39	6.17
A-R12	99.94	5.87	B-R12	97.39	5.99
A-R13	99.96	6.58	B-R13	99.75	7.24
A-R14	99.92	6.85	B-R14	99.75	6.88
A-R15	99.83	8.22	B-R15	98.51	8.12
A-R16	99.86	8.22	B-R16	96.73	8.18
A-R17	99.96	8.59	B-R17	98.89	9.33
A-R18	99.81	8.82	B-R18	98.89	8.98
Mean	**99.19**	**5.39**	Mean	**98.32**	**5.53**

6.1.1. *Unfeasible solutions*

After an analysis of the performed tests, we noticed that among the random instances of set A, there are four instances for which it was not possible to identify an admissible solution. These cases are shown in bold in Tables 7 and 11. In the two papers of Ferone et al. (2019), Ferone et al. (2020), if we look at random instances of set A, there are four cases for which neither optimal nor feasible solutions were identified. These instances are given below:

- 1 = Random_75000x75000_150000_27003;

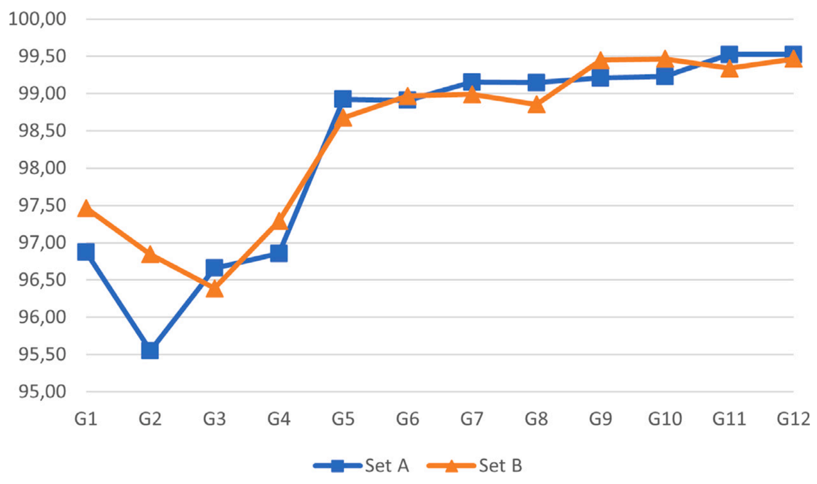


Fig. 5. Reduction percentage for grid instances. In this figure, each value represents the mean reduction percentage of ten instances of the same size, generated with different seeds. The values for the grid instances of both set A and set B are higher than 95%. Furthermore, after an initial decrease, this percentage increases as the instance size increases, achieving values close to 99.5%.

- 2 = Random_75000x1125000_168750_27006;
- 3 = Random_75000x1125000_225000_27006;
- 4 = Random_100000x1000000_200000_27014.

Given that these are the only instances for which our approaches could not obtain optimal or admissible solutions, we performed further analyses to understand the reasons behind this problem. In particular, we implemented a simplified version, for one single path, of the model proposed by Yuan et al. (2005). Their model is able to identify the minimum number of colours that must be used to define several paths between a set of source nodes and a set of destination nodes. We simplified the proposed model for a single path and tried to use this formulation to determine the minimum number of colours required to identify a path in the given instances. We implemented the model in CPLEX, relaxing the integer constraints of the model, and then we executed it. The results, shown in Table 13, highlight that for each instance, at least one more colour is required to obtain a feasible solution (the values reported in the table were rounded to the next integer value). Next, we modified the instances, increasing the k colour value of 1, and executed our approaches again. For these new instances, our approaches were able to identify the optimal solution in all cases, as shown in Table 13.

6.2. Reduction analysis

In this section, we propose two further analyses to study the GRA and determine how its application could modify the input graph. First, we analysed the size of the graph G and the percentage of nodes and edges removed after the application of the reduction approach proposed in Section 4. As can be seen in Tables 14 and 15 and in Figs. 5 and 6, respectively, for grid and random instances, the percentage of the graph removed goes from 95% to 99% for grid instances and from 89% to 99% for random instances. The effectiveness of this GRA is due to the effectiveness of the CCDA, which, as shown previously in this section, obtains optimal or near-optimal solutions in all cases. As can be seen, for grid instances, the larger the instance, the higher the reduction percentage; this is probably because the larger the graph, the higher the number of nodes and edges that are too far from the optimal path to be part of a good solution. Furthermore, the high quality of solutions provided by the CCDA translates into a threshold for the GRA that removes most of the graph. As shown in Fig. 5, for grid instances, the reduction percentage increases linearly with respect to the size of the graph, while the random instances (Fig. 6) have a more random behaviour, as expected. Therefore, for the random instances of both set A and set B, the reduction is globally very high and stable, except for some boundary cases.

The second analysis performed on the GRA takes into account the number of residual edges per colour remaining in the graph after the reduction process. This analysis is interesting because it allows us to understand not only the new size of the graph but also the distribution of the edges according to their colours. Before the reduction process, the distribution of edges per colour is uniform between instances of the same set. An effective reduction process, like the one proposed in this work, removes most of the edges and colours; the remaining colours are each only associated with a few edges. This means that on the reduced graphs, it is quite easy to identify the optimal solutions.

As can be seen in Figs. 7 and 8, most of the colours for both random and grid instances were removed because they do not have any remaining edges; however, if there are any remaining edges for a

Table 16
The set of grid instances, the complete names of the grid instances and the names used in Section 6 for reference.

Set	Complete name	Name
Grid A	Grid_100 × 100_5940	A-G1
	Grid_100 × 100_7920	A-G2
	Grid_100 × 200_11910	A-G3
	Grid_100 × 200_15880	A-G4
	Grid_250 × 250_37350	A-G5
	Grid_250 × 250_49800	A-G6
	Grid_250 × 500_74775	A-G7
	Grid_250 × 500_99700	A-G8
	Grid_500 × 500_149700	A-G9
	Grid_500 × 500_199600	A-G10
	Grid_500 × 1000_299550	A-G11
	Grid_500 × 1000_399400	A-G12
Grid B	Grid_100 × 100_396	B-G1
	Grid_100 × 100_792	B-G2
	Grid_100 × 200_794	B-G3
	Grid_100 × 200_1588	B-G4
	Grid_250 × 250_2490	B-G5
	Grid_250 × 250_4980	B-G6
	Grid_250 × 500_4985	B-G7
	Grid_250 × 500_9970	B-G8
	Grid_500 × 500_9980	B-G9
	Grid_500 × 500_19960	B-G10
	Grid_500 × 1000_19970	B-G11
	Grid_500 × 1000_39940	B-G12

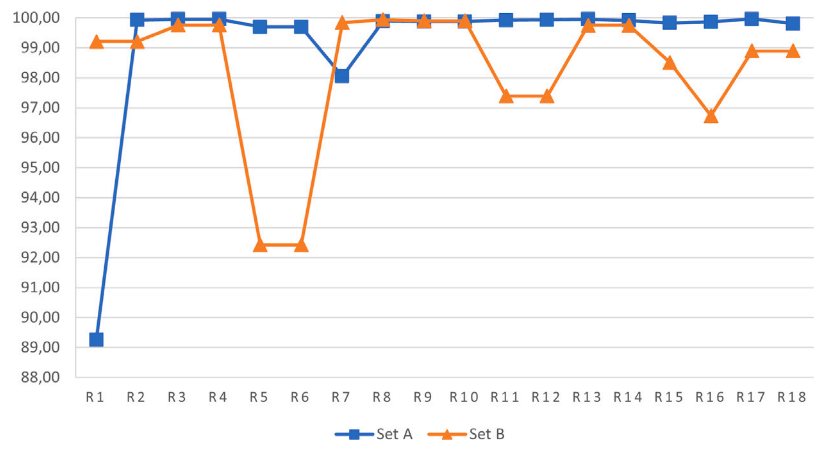


Fig. 6. Reduction percentage for random instances. In this figure, each value represents the mean reduction percentage of ten instances of the same size, generated with different seeds. The values for the random instances of both set A and set B are higher than 89%. Furthermore, while for set A this percentage is very close to 99%, excluding the instance R1, for set B, it has a more random behaviour, as expected considering the instance type.

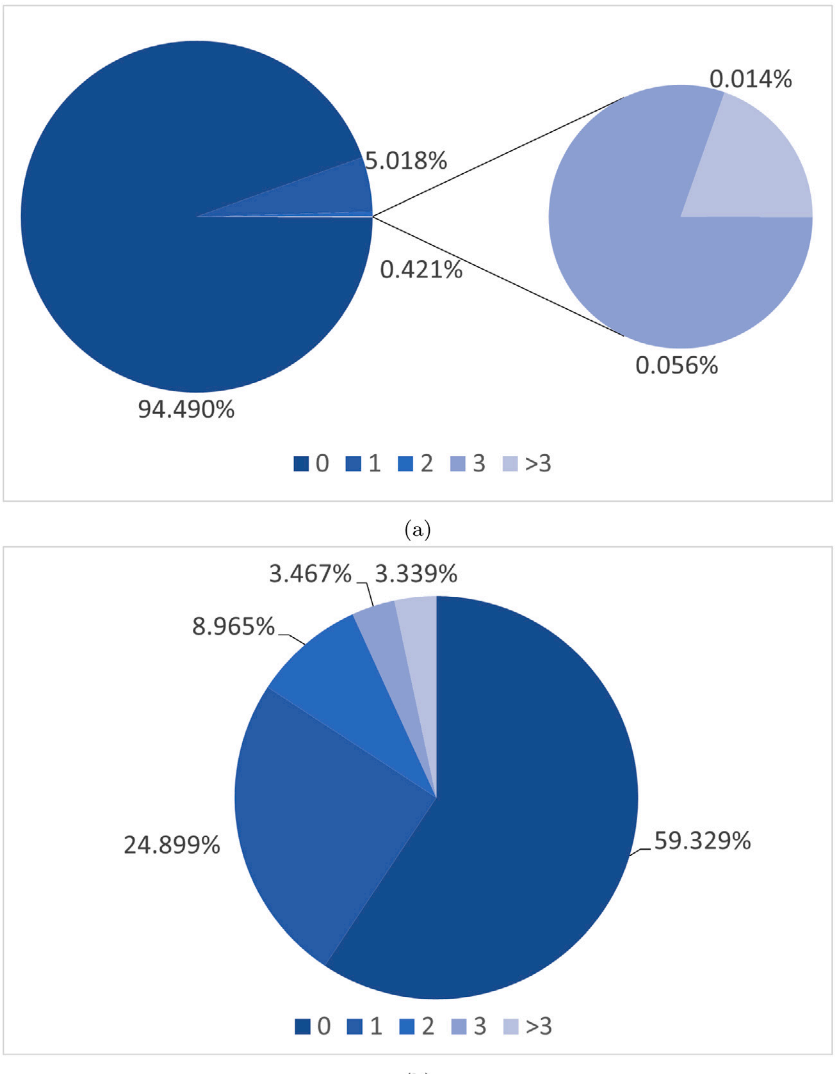
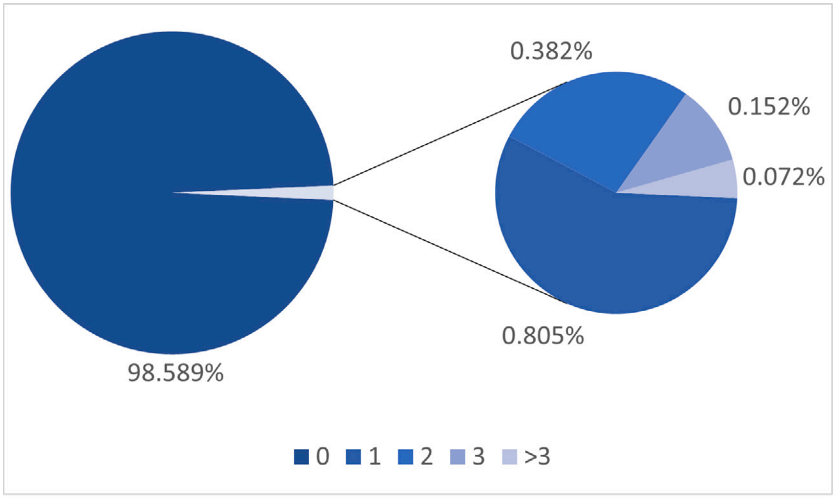
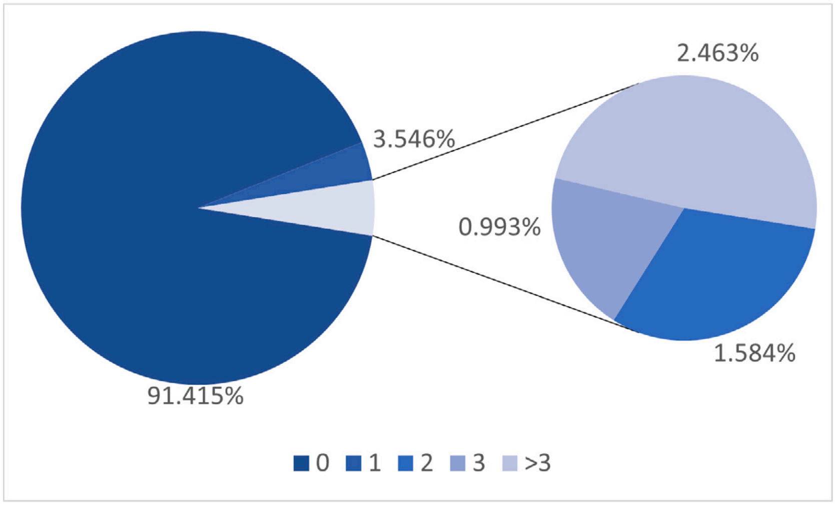


Fig. 7. The number of residual edges per colour after the execution of the reduction process for grid instances of (a) set A and (b) set B. In particular, in (a), the chart indicates that, after the reduction, most of the colours (94.490%) have 0 edges left, while 5.018% have 1 edge left and so on.



(a)



(b)

Fig. 8. The number of residual edges per colour after the execution of the reduction process for random instances of (a) set A and (b) set B. In particular, in (a), the chart indicate that, after the reduction, most of the colours (98.589%) have 0 edges left, while 0.805% have 1 edge left and so on.

given colour, the number of edges is small and almost always lower than 3. These figures show in blue the percentage of colours removed; the other percentages represent the percentage of colours that, after the reduction, have one, two, three or more than three edges remaining. It is clear that the percentages decrease as the number of edges per colour increases. The percentage for more than three colours is low compared to the rest of the percentages; this means that the reduction process proposed in this work considerably reduces the complexity of the instances, allowing the application of exact approaches like the one proposed in Section 5. The main exceptions are the grid instances of set B. In some of these instances, the reduction percentage is a bit lower, and thus, the number of edges per colour is higher. Therefore, even in this case, the effectiveness of the GRA is clear. The Appendix includes tables containing detailed information about this analysis.

7. Conclusions

In this paper, we analysed the k-Colour Shortest Path Problem. The k-CSPP is defined on a weighted edge-coloured graph, where the maximum number of different colours that can be used is fixed to be k. It is a variant of the Shortest Path Problem that consists of finding a shortest path for which the maximum number of different useable colours is defined *a priori*. The k-CSPP can be used to model several real-world scenarios, in particular in the field of network optimisation; it is able to improve the reliability of networks while reducing the connection cost between nodes.

To solve this problem, defined in Ferone et al. (2019), we first propose a heuristic approach, namely Colour-Constrained Dijkstra Algorithm (CCDA), that, in a small amount of time, provides very good solutions, regardless of the size of the instances. We propose a graph reduction technique, called the GRA, that is able to remove on average more than 90% of the nodes and edges from the input graph, drastically reducing the size of the instances. Finally, we present an exact approach, called Reduced ILP (RILP), that, taking advantage of the heuristic and the Graph Reduction Algorithm, can provide optimal solutions in a reasonable amount of time, even for large instances.

Several tests were performed to verify the effectiveness of the proposed approaches. First, we compared our algorithms with state-of-the-art methods using the benchmark instances proposed in Ferone et al. (2020). Furthermore, we performed other tests to understand the behaviour of our approaches for different kinds of instances. We analysed how the reduction process affects the instances in terms of the residual nodes and residual edges per colour.

Starting from the approach proposed in this paper, for future research, it may be interesting to work on the definition of new heuristic or meta-heuristic approaches such as, for example, a combination of our

Table 17
The set of random instances, the complete names of the random instances and the names used in Section 6 for reference.

Set	Complete name	Name
Random A	Random_75000 × 750000_112500	A-R1
	Random_75000 × 750000_150000	A-R2
	Random_75000 × 1125000_168750	A-R3
	Random_75000 × 1125000_225000	A-R4
	Random_75000 × 1500000_225000	A-R5
	Random_75000 × 1500000_300000	A-R6
	Random_100000 × 1000000_150000	A-R7
	Random_100000 × 1000000_200000	A-R8
	Random_100000 × 1500000_225000	A-R9
	Random_100000 × 1500000_300000	A-R10
	Random_100000 × 2000000_300000	A-R11
	Random_100000 × 2000000_400000	A-R12
	Random_125000 × 1250000_187500	A-R13
	Random_125000 × 1250000_250000	A-R14
	Random_125000 × 1875000_281250	A-R15
	Random_125000 × 1875000_375000	A-R16
	Random_125000 × 2500000_375000	A-R17
	Random_125000 × 2500000_500000	A-R18
Random B	Random_75000 × 750000_07500	B-R1
	Random_75000 × 750000_15000	B-R2
	Random_75000 × 1125000_11250	B-R3
	Random_75000 × 1125000_22500	B-R4
	Random_75000 × 1500000_15000	B-R5
	Random_75000 × 1500000_30000	B-R6
	Random_100000 × 1000000_10000	B-R7
	Random_100000 × 1000000_20000	B-R8
	Random_100000 × 1500000_15000	B-R9
	Random_100000 × 1500000_30000	B-R10
	Random_100000 × 2000000_20000	B-R11
	Random_100000 × 2000000_40000	B-R12
	Random_125000 × 1250000_12500	B-R13
	Random_125000 × 1250000_25000	B-R14
	Random_125000 × 1875000_18750	B-R15
	Random_125000 × 1875000_37500	B-R16
	Random_125000 × 2500000_25000	B-R17
	Random_125000 × 2500000_50000	B-R18

Table 18
Percentage of colours that have 0, 1, 2, 3 or more that 3 residual edges after the GRA is applied for grid instances.

Instances	$\|C\|$	0	1	2	3	>3
A-G1	5940	88.859%	10.281%	0.806%	0.049%	0.005%
A-G2	7920	87.674%	9.994%	1.806%	0.404%	0.122%
A-G3	11910	87.301%	11.084%	1.405%	0.176%	0.034%
A-G4	15880	90.800%	8.494%	0.662%	0.039%	0.004%
A-G5	37350	96.076%	3.833%	0.088%	0.003%	0.000%
A-G6	49800	97.015%	2.933%	0.051%	0.001%	0.000%
A-G7	74775	96.973%	2.974%	0.052%	0.001%	0.000%
A-G8	99700	97.703%	2.261%	0.035%	0.001%	0.000%
A-G9	149700	96.824%	3.093%	0.081%	0.002%	0.000%
A-G10	199600	97.697%	2.259%	0.044%	0.001%	0.000%
A-G11	299550	98.265%	1.718%	0.017%	0.000%	0.000%
A-G12	399400	98.696%	1.294%	0.010%	0.000%	0.000%
Mean	**112627.08**	**94.490%**	**5.018%**	**0.421%**	**0.056%**	**0.014%**
B-G1	396	25.480%	33.258%	23.283%	10.783%	7.197%
B-G2	792	42.917%	33.093%	14.558%	6.035%	3.396%
B-G3	794	23.186%	25.088%	16.788%	10.579%	24.358%
B-G4	1588	48.558%	31.115%	13.199%	5.038%	2.091%
B-G5	2490	49.297%	32.787%	12.173%	4.096%	1.647%
B-G6	4980	75.811%	20.990%	2.890%	0.285%	0.024%
B-G7	4985	57.482%	31.045%	9.194%	1.864%	0.415%
B-G8	9970	73.224%	21.530%	4.323%	0.777%	0.146%
B-G9	9980	74.151%	21.974%	3.463%	0.381%	0.031%
B-G10	19960	86.555%	12.449%	0.952%	0.043%	0.002%
B-G11	19970	69.418%	22.541%	5.627%	1.659%	0.755%
B-G12	39940	85.872%	12.924%	1.128%	0.069%	0.007%
Mean	**9653.75**	**59.329%**	**24.899%**	**8.965%**	**3.467%**	**3.339%**

Table 19
Percentage of colours that have 0, 1, 2, 3 or more that 3 residual edges after the GRA is applied for random instances.

Instances	$\|C\|$	0	1	2	3	>3
A-R1	112500	79.402%	10.456%	6.210%	2.644%	1.289%
A-R2	150000	99.959%	0.041%	0.000%	0.000%	0.000%
A-R3	168750	99.976%	0.024%	0.000%	0.000%	0.000%
A-R4	225000	99.982%	0.018%	0.000%	0.000%	0.000%
A-R5	225000	99.836%	0.163%	0.001%	0.000%	0.000%
A-R6	300000	99.877%	0.123%	0.000%	0.000%	0.000%
A-R7	150000	96.249%	2.980%	0.661%	0.098%	0.013%
A-R8	200000	99.963%	0.037%	0.000%	0.000%	0.000%
A-R9	225000	99.921%	0.079%	0.000%	0.000%	0.000%
A-R10	300000	99.941%	0.059%	0.000%	0.000%	0.000%
A-R11	300000	99.980%	0.020%	0.000%	0.000%	0.000%
A-R12	400000	99.985%	0.015%	0.000%	0.000%	0.000%
A-R13	187500	99.932%	0.068%	0.000%	0.000%	0.000%
A-R14	250000	99.949%	0.051%	0.000%	0.000%	0.000%
A-R15	281250	99.917%	0.083%	0.000%	0.000%	0.000%
A-R16	375000	99.937%	0.063%	0.000%	0.000%	0.000%
A-R17	375000	99.883%	0.117%	0.001%	0.000%	0.000%
A-R18	500000	99.912%	0.088%	0.000%	0.000%	0.000%
Mean	**262500**	**98.589%**	**0.805%**	**0.382%**	**0.152%**	**0.072%**
B-R1	7500	89.877%	7.353%	2.128%	0.536%	0.105%
B-R2	15000	94.124%	5.032%	0.745%	0.093%	0.006%
B-R3	11250	97.663%	2.138%	0.188%	0.012%	0.000%
B-R4	22500	98.779%	1.170%	0.050%	0.001%	0.000%
B-R5	15000	75.343%	8.757%	3.615%	1.569%	10.716%
B-R6	30000	80.872%	6.906%	1.749%	0.392%	10.081%
B-R7	10000	98.003%	1.886%	0.109%	0.002%	0.000%
B-R8	20000	99.664%	0.335%	0.002%	0.000%	0.000%
B-R9	15000	99.163%	0.829%	0.008%	0.000%	0.000%
B-R10	30000	99.581%	0.415%	0.004%	0.000%	0.000%
B-R11	20000	80.435%	3.876%	3.824%	3.552%	8.314%
B-R12	40000	84.011%	6.488%	4.612%	2.712%	2.177%
B-R13	12500	96.801%	3.042%	0.154%	0.003%	0.000%
B-R14	25000	98.361%	1.599%	0.039%	0.001%	0.000%
B-R15	18750	88.728%	2.337%	1.634%	2.026%	5.276%
B-R16	37500	81.288%	4.806%	4.827%	3.966%	5.113%
B-R17	25000	90.161%	2.855%	2.649%	2.124%	2.210%
B-R18	50000	92.611%	4.000%	2.176%	0.879%	0.334%
Mean	**22500**	**91.415%**	**3.546%**	**1.584%**	**0.993%**	**2.463%**

reduction process and the DP approach. The effectiveness of the reduction technique introduced in this paper encourages us to investigate the applicability of this technique to other path identification problems.

Compliance with ethical standards

Research involving human participants and/or animals

This article does not contain any studies with human participants or animals performed by any of the authors.

CRediT authorship contribution statement

Carmine Cerrone: Conceptualization, Methodology, Validation, Writing – review & editing, Supervision. **Davide Donato Russo:** Conceptualization, Methodology, Software, Data curation, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

See Tables 16–19.

References

Avella, P., Boccia, M., Sforza, A., 2004. Resource constrained shortest path problems in path planning for fleet management. J. Math. Model. Algorithms 3 (1), 1–17.

Beasley, J.E., Christofides, N., 1989. An algorithm for the resource constrained shortest path problem. Networks 19 (4), 379–394.

Bilge, Y.C., Çağatay, D., Genç, B., Sarı, M., Akcan, H., Evrendilek, C., 2015. All colors shortest path problem. arXiv Preprint, arXiv:1507.06865.

Broersma, H., Li, X., Woeginger, G.J., Zhang, S., 2005. Paths and cycles in colored graphs. Australas. J. Combin. 31, 299–312.

Captivo, M.E., Clímaco, J.C., Pascoal, M.M., 2009. A mixed integer linear formulation for the minimum label spanning tree problem. Comput. Oper. Res. 36 (11), 3082–3085. http://dx.doi.org/10.1016/j.cor.2009.02.003.

Carrabs, F., Cerrone, C., Cerulli, R., Silvestri, S., 2018a. On the complexity of rainbow spanning forest problem. Optim. Lett. 12 (3), 443–454.

Carrabs, F., Cerrone, C., Cerulli, R., Silvestri, S., 2018b. The rainbow spanning forest problem. Soft Comput. 22 (8), 2765–2776.

Carrabs, F., Cerulli, R., Felici, G., Singh, G., 2019. Exact approaches for the orderly colored longest path problem: Performance comparison. Comput. Oper. Res. 101, 275–284.

Carrabs, F., Cerulli, R., Pentangelo, R., Raiconi, A., 2018c. A two-level metaheuristic for the all colors shortest path problem. Comput. Optim. Appl. 71 (2), 525–551.

Cerrone, C., D'Ambrosio, C., Raiconi, A., 2019. Heuristics for the strong generalized minimum label spanning tree problem. Networks 74 (2), 148–160.

Cerulli, R., Fink, A., Gentili, M., Raiconi, A., 2014. The k-labeled spanning forest problem. Proc.-Soc. Behav. Sci. 108, 153–163.

Cerulli, R., Fink, A., Gentili, M., Voß, S., 2006. Extensions of the minimum labelling spanning tree problem. J. Telecommun. Inform. Technol. 39–45.

Consoli, S., Moreno Perez, J.A., Mladenović, N., 2017. Comparison of metaheuristics for the k-labeled spanning forest problem. Int. Trans. Oper. Res. 24 (3), 559–582.

da Silva, T.G., Queiroga, E., Ochi, L.S., dos Anjos Formiga Cabral, L., Gueye, S., Michelon, P., 2019. A hybrid metaheuristic for the minimum labeling spanning tree problem. European J. Oper. Res. 274 (1), 22–34. http://dx.doi.org/10.1016/j.ejor.2018.09.044.

Di Puglia Pugliese, L., Guerriero, F., Poss, M., 2019. The resource constrained shortest path problem with uncertain data: A robust formulation and optimal solution approach. Comput. Oper. Res. 107, 140–155. http://dx.doi.org/10.1016/j.cor.2019.03.010.

Dijkstra, E.W., et al., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271.

Ferone, D., Festa, P., Fugaro, S., Pastore, T., 2020. A dynamic programming algorithm for solving the K-color shortest path problem. Optim. Lett. 1–20.

Ferone, D., Festa, P., Guerriero, F., Laganà, D., 2016. The constrained shortest path tour problem. Comput. Oper. Res. 74, 64–77. http://dx.doi.org/10.1016/j.cor.2016.04.002.

Ferone, D., Festa, P., Pastore, T., 2019. The k-color shortest path problem. In: Advances in Optimization and Decision Science for Society, Services and Enterprises. Springer, pp. 367–376.

Horváth, M., Kis, T., 2016. Solving resource constrained shortest path problems with LP-based methods. Comput. Oper. Res. 73, 150–164. http://dx.doi.org/10.1016/j.cor.2016.04.013.

Irnich, S., Desaulniers, G., 2005. Shortest path problems with resource constraints. In: Column Generation. Springer, pp. 33–65.

Madkour, A., Aref, W.G., Rehman, F.U., Rahman, M.A., Basalamah, S., 2017. A survey of shortest-path algorithms. arXiv Preprint, arXiv:1705.02044.

Marinakis, Y., Migdalas, A., Sifaleras, A., 2017. A hybrid particle swarm optimization – Variable neighborhood search algorithm for constrained shortest path problems. European J. Oper. Res. 261 (3), 819–834. http://dx.doi.org/10.1016/j.ejor.2017.03.031.

Naji-Azimi, Z., Salari, M., Golden, B., Raghavan, S., Toth, P., 2010. Variable neighborhood search for the cost constrained minimum label spanning tree and label constrained minimum spanning tree problems. Comput. Oper. Res. 37 (11), 1952–1964. http://dx.doi.org/10.1016/j.cor.2009.12.013.

Pugliese, L.D.P., Guerriero, F., 2013. A survey of resource constrained shortest path problems: Exact solution approaches. Networks 62 (3), 183–200.

Smith, O.J., Boland, N., Waterer, H., 2012. Solving shortest path problems with a weight constraint and replenishment arcs. Comput. Oper. Res. 39 (5), 964–984. http://dx.doi.org/10.1016/j.cor.2011.07.017.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., Irnich, S., 2017. Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. European J. Oper. Res. 261 (2), 530–539. http://dx.doi.org/10.1016/j.ejor.2017.03.017.

Yuan, S., Varma, S., Jue, J.P., 2005. Minimum-color path problems for reliability in mesh networks. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 4. IEEE, pp. 2658–2669.

Zhu, X., Wilhelm, W.E., 2013. Implementation of a three-stage approach for the dynamic resource-constrained shortest-path sub-problem in branch-and-price. Comput. Oper. Res. 40 (1), 385–394. http://dx.doi.org/10.1016/j.cor.2012.07.007.