Optimization Letters (2021) 15:1973–1992
https://doi.org/10.1007/s11590-020-01659-z

ORIGINAL PAPER


# A dynamic programming algorithm for solving the *k*-Color Shortest Path Problem

**Daniele Ferone[1] · Paola Festa[2] · Serena Fugaro[2] · Tommaso Pastore[3]**

Received: 29 January 2020 / Accepted: 21 October 2020 / Published online: 7 November 2020


**Abstract**
Several variants of the classical Constrained Shortest Path Problem have been presented in the literature so far. One of the most recent is the *k-Color Shortest Path Problem* (*k*-CSPP), that arises in the field of transmission networks design. The problem is formulated on a weighted edge-colored graph and the use of the colors as edge labels allows to take into account the matter of path reliability while optimizing its cost. In this work, we propose a dynamic programming algorithm and compare its performances with two solution approaches: a Branch and Bound technique proposed by the authors in their previous paper and the solution of the mathematical model obtained with CPLEX solver. The results gathered in the numerical validation evidenced how the dynamic programming algorithm vastly outperformed previous approaches.

**Keywords** Edge-colored network · Constrained Shortest Path · Dynamic programming · Network optimization

✉ Daniele Ferone
daniele.ferone@unical.it

Paola Festa
paola.festa@unina.it

Serena Fugaro
serena.fugaro@unina.it

Tommaso Pastore
tommaso.pastore@unina.it

[1] Department of Mechanical, Energy and Management Engineering, University of Calabria, Rende, Italy

[2] Department of Mathematics and Applications, University of Naples "Federico II", Naples, Italy

[3] Department of Structures for Engineering and Architecture, University of Naples "Federico II", Naples, Italy

## 1 Introduction

In the past few decades, communication networks have swiftly become a key infrastructure in any modern society. Indeed, nowadays they play an essential role in the organization of almost every human activity. As a consequence, a growing stream of research is focused on guaranteeing network reliability and robust service. At this purpose, in the field of planning and optimization of transmission networks, recent contributions dealt with the use of path protection schemes to ensure network operation in case of single link failures [14]. In particular, in path protection two link-disjoint paths sharing the same origin and destination are stored, namely *primary* and *back-up*, such that, whenever the connection in the primary path fails, the back-up can be used to prevent traffic loss.

To broaden the applicability of the traffic optimization approach, Yuan et al. [15] considered the simultaneous outage of several links in the network. Specifically, their optimization framework took into account a network in which a single happening could cause a synchronous breakdown of multiple physical links.

This scenario is related, for example, to the nature of wavelength division multiplexing networks, in which it is customary to bundle multiple fiber links in the same conduit, so that the consequences of damages to it would affect all the links there bundled. To properly model this network topology, the authors considered an *edge-colored graph*—in which links that could be damaged by a single event are modeled with arcs of the same color (see Fig. 1)—and solved the failure minimization problem as a minimum-color path problem.

In fact, with the hypothesis of mutually independent and equi-probable failure events—with probability $p \in [0, 1]$—minimizing the number $k$ of different colors traversed in the path would consequently maximize its reliability, namely $(1 - p)^k$.

In particular, the above mentioned example collocates the problem of interest—the guaranty of reliability and robustness—in the framework of edge-colored networks. Actually, there is a growing interest of the scientific literature for these networks due to their ability to represent different interrelations between their nodes [2,3].

More specifically, this paper addresses a recent problem, originally proposed in [7], named *k-Color Shortest Path Problem* ($k$-CSPP), in which a weighted edge-colored network is considered. The objective is to find a shortest path between a *source* node $s$ and a *target* node $t$, with a constraint on the maximum number $k$ of different colors that the path can traverse. With reference to the path reliability scenario, we can observe how this problem takes into account the risk adversity while optimizing the length of

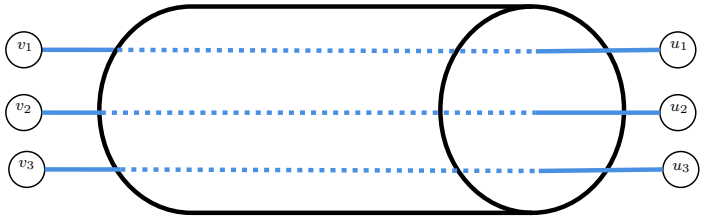


**Fig. 1** Multiple fibers bundled in the same conduit modelled as arcs sharing the same color

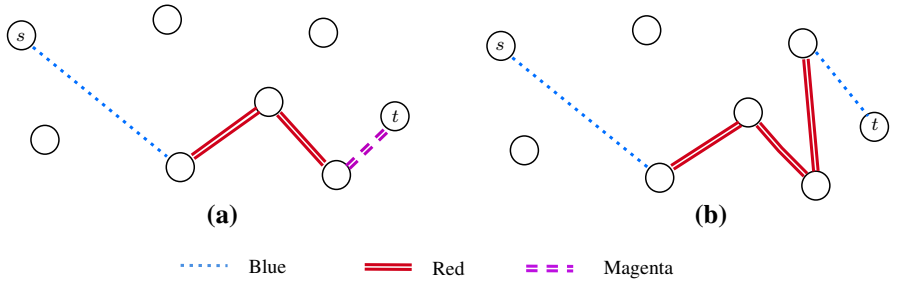


**Fig. 2** Example of two different edge-colored paths. The first characterized by 4 edges and 3 different colors, and the second by 5 edges and 2 colors

the path. Considering as an example the scenario depicted in Fig. 2—with $k = 2$—the path from $s$ to $t$ composed by four edges and three colors in Fig. 2a is deemed infeasible, while the right-hand side path (Fig. 2b) made up by two different colors and five edges is preferable even if potentially longer.

The $k$-CSPP has already been tackled by means of a Branch and Bound algorithm; on the other hand the dynamic programming framework has been widely used to deal with Resource Constrained Shortest Path Problems [4,5]. With this in mind, the main contribution of the present work is the design of a dynamic programming algorithm based on a path-labeling approach and an $A^*$-like exploration strategy.

The paper is organized as follows. Section 2 briefly outlines the problem. A detailed description of the devised solution approach is given in Sect. 3. Section 4 summarizes the benchmark of instances used to test the algorithms and the experiments set up. Moreover, in this section, the computational result and a comparative analysis of the performances of our algorithm with respect to the CPLEX solver and the Branch and Bound approach introduced in [7] are presented. Section 5 is devoted to conclusions and future works.

## 2 Problem description

To formally describe the problem, an undirected weighted and colored graph $G = (V, E, w, C)$ is introduced, where $w : E \longrightarrow \mathbb{R}_0^+$ is a function that assigns a non-negative distance $w_{ij}$ to each edge $[i, j] \in E$ and $C : E \longrightarrow \mathbb{N}$ is a labeling function that assigns a color to each $[i, j] \in E$. In the following, we will refer to $G$ as an *edge-colored graph*. Additionally, let $C(\hat{E})$ be the set of different colors appearing in any subset $\hat{E} \subseteq E$, and $c(\hat{E}) = \left|C(\hat{E})\right|$; thus $c(E)$ is the total number of colors used to label the edges of $G$. Moreover, letting $E_h, \forall h \in \{1, \ldots C(E)\}$, be the set of all the edges labeled with the color $h$, the set of edges $E$ can be partitioned as $\bigcup_{h=1}^{C(E)} E_h$.

Given a source node $s$ and a target node $t, s, t \in V, s \neq t$, the $k$-Color Shortest Path Problem ($k$-CSPP) aims at finding a minimum distance path $P^*$ from $s$ to $t$, consisting of edges of at most $k$ different colors. A solution for this problem can be modelled through the introduction of a Boolean decision variable $x_{ij}$ for each edge $[i, j] \in E$

such that

$$x_{ij} = \begin{cases} 1, & \text{if } [i, j] \text{ belongs to } P^*; \\ 0, & \text{otherwise.} \end{cases}$$

Then, as done in [7], the $k$-CSPP is formally described as the following integer linear program:

$$z = \min \sum_{[i,j] \in E} w_{ij} x_{ij} \tag{1a}$$

subject to:

$$\sum_{\{j\,:\,[i,j] \in E\}} x_{ji} - \sum_{\{j\,:\,[i,j] \in E\}} x_{ij} = b_i, \qquad \forall i \in V \tag{1b}$$

$$x_{ij} \leq y_h, \quad \forall [i, j] \in E_h, h = 1, \ldots, C(E) \tag{1c}$$

$$\sum_{h=1}^{C(E)} y_h \leq k \tag{1d}$$

$$x_{ij} \in \{0, 1\}, \qquad [i, j] \in E \tag{1e}$$

$$y_h \in \{0, 1\}, \qquad \forall h = 1, \ldots, C(E) \tag{1f}$$

with $b_i = -1$ for $i = s$, $b_i = 1$ for $i = t$, and $b_i = 0$ otherwise.

The objective function (1a) minimizes the total distance of the path. Constraints (1b) are classical flow-balancing restrictions; those (1c) connect edge traversal and color selection, while constraints (1d) limit the maximum number of different colors that can be used in the solution. Finally, the Boolean nature of the decision variables is expressed by (1e) and (1f).

In [2], it is proved that finding a simple path $\bar{P}$ from $s$ to $t$ in an edge-colored graph, such that $c(\bar{P}) \leq k$ with a given $k$, is an **NP**-complete problem. As a consequence of this computational complexity result, we have the following claim:

**Lemma 1** *There does not exist a polynomial-time algorithm $\mathcal{A}$ to find a feasible solution for an arbitrary instance $\mathcal{I}$ of the $k$-CSPP, unless* $\mathbf{P} = \mathbf{NP}$.

Since approximation algorithms find feasible solutions with provable guarantees on solution quality, a polynomial-time approximation algorithm would find—if existing—a feasible solution in polynomial time. Consequently, Corollary 1 follows from Lemma 1.

**Corollary 1** *There does not exist a polynomial-time approximation algorithm for the $k$-CSPP, unless* $\mathbf{P} = \mathbf{NP}$.

## 3 Solution approach

To optimally solve the $k$-CSPP, we propose a dynamic programming algorithm (DP), whose design has been encouraged by the successful results that this family of solution framework gathered in the field of constrained shortest path problems [4,12,13].

To present our solution algorithm DP, let $P_{si}$ be a path in $G$ connecting the source $s$ with a generic node $i$, and let $L_i = (d_i, C_i, P_{si})$ be a label associated to $P_{si}$, where $d_i$ and $C_i$ are the total distance of the path and the set of different colors traversed by $P_{si}$, respectively. Moreover, let $D(i)$ define the set of all the labels $L_i$ associated with the different paths connecting $s$ to $i$.

Following the general scheme of a label correcting technique, DP explores the solution space to extend the paths under construction by analyzing the set of their labels. Given a path $P_{si}$, the result of path extension operation is a path $P_{sh}$ obtained concatenating $P_{si}$ and $[i, h] \in E \backslash P_{si}$, and denoted as $\langle P_{si}, [i, h]\rangle$. The source node $s$ is given an initial label $L_s = (0, \emptyset, \langle s\rangle)$. Then, starting from a generic label $L_i = (d_i, C_i, P_{si})$, for each node $j \in V$ such that $[i, j] \in E$, new labels $L_j$ are generated. In particular, the distance of the path $P_{sj} = \langle P_{si}, [i, j]\rangle$ is defined as $d_j = d_i + w_{ij}$, and the set of colors traversed is $C_j = C_i \cup \{C([i, j])\}$.

Once the labels are generated, their evaluation is mainly based upon two fundamental concepts: *feasibility* and *dominance*.

**Definition 1** (*Feasibility*) A generated label $L_j$ is feasible if $\left|C_j\right| \leq k$.

**Definition 2** (*Dominance*) Given two labels $L_i$ and $\hat{L}_i$ associated with the same node $i \in V$, $L_i$ dominates $\hat{L}_i$ if the following conditions hold:

$$d_i \leq \hat{d}_i;$$
$$C_i \subseteq \hat{C}_i,$$

and at least one of such conditions is strict.

Thus, the generated labels are discarded if they are either associated with infeasible paths or dominated by other labels.

**Theorem 1** *Let $L_i$ and $\hat{L}_i$ be the labels associated to the paths $P_{si}$ and $\hat{P}_{si}$, respectively. If $L_i$ dominates $\hat{L}_i$, then, for any feasible extension $\left\langle \hat{P}_{si}, [i, h]\right\rangle$, there exists at least a feasible path $\mathcal{P}$ from $s$ to $h$ such that:*

1. *$\mathcal{P}$ is not longer than $\left\langle \hat{P}_{si}, [i, h]\right\rangle$;*
2. *$C(\mathcal{P}) \subseteq C\left(\left\langle \hat{P}_{si}, [i, h]\right\rangle\right)$.*

***Proof*** Let $\hat{\mathcal{P}} = \left\langle \hat{P}_{si}, [i, h]\right\rangle$ be a feasible extension of the path $\hat{P}_{si}$ and let $\mathcal{P}' = \langle P_{si}, [i, h]\rangle$ be the extension obtained by substituting path $\hat{P}_{si}$ with path $P_{si}$. For path $\mathcal{P}'$ two cases can occur: either it is a feasible extension too or it is an infeasible extension due to the presence of a cycle.

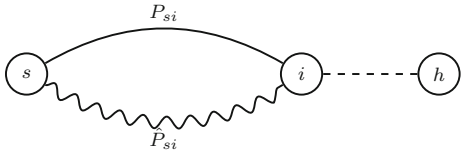


**Fig. 3** Feasible concatenation of a "dominant" path (curved) with the extension (dashed) of a "dominated" path (normal)

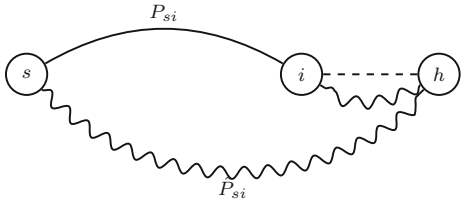


**Fig. 4** Concatenation of a "dominant" path (curved) with the extension (dashed) of a "dominated" path (normal) containing a cycle

In the former case (Fig. 3), given the dominance of $L_i$ over $\hat{L}_i$, the extension $\mathcal{P}'$ verifies conditions 1 and 2. In fact, according to Definition 2, path feasibility is preserved—$C(\mathcal{P}') = C_i \cup \{C([i,h])\} \subseteq \hat{C}_i \cup \{C([i,h])\} = C(\hat{\mathcal{P}})$—while the distance of $P_{si}$ being not greater than that of $\hat{P}_{si}$—i.e. $d_i \leq \hat{d}_i$—ensures that distance-wise $\mathcal{P}'$ is not less favorable with respect to $\hat{\mathcal{P}}$. Hence, $\mathcal{P} = \mathcal{P}'$.

In the latter case, a cycle in the extension $\mathcal{P}'$ occurs when the node $h$ is visited by the dominant path $P_{si}$, as depicted in Fig. 4.

Thus, given that $P_{sh} \subset P_{si}$ and $\hat{P}_{si} \subset \hat{\mathcal{P}}$, the dominance of $L_i$ over $\hat{L}_i$ ensures that $C(P_{sh}) \subseteq C(P_{si}) \subseteq C(\hat{P}_{si}) \subseteq C(\hat{\mathcal{P}})$ while the distance of $P_{sh}$ being not greater than the distance of $\hat{\mathcal{P}}$. As a consequence, the subpath $P_{sh}$ is a path from $s$ to $h$ which verifies conditions 1 and 2, i.e. $\mathcal{P} = P_{sh}$.

In conclusion, extending a dominated path yields to the construction of a dominated path. □

With the aim of pruning the space of solutions from those unfavourable, the dynamic programming algorithm collects in sets $D(i), \forall\, i \in V$, only feasible and non-dominated labels. It is duly noted how this approach preserves optimality, since there exists at least one optimal solution associated to a feasible and non-dominated label. In fact, if $P^*$ is an optimal path associated to a dominated label, then there exists another feasible $s$-$t$ path $P^{**}$ such that $C(P^{**}) \subseteq C(P^*)$, and $P^{**}$ is not longer than $P^*$. Since $P^*$ is optimal, $P^{**}$ and $P^*$ have the same length—in terms of distance—, and thus $P^{**}$ is an optimal solution.

The `DP` algorithm is summarized in Algorithm 1. Let $\Lambda$ be the cost of the current incumbent. Lines from 2 to 4 initialize the first label, the list of labels $L$, the lists $D(j), \forall\, j \in V$ and the cost $\Lambda$. While $L$ is not empty, the algorithm extracts a non-dominated feasible label $L_i$ from $L$ (Line 6). Then, if the total distance related to $L_i$ is less than $\Lambda$, the incumbent is updated whenever $i = t$. Otherwise, the labels of each node $j \in V$ such that $[i, j] \in E$ are generated and added to the list $L$ by means of the procedure `AddLabel` (Line 14). In particular, given a certain label $L_j$, this label-adding procedure includes $L_j$ in $L$ and $D(j)$ if it is feasible and non-dominated.

Finally, `DP` returns the best found solution corresponding to the optimal one.

```
1  Function DP(G = (V, E, d, C), s, t)
2      L_s ← (0, ∅, ⟨s⟩)
3      L ← {L_s}; D(s) ← {L_s}; D(j) ← ∅, ∀ j ∈ V, j ≠ s
4      best ← Nil; Λ ← +∞
5      while L ≠ ∅ do
6          L_i ← Extract(L); L ← L\{L_i}
7          if d_i < Λ then
8              if i = t then
9                  Λ = d_i
10                 best ← L_i
11             else
12                 foreach j ∈ V : [i, j] ∈ E do
13                     L_j ← (d_i + w_ij, C_i ∪ {C([i, j])}, ⟨P_si, [i, j]⟩)
14                     AddLabel(L, D, L_j)
15     return best
```

**Algorithm 1:** Pseudo-code of the proposed Dynamic Programming algorithm.

In a dynamic programming technique, the choice of a well-performing label extraction policy (Algorithm 1, line 6) can be a determining factor in favoring the fast convergence to good quality solutions, that combined with the pruning of dominated labels can increase the performances of the algorithm.

The following list briefly introduces some of the best-known strategies that can guide the label-extraction operations.

**Dijkstra-like rule (DR)**: the extracted label is the one with the smallest distance;

**First-In First-Out (FIFO)**: the extracted label is the one that has been in the queue for the longest time;

**Last-In First-Out (LIFO)**: the last inserted label is the first extracted;

**Small-Label-First (SLF)** [1]: the extraction is performed from the top of the list. Indeed, whenever a new label $L_j$ has to be added to $L$, its total distance $d_j$ is compared with the one $d_p$ of the currently top label $L_p$ of $L$. If $d_j < d_p$, label $L_j$ is entered at the top of $L$; otherwise $L_j$ is entered at the bottom of $L$ ;

**A***: the extracted label is the one that presents the smallest value of the sum between the distance $d_i$ and the distance of the simple shortest path from the current node $i$ to the target node $t$.

```
1  Procedure AddLabel(L, D, L_j)
2      if L_j is feasible then
3          if L_j is not dominated by any label L'_j belonging to D(j) then
4              Remove from D(j) and L all labels L'_j that are dominated by L_j
5              L ← L ∪ {L_j}
6              D(j) ← D(j) ∪ {L_j}
```

**Algorithm 2:** Pseudo-code of label-adding procedure.

***Remark 1*** Interestingly enough, we observe that the function used in the A* strategy, to evaluate the length of the path from the current node $i$ to the target node $t$, gives a lower bound on the cost of the optimal path from $i$ to $t$. In fact, that minimum cost path is computed without taking into account constraints (1d).

As reported in [11], A* finds an optimal solution whenever the length of the path from the current node to the target one is estimated with a lower bound. As a consequence, the first feasible solution found by the A* strategy is indeed optimal, thus allowing to prune all the remaining labels. □

***Remark 2*** As a last observation, we note how the computational effort related to the execution of a DP algorithm is related to the number of explored labels. Without assuming the use of a specific extraction policy, in the worst possible case the number of explored labels is equal to the number of feasible $k$-colored paths from $s$ to any node $i \in V$.

Assuming the use of $A^*$ as extraction policy, let $N(k)$ be the total number of combination without repetitions of $l$ elements of $C$, with $l = 1, \ldots, k$, i.e.

$$N(k) = \sum_{l=1}^{k} \binom{|C|}{l}. \tag{2}$$

The number of labels extracted for each node $v \in V \backslash \{s, t\}$ is bounded by $N(k)$, since for each feasible combination of colors, the $A^*$ strategy extracts only the most favorable path that dominates others characterized by the same set of colors. As a consequence of this, the number of iterations of the `DP` algorithm is bounded by $(|V| - 2) \cdot N(k)$. Each one of the operations executed in a single iteration can be carried out in $O(1)$, except for the `AddLabel` operation 14, whose complexity is linear in the size of $D(j)$. The size of $D(j)$ can be estimated by $N(k)$, since for each feasible combination of colors, only the dominating path is kept in memory. Therefore, an upper bound for the complexity of `DP` with the $A^*$ strategy is $O((|V|-2) \cdot N(k)^2)$. □

## 4 Computational experiments

In this Section, we discuss about the computational experiments we have designed to appraise the performances of `DP` when compared to two other different solution approaches, namely a Branch and Bound (B&B) algorithm described in [7], and the direct solution of the mathematical model obtained by means of the ILOG CPLEX 12.9 solver.

All the algorithms here compared were coded in C++ using the flags `-std=c++17 -O3` and compiled with `g++ 8.2`. The experiments were run on a INTEL i5-6400@2.70 GHz processor with 8GB of RAM. A time limit of 10 min has been used for each solution method.

**Table 1** Instance parameters for data-set $\mathcal{A}$

| Fully random graphs | | | | | Grid graphs | | | |
|---|---|---|---|---|---|---|---|---|
| Problem | $p$ | Nodes | Arcs | Colors | Problem | $p$ | Size | Colors |
| R1 | 0.15 | 75,000 | 750,000 | 112,500 | G1 | 0.15 | $100 \times 100$ | 5940 |
| R1 | 0.20 | 75,000 | 750,000 | 150,000 | G1 | 0.20 | $100 \times 100$ | 7920 |
| R2 | 0.15 | 75,000 | 112,500 | 168,750 | G2 | 0.15 | $100 \times 200$ | 11,910 |
| R2 | 0.20 | 75,000 | 112,500 | 225,000 | G2 | 0.20 | $100 \times 200$ | 15,880 |
| R3 | 0.15 | 75,000 | 150,000 | 225,000 | G3 | 0.15 | $250 \times 250$ | 37,350 |
| R3 | 0.20 | 75,000 | 150,000 | 300,000 | G3 | 0.20 | $250 \times 250$ | 49, 800 |
| R4 | 0.15 | 100,000 | 1,000,000 | 150,000 | G4 | 0.15 | $250 \times 500$ | 74,775 |
| R4 | 0.20 | 100,000 | 1,000,000 | 200,000 | G4 | 0.20 | $250 \times 500$ | 99,700 |
| R5 | 0.15 | 100,000 | 1,500, 000 | 225,000 | G5 | 0.15 | $500 \times 500$ | 149,700 |
| R5 | 0.20 | 100,000 | 1,500,000 | 300,000 | G5 | 0.20 | $500 \times 500$ | 199,600 |
| R6 | 0.15 | 100,000 | 2,000, 000 | 300,000 | G6 | 0.15 | $500 \times 1000$ | 299,550 |
| R6 | 0.20 | 100,000 | 2,000,000 | 400,000 | G6 | 0.20 | $500 \times 1000$ | 399,400 |
| R7 | 0.15 | 125,000 | 1,250,000 | 187,500 | | | | |
| R7 | 0.20 | 125,000 | 1,250,000 | 250,000 | | | | |
| R8 | 0.15 | 125,000 | 1,875,000 | 281,250 | | | | |
| R8 | 0.20 | 125,000 | 1,875,000 | 375,000 | | | | |
| R9 | 0.15 | 125,000 | 2,500,000 | 375,000 | | | | |
| R9 | 0.20 | 125,000 | 2,500,000 | 500,000 | | | | |

### 4.1 Test instances

The experimentation has been conducted on two data-sets, whose characteristics are summarized in Tables 1 and 2.

Explicitly, the first set of testing instances considered, namely $\mathcal{A}$, consists of the networks described in [7]. These instances were randomly generated through an adaption of the generator presented in [9], and can be divided in two classes: fully random and grid graphs.[1]

The number $m$ of edges in the fully random graphs has been selected to belong to $\{10 \cdot n, 15 \cdot n, 20 \cdot n\}$, where $n = |V|$. Moreover, the total number of colors for each instance is $p \cdot m$ with $p \in \{0.15, 0.20\}$. Finally, in order to avoid instance-triviality, the value for $k$ has been determined solving a shortest path problem on $G$. In particular, if $\pi^*$ is a shortest path connecting $s$ and $t$ in $G$, and $C^*$ is the number of different colors traversed by $\pi^*$, then $k$ is selected as $C^* - 2$.

Each combination of graph size, denoted as $\{R1, \ldots, R9, G1, \ldots, G6\}$, and number of colors characterizes a collection of similar instances. Each collection contains ten different instances of the same type. The characteristics of the data-set are summarized in Table 1.

[1] The full data-set is available on Figshare [6].

**Table 2** Instance parameters of data-set $\mathcal{B}$

| Fully random graphs | | | | | Grid graphs | | | |
|---|---|---|---|---|---|---|---|---|
| Problem | $p$ | Nodes | Arcs | Colors | Problem | $p$ | Size | Colors |
| R1 | 0.01 | 75,000 | 750,000 | 7500 | G1 | 0.01 | 100 × 100 | 396 |
| R1 | 0.02 | 75,000 | 750,000 | 15,000 | G1 | 0.02 | 100 × 100 | 792 |
| R2 | 0.01 | 75,000 | 112,500 | 11,250 | G2 | 0.01 | 100 × 200 | 794 |
| R2 | 0.02 | 75,000 | 112,500 | 22,500 | G2 | 0.02 | 100 × 200 | 1588 |
| R3 | 0.01 | 75,000 | 150,000 | 15,000 | G3 | 0.01 | 250 × 250 | 2490 |
| R3 | 0.02 | 75,000 | 150,000 | 30,000 | G3 | 0.02 | 250 × 250 | 4980 |
| R4 | 0.01 | 100,000 | 1,000,000 | 10,000 | G4 | 0.01 | 250 × 500 | 4985 |
| R4 | 0.02 | 100,000 | 1,000,000 | 20,000 | G4 | 0.02 | 250 × 500 | 9970 |
| R5 | 0.01 | 100,000 | 1,500,000 | 15,000 | G5 | 0.01 | 500 × 500 | 9980 |
| R5 | 0.02 | 100,000 | 1,500,000 | 30,000 | G5 | 0.02 | 500 × 500 | 19,960 |
| R6 | 0.01 | 100,000 | 2,000,000 | 20,000 | G6 | 0.01 | 500 × 1000 | 19,970 |
| R6 | 0.02 | 100,000 | 2,000,000 | 40,000 | G6 | 0.02 | 500 × 1000 | 39,940 |
| R7 | 0.01 | 125,000 | 1,250,000 | 1250 | | | | |
| R7 | 0.02 | 125,000 | 1,250,000 | 25,000 | | | | |
| R8 | 0.01 | 125,000 | 1,875,000 | 18,750 | | | | |
| R8 | 0.02 | 125,000 | 1,875,000 | 37,500 | | | | |
| R9 | 0.01 | 125,000 | 2,500,000 | 25,000 | | | | |
| R9 | 0.02 | 125,000 | 2,500,000 | 50,000 | | | | |

Moreover, in order to better analyze how the performances of the algorithms are affected by a variation in the number of colors, we generated a second set of instances, namely $\mathcal{B}$. This data-set replicates the graph sizes of set $\mathcal{A}$, but $p$ ranges in $\{0.01, 0.02\}$ meaning that the total number of colors in the networks has been reduced. The characteristics are summarized in Table 2.

### 4.2 Analysis of the extraction policies for DP

The analysis here presented aims to appraise how the computational performance of the proposed solution approach is affected by the different label selection policies.

At this purpose, we randomly selected 2 out of 10 instances of each type from data-set $\mathcal{A}$ and left `DP`—with one of the extraction policies, in turn—run with a time limit of 10 min. Tables 3 and 4 report, for each instance type and each extraction policy, the average running time and the number O of optimal solutions found within the time limit.

The results point out that the most performing strategy is A*, both in terms of number of optimal solutions found and running times. This behaviour depends on the criterion used for the selection of the label $L_i$ to be extracted: the value of the path from node $i$ to the target $t$ is an estimation (i.e. a lower bound) of the final cost that can be obtained starting from the path associated with $L_i$.

**Table 3** Comparison of extraction policies on fully random graphs

| Problem | *p* | DR | | FIFO | | LIFO | | SLF | | A* | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Avg. time | O | Avg. time | O | Avg. time | O | Avg. time | O | Avg. time | O |
| R1 | 0.15 | 300.23 | 1 | 1.70 | 2 | 3.68 | 2 | 0.86 | 2 | 0.30 | 2 |
| R1 | 0.20 | 301.37 | 1 | 1.04 | 2 | 10.99 | 2 | 1.97 | 2 | 0.32 | 2 |
| R2 | 0.15 | 300.75 | 1 | 300.22 | 1 | 305.07 | 1 | 300.45 | 1 | 300.20 | 1 |
| R2 | 0.20 | 9.20 | 2 | 1.75 | 2 | 5.05 | 2 | 2.77 | 2 | 0.36 | 2 |
| R3 | 0.15 | 307.33 | 1 | 5.20 | 2 | 11.10 | 2 | 2.85 | 2 | 0.81 | 2 |
| R3 | 0.20 | 600.00 | 0 | 302.06 | 1 | 2.92 | 2 | 14.08 | 2 | 0.53 | 2 |
| R4 | 0.15 | 1.49 | 2 | 0.58 | 2 | 6.41 | 2 | 2.04 | 2 | 0.34 | 2 |
| R4 | 0.20 | 301.41 | 1 | 3.28 | 2 | 17.60 | 2 | 6.64 | 2 | 0.36 | 2 |
| R5 | 0.15 | 600.00 | 0 | 4.20 | 2 | 16.43 | 2 | 302.31 | 1 | 1.10 | 2 |
| R5 | 0.20 | 600.00 | 0 | 302.01 | 1 | 25.24 | 2 | 5.61 | 2 | 0.71 | 2 |
| R6 | 0.15 | 4.36 | 2 | 3.17 | 2 | 304.80 | 1 | 9.49 | 2 | 0.51 | 2 |
| R6 | 0.20 | 301.12 | 1 | 302.83 | 1 | 326.87 | 1 | 303.83 | 1 | 0.51 | 2 |
| R7 | 0.15 | 301.49 | 1 | 300.79 | 1 | 306.41 | 1 | 1.23 | 2 | 0.50 | 2 |
| R7 | 0.20 | 6.57 | 2 | 300.32 | 1 | 12.95 | 2 | 4.69 | 2 | 0.52 | 2 |
| R8 | 0.15 | 13.84 | 2 | 2.24 | 2 | 17.78 | 2 | 3.56 | 2 | 0.59 | 2 |
| R8 | 0.20 | 301.24 | 1 | 11.94 | 2 | 14.14 | 2 | 7.79 | 2 | 0.93 | 2 |
| R9 | 0.15 | 303.51 | 1 | 300.75 | 1 | 32.45 | 2 | 301.13 | 1 | 0.82 | 2 |
| R9 | 0.20 | 600.00 | 0 | 301.07 | 1 | 331.79 | 1 | 302.31 | 1 | 0.83 | 2 |
| Average | | 286.33 | | 135.84 | | 97.32 | | 87.42 | | 17.24 | |
| Sum | | | 19 | | 28 | | 31 | | 31 | | 35 |

**Table 4** Comparison of extraction policies on grid graphs

| Problem | $p$ | DR | | FIFO | | LIFO | | SLF | | A* | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Avg. time | O | Avg. time | O | Avg. time | O | Avg. time | O | Avg. time | O |
| G1 | 0.15 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 0.20 | 2 |
| G1 | 0.20 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 0.19 | 2 |
| G2 | 0.15 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 12.33 | 2 |
| G2 | 0.20 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 0.10 | 2 |
| G3 | 0.15 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 0.70 | 2 |
| G3 | 0.20 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 3.59 | 2 |
| G4 | 0.15 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 3.33 | 2 |
| G4 | 0.20 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 1.62 | 2 |
| G5 | 0.15 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 600.00 | 0 |
| G5 | 0.20 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 9.38 | 2 |
| G6 | 0.15 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 120.21 | 2 |
| G6 | 0.20 | 600 | 0 | 600 | 0 | 600 | 0 | 600 | 0 | 101.83 | 2 |
| Average | | 600 | | 600 | | 600 | | 600 | | 71.12 | |
| Sum | | | 0 | | 0 | | 0 | | 0 | | 22 |

**Table 5** Comparison of branching strategy on fully random graphs

| Problem | $p$ | BF | | DF | |
|---|---|---|---|---|---|
| | | Avg.time | O + F | Avg. time | O + F |
| R1 | 0.15 | 0.19 | 2 + 0 | 1.17 | 2 + 0 |
| R1 | 0.20 | 9.52 | 2 + 0 | 40.55 | 2 + 0 |
| R2 | 0.15 | 300.39 | 1 + 0 | 302.18 | 1 + 0 |
| R2 | 0.20 | 300.13 | 1 + 1 | 300.74 | 1 + 1 |
| R3 | 0.15 | 600.00 | 0 + 2 | 600.01 | 0 + 2 |
| R3 | 0.20 | 314.10 | 1 + 1 | 378.88 | 1 + 1 |
| R4 | 0.15 | 0.30 | 2 + 0 | 1.06 | 2 + 0 |
| R4 | 0.20 | 0.28 | 2 + 0 | 1.09 | 2 + 0 |
| R5 | 0.15 | 600.00 | 0 + 2 | 600.00 | 0 + 2 |
| R5 | 0.20 | 600.00 | 0 + 2 | 600.00 | 0 + 2 |
| R6 | 0.15 | 0.41 | 2 + 0 | 1.18 | 2 + 0 |
| R6 | 0.20 | 0.41 | 2 + 0 | 2.07 | 2 + 0 |
| R7 | 0.15 | 300.49 | 1 + 1 | 300.61 | 1 + 1 |
| R7 | 0.20 | 394.25 | 1 + 1 | 420.85 | 1 + 1 |
| R8 | 0.15 | 300.27 | 1 + 1 | 300.32 | 1 + 1 |
| R8 | 0.20 | 600.00 | 0 + 2 | 600.00 | 0 + 2 |
| R9 | 0.15 | 309.33 | 1 + 1 | 311.26 | 1 + 1 |
| R9 | 0.20 | 313.93 | 1 + 1 | 316.92 | 1 + 1 |
| Average | | 274.67 | | 282.16 | |
| Sum | | | 20 + 15 | | 20 + 15 |

As a consequence, the convergence to a feasible—indeed optimal—solution is more rapid with respect to the other strategies and a significant number of opened labels can be pruned when the optimum is found (see Remark 1).

### 4.3 Analysis of the branching strategy

We compared a best-first (BF) and a depth-first (DF) strategy of exploration of the branching tree. On the one hand, BF extracts the branching node that presents a relaxed solution with the highest cost. On the other hand, DF extracts a node from the deepest level. The results are collected in Tables 5 and 6 and report

- the average time for the resolution of instances of the reference type;
- the number O of optimal solutions found within the time limit;
- the number F of feasible (non optimal) solutions found within the time limit.

Though none of the two strategies outperforms the other one, we observe that BF presents slightly lower computational times and is able to find an extra optimal solution (compared to DF) for the grid graphs.

**Table 6** Comparison of branching strategy on fully grid graphs

| Problem | $p$ | BF | | DF | |
|---|---|---|---|---|---|
| | | Avg. time | O + F | Avg. time | O + F |
| G1 | 0.15 | 600.00 | 0 + 2 | 600.00 | 0 + 2 |
| G1 | 0.20 | 300.00 | 1 + 1 | 300.01 | 1 + 1 |
| G2 | 0.15 | 600.00 | 0 + 1 | 600.82 | 0 + 1 |
| G2 | 0.20 | 362.58 | 1 + 0 | 600.02 | 0 + 2 |
| G3 | 0.15 | 462.39 | 1 + 1 | 493.50 | 1 + 1 |
| G3 | 0.20 | 600.00 | 0 + 2 | 600.00 | 0 + 2 |
| G4 | 0.15 | 116.79 | 2 + 0 | 589.58 | 1 + 1 |
| G4 | 0.20 | 600.00 | 0 + 2 | 379.91 | 1 + 1 |
| G5 | 0.15 | 600.00 | 0 + 2 | 600.28 | 0 + 2 |
| G5 | 0.20 | 300.21 | 1 + 1 | 300.77 | 1 + 1 |
| G6 | 0.15 | 300.49 | 1 + 1 | 301.51 | 1 + 1 |
| G6 | 0.20 | 300.48 | 1 + 1 | 302.12 | 1 + 1 |
| Average | | 428.58 | | 472.38 | |
| Sum | | | 8 + 14 | | 7 + 16 |

### 4.4 Algorithms comparison

Finally, the last experimentation compares DP with A* extraction strategy, B&B with best-first branching strategy, and the model solved by CPLEX. The results are collected in Tables 7 and 8.

Analyzing the performances achieved by the algorithms on random instances (Table 7), it is possible to observe how the average computational times spent by DP are sensibly smaller, being at least an order of magnitude lower with respect to those achieved by B&B and CPLEX. Additionally, the number of optimal solutions encountered by means of DP ($^{176}/_{180} = 97.78\%$) almost doubles the number of optimal solutions of the second best algorithm, i.e., 93 optima found by B&B.

Instead, the solution of $k$-CSPP on grid graphs (Table 8) requires on average higher computational times and comparing the number of optimally solved instances with those reported in Table 7, it is possible to note a sensible decrease. This behavior, observed in all the solution techniques here considered, can be attributed to the specific topology characterizing grids. In fact, such graphs are much sparser than the random networks of R1-R9, and the search for a shortest path coupled with a color restriction represents a harder task. Nonetheless, both the number of optimal solutions and the average computational times exhibited by DP outperform its two competitors.

Additionally, given the Remark 1, both on random and grid graphs, DP either converges to the optimal solution in the time-limit or is not able to find a feasible solution. On the contrary, B&B is able to encounter feasible solutions in almost the totality of the tackled instances ($^{277}/_{300} = 92.33\%$).

It is properly noted how the average times alone are not sufficient to grasp the performances achieved by the algorithms. This is clear, for example, when considering the distribution of computational times spent by the algorithms on G1 instances with

**Table 7** Computational results on fully random graphs of the data-set $\mathcal{A}$

| Problem | $p$ | B&B BF | | CPLEX | | DP | |
|---|---|---|---|---|---|---|---|
| | | Avg. time | O + F | Avg. time | O + F | Avg. time | O + F |
| R1 | 0.15 | 301.44 | 5 + 2 | 92.43 | 10 + 0 | 1.56 | 10 + 0 |
| R1 | 0.20 | 301.98 | 5 + 4 | 89.12 | 9 + 0 | 2.35 | 9 + 0 |
| R2 | 0.15 | 182.08 | 7 + 2 | 158.54 | 9 + 0 | 60.24 | 9 + 0 |
| R2 | 0.20 | 182.09 | 7 + 2 | 163.05 | 9 + 0 | 60.24 | 9 + 0 |
| R3 | 0.15 | 304.27 | 5 + 5 | 521.92 | 2 + 0 | 0.47 | 10 + 0 |
| R3 | 0.20 | 304.21 | 5 + 5 | 513.47 | 2 + 0 | 0.49 | 10 + 0 |
| R4 | 0.15 | 300.18 | 5 + 3 | 123.57 | 10 + 0 | 0.65 | 10 + 0 |
| R4 | 0.20 | 258.91 | 6 + 1 | 218.52 | 8 + 0 | 3.95 | 9 + 0 |
| R5 | 0.15 | 360.18 | 4 + 6 | 600 | 0 + 0 | 0.47 | 10 + 0 |
| R5 | 0.20 | 360.18 | 4 + 6 | 600 | 0 + 0 | 0.50 | 10 + 0 |
| R6 | 0.15 | 120.36 | 8 + 2 | 600 | 0 + 0 | 0.49 | 10 + 0 |
| R6 | 0.20 | 120.36 | 8 + 2 | 600 | 0 + 0 | 0.50 | 10 + 0 |
| R7 | 0.15 | 379.21 | 4 + 5 | 600 | 0 + 0 | 0.43 | 10 + 0 |
| R7 | 0.20 | 379.44 | 4 + 5 | 512.84 | 2 + 0 | 0.43 | 10 + 0 |
| R8 | 0.15 | 420.31 | 3 + 5 | 600 | 0 + 0 | 0.63 | 10 + 0 |
| R8 | 0.20 | 420.26 | 3 + 5 | 600 | 0 + 0 | 0.64 | 10 + 0 |
| R9 | 0.15 | 305.46 | 5 + 5 | 600 | 0 + 0 | 0.71 | 10 + 0 |
| R9 | 0.20 | 304.75 | 5 + 5 | 600 | 0 + 0 | 0.72 | 10 + 0 |
| Average | | 294.76 | | 432.97 | | 7.53 | |
| Sum | | | 93 + 70 | | 61 + 0 | | 176 + 0 |

**Table 8** Computational results on grid graphs of the data-set $\mathcal{A}$

| Problem | $p$ | B&B BF | | CPLEX | | DP | |
|---|---|---|---|---|---|---|---|
| | | Avg. time | O + F | Avg. time | O + F | Avg. time | O + F |
| G1 | 0.15 | 422.02 | 3 + 6 | 41.05 | 10 + 0 | 0.92 | 10 + 0 |
| G1 | 0.20 | 420.95 | 3 + 6 | 42.16 | 10 + 0 | 60.81 | 9 + 0 |
| G2 | 0.15 | 483.70 | 2 + 6 | 200.71 | 10 + 0 | 3.81 | 10 + 0 |
| G2 | 0.20 | 498.76 | 2 + 6 | 225.43 | 9 + 0 | 12.69 | 10 + 0 |
| G3 | 0.15 | 575.96 | 1 + 9 | 596.76 | 1 + 7 | 130.83 | 8 + 0 |
| G3 | 0.20 | 544.83 | 1 + 9 | 587.55 | 2 + 7 | 130.01 | 8 + 0 |
| G4 | 0.15 | 434.49 | 4 + 6 | 602.28 | 0 + 10 | 64.99 | 9 + 0 |
| G4 | 0.20 | 421.90 | 4 + 6 | 602.36 | 0 + 10 | 62.00 | 9 + 0 |
| G5 | 0.15 | 540.04 | 1 + 9 | 607.63 | 0 + 9 | 367.50 | 4 + 0 |
| G5 | 0.20 | 486.18 | 2 + 8 | 607.07 | 0 + 10 | 362.37 | 4 + 0 |
| G6 | 0.15 | 540.11 | 1 + 9 | 600.00 | 0 + 0 | 232.51 | 7 + 0 |
| G6 | 0.20 | 540.10 | 1 + 9 | 600.00 | 0 + 0 | 275.19 | 6 + 0 |
| Average | | 492.42 | | 442.75 | | 141.97 | |
| Sum | | | 25 + 89 | | 42 + 53 | | 94 + 0 |

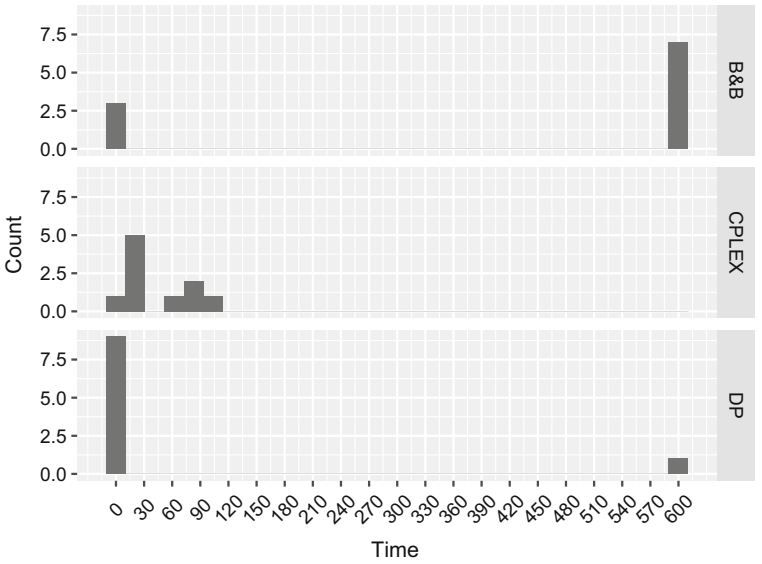


**Fig. 5** Distribution of computational times on G1 instances with $p = 0.20$

$p = 0.20$, reported in the bar-chart of Fig. 5. Among the three methods, it is evident how DP reaches the optimal solution in less than 1 s on 9 graphs, while on a single instance it is not able to encounter a feasible solution within the given time limit (600 s). The resulting average time of 60 s can not be clearly compared with the one achieved by CPLEX, equal to 42 s, resulting from generally higher times with smaller variance.

Once again, the study of the distribution of computational times on the whole data-set (Fig. 6) shows that DP is often extremely fast in the construction of an optimal solution, and in few cases struggles in the pursuit of a feasible solution. On the contrary, for B&B and CPLEX, the number of instances requiring the whole allotted time limit, or in general times greater than few seconds, is sensibly higher.

In order to study the performances while varying the number of colors, we executed the three algorithms on the instances of data-set $\mathcal{B}$, and the results are reported in Tables 9 and 10.

Table 11 reports a summarized comparison between the results on data-sets $\mathcal{A}$ and $\mathcal{B}$. The results highlight that DP efficiency is strongly affected by the number of colors. DP presents a ratio of computational time on $\mathcal{B}$ over computational time on $\mathcal{A}$ equal to 0.07 and 0.11 for random and grid graphs, respectively. This behavior was expected, since the lower the number of colors the lower the number of feasible non-dominated labels (see Remark 2).

On the contrary, it is not possible to identify a clear trend in the performances of B&B and CPLEX at varying the number of colors. On the one hand, on grid graphs, the computational times tend to decrease when lowering the number of colors. On the contrary, the trend appears to be inverse on fully random graphs.

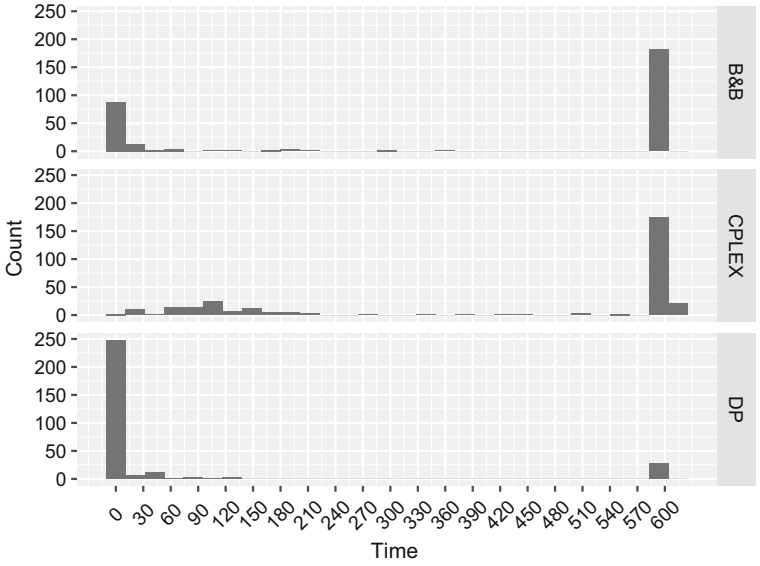


**Fig. 6** Distribution of computational times on the whole data-set

**Table 9** Computational results on random graphs of the data-set $\mathcal{B}$

| Problem | $p$ | B&B BF | | CPLEX | | DP | |
|---|---|---|---|---|---|---|---|
| | | Avg. time | O + F | Avg. time | O + F | Avg. time | O + F |
| R1 | 0.01 | 361.88 | 4 + 4 | 151.50 | 10 + 0 | 0.28 | 10 + 0 |
| R1 | 0.02 | 361.89 | 4 + 4 | 98.44 | 10 + 0 | 0.29 | 10 + 0 |
| R2 | 0.01 | 313.24 | 5 + 5 | 200.60 | 10 + 0 | 0.29 | 10 + 0 |
| R2 | 0.02 | 312.99 | 5 + 5 | 142.71 | 10 + 0 | 0.29 | 10 + 0 |
| R3 | 0.01 | 370.50 | 4 + 4 | 600.00 | 0 + 0 | 0.91 | 10 + 0 |
| R3 | 0.02 | 370.58 | 4 + 4 | 560.25 | 1 + 0 | 1.01 | 10 + 0 |
| R4 | 0.01 | 260.80 | 6 + 2 | 393.65 | 5 + 0 | 0.32 | 10 + 0 |
| R4 | 0.02 | 199.88 | 7 + 1 | 363.72 | 5 + 0 | 0.31 | 10 + 0 |
| R5 | 0.01 | 360.19 | 4 + 6 | 600.00 | 0 + 0 | 0.40 | 10 + 0 |
| R5 | 0.02 | 360.19 | 4 + 6 | 600.00 | 0 + 0 | 0.40 | 10 + 0 |
| R6 | 0.01 | 360.21 | 4 + 5 | 600.00 | 0 + 0 | 0.80 | 10 + 0 |
| R6 | 0.02 | 360.21 | 4 + 5 | 600.00 | 0 + 0 | 0.84 | 10 + 0 |
| R7 | 0.01 | 378.94 | 4 + 5 | 429.33 | 5 + 0 | 0.45 | 10 + 0 |
| R7 | 0.02 | 378.89 | 4 + 5 | 391.06 | 5 + 0 | 0.47 | 10 + 0 |
| R8 | 0.01 | 420.26 | 3 + 6 | 600.00 | 0 + 0 | 1.01 | 10 + 0 |
| R8 | 0.02 | 420.25 | 3 + 6 | 600.00 | 0 + 0 | 1.16 | 10 + 0 |
| R9 | 0.01 | 446.22 | 3 + 5 | 600.00 | 0 + 0 | 0.81 | 10 + 0 |
| R9 | 0.02 | 446.08 | 3 + 5 | 600.00 | 0 + 0 | 0.82 | 10 + 0 |
| Average | | 360.18 | | 451.74 | | 0.60 | |
| Sum | | | 75 + 83 | | 61 + 0 | | 180 + 0 |

**Table 10** Computational results on grid graphs of the data-set $\mathcal{B}$

| Problem | $p$ | B&B BF | | CPLEX | | DP | |
|---|---|---|---|---|---|---|---|
| | | Avg. time | O + F | Avg. time | O + F | Avg. time | O + F |
| G1 | 0.01 | 228.44 | 7 + 3 | 25.36 | 10 + 0 | 0.03 | 10 + 0 |
| G1 | 0.02 | 365.88 | 4 + 6 | 34.44 | 10 + 0 | 0.07 | 10 + 0 |
| G2 | 0.02 | 360.74 | 4 + 5 | 110.25 | 10 + 0 | 0.11 | 10 + 0 |
| G2 | 0.01 | 296.94 | 6 + 4 | 122.44 | 10 + 0 | 0.09 | 10 + 0 |
| G3 | 0.01 | 445.39 | 3 + 7 | 600.62 | 0 + 0 | 0.92 | 10 + 0 |
| G3 | 0.02 | 426.04 | 3 + 7 | 559.21 | 4 + 5 | 0.79 | 10 + 0 |
| G4 | 0.01 | 414.45 | 4 + 6 | 601.25 | 0 + 0 | 1.01 | 10 + 0 |
| G4 | 0.02 | 432.01 | 3 + 7 | 601.22 | 0 + 10 | 2.41 | 10 + 0 |
| G5 | 0.01 | 369.38 | 4 + 6 | 600.00 | 0 + 0 | 50.67 | 10 + 0 |
| G5 | 0.02 | 371.41 | 4 + 6 | 600.00 | 0 + 0 | 18.37 | 10 + 0 |
| G6 | 0.01 | 425.70 | 3 + 7 | 600.00 | 0 + 0 | 46.33 | 10 + 0 |
| G6 | 0.02 | 481.60 | 2 + 8 | 600.00 | 0 + 0 | 69.49 | 10 + 0 |
| Average | | 384.83 | | 421.23 | | 15.86 | |
| Sum | | | 47 + 72 | | 44 + 15 | | 120 + 0 |

**Table 11** Comparison for data-sets $\mathcal{A}$ and $\mathcal{B}$

| data-set | B&B BF | | | CPLEX | | DP | |
|---|---|---|---|---|---|---|---|
| | Topology | Avg. time | O + F | Avg. time | O + F | Avg. time | O + F |
| $\mathcal{A}$ | Random | 294.76 | 93 + 70 | 432.97 | 61 + 0 | 7.53 | 176+ 0 |
| $\mathcal{A}$ | Grid | 492.42 | 25 + 89 | 442.75 | 42 + 53 | 141.97 | 94 + 0 |
| $\mathcal{B}$ | Random | 360.18 | 75 + 83 | 451.74 | 61 + 0 | 0.60 | 180 + 0 |
| $\mathcal{B}$ | Grid | 384.83 | 47 + 72 | 421.23 | 44 + 15 | 15.86 | 120 + 0 |

## 5 Conclusions and future works

The problem addressed in this paper is collocated in the frame of Constrained Shortest Path Problems (CSPP). Specifically, we investigated a recent variant of CSPP, named $k$-Colored Shortest Path ($k$-CSPP) and proposed in [7].

The solution framework we devised consists of a dynamic programming (`DP`) algorithm based on a path-labeling approach and an $A^*$-like exploration strategy. With the aim of validating this approach, we compared the performances of `DP` with those achieved by two alternative methods: the direct solution of the mathematical model obtained with CPLEX solver, and the Branch and Bound technique described in [7]. These three techniques have been tested on two data-sets that comprises two different graph topologies: random and grid graphs. The experimental results show that `DP` outperforms its two competitors both in terms of computational times and number of optimal solutions found within the given time limit.

Due to the computational intractability of the problem, future research streams will be mainly focused on the design of efficient heuristic approaches with the aim to solve large size instances in short computational time. Moreover, in order to properly model the complexity of real-world networks, different failure probabilities $p_c$, for each color $c \in C(E)$, could be considered; then the resulting bi-objective problem could be solved through a sim-heuristic approach [8,10].

**Funding** Open access funding provided by Università della Calabria within the CRUI-CAREAgreement.



## References

1. Bertsekas, D.P.: A simple and fast label correcting algorithm for shortest paths. Networks **23**(8), 703–709 (1993). https://doi.org/10.1002/net.3230230808
2. Broersma, H., Li, X., Woeginger, G.J., Zhang, S.: Paths and cycles in colored graphs. Australas. J. Comb. **31**, 299–312 (2005)
3. Carrabs, F., Cerulli, R., Felici, G., Singh, G.: Exact approaches for the orderly colored longest path problem: performance comparison. Comput. Oper. Res. **101**, 275–284 (2019)
4. Di Puglia Pugliese, L., Ferone, D., Festa, P., Guerriero, F.: Shortest path tour with time windows. Eur. J. Oper. Res. **282**(1), 334–344 (2020). https://doi.org/10.1016/j.ejor.2019.08.052
5. Di Puglia Pugliese, L., Guerriero, F.: A survey of resource constrained shortest path problems: exact solution approaches. Networks **62**(3), 183–200 (2013)
6. Ferone, D., Festa, P., Fugaro, S., Pastore, T.: Instances for the k-color shortest path problem (2020). https://doi.org/10.6084/m9.figshare.11762163.v1
7. Ferone, D., Festa, P., Pastore, T.: The k-color shortest path problem. In: Paolucci, M., Sciomachen, A., Uberti, P. (eds.) Advances in Optimization and Decision Science for Society, Services and Enterprises: ODS, Genoa, Italy, September 4–7, 2019, pp. 367–376. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-34960-8_32
8. Ferone, D., Gruler, A., Festa, P., Juan, A.A.: Enhancing and extending the classical GRASP framework with biased randomisation and simulation. J. Oper. Res. Soc. **70**(8), 1362–1375 (2019). https://doi.org/10.1080/01605682.2018.1494527
9. Festa, P., Pallottino, S.: A pseudo-random networks generator. Technical report, Department of Mathematics and Applications "R. Caccioppoli", University of Naples FEDERICO II, Italy (2003)
10. Festa, P., Pastore, T., Ferone, D., Juan, A.A., Bayliss, C.: Integrating biased-randomized GRASP with Monte Carlo simulation for solving the vehicle routing problem with stochastic demands. In: 2018 Winter Simulation Conference (WSC), pp. 2989–3000 (2019). https://doi.org/10.1109/WSC.2018.8632348
11. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybernet. **4**(2), 100–107 (1968)
12. Powell, W.B., Chen, Z.: A generalized threshold algorithm for the shortest path problem with time windows. DIMACS Series Discrete Math. Theor. Comput. Sci. **40**, 303–318 (1998)
13. Righini, G., Salani, M.: Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optim. **3**(3), 255–273 (2006). https://doi.org/10.1016/J.DISOPT.2006.05.007
14. Yuan, S., Jue, J.P., et al.: Shared protection routing algorithm for optical network. Opt. Netw. Mag. **3**(3), 32–39 (2002)

15. Yuan, S., Varma, S., Jue, J.P.: Minimum-color path problems for reliability in mesh networks. In: IEEE INFOCOM, vol. 4, p. 2658 (2005)