Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-10 (2022) 572–577

A Problem-Specific Branch-and-Bound Algorithm for the Protected Shortest Simple Path Problem with Must-Pass Nodes

Yuri Ogorodnikov *,** Roman Rudakov *,** Daniil Khachai *** Michael Khachay *

* *Krasovsky Institute of Mathematics and Mechanics, Ekaterinburg, Russia (e-mail: {khachay,yogorodnikov}@imm.uran.ru)*
** *Ural State Federal University, Ekaterinburg, Russia, (e-mail: r.a.rudakov@gmail.com)*
*** *Kedge Business School, Bordeaux, France (e-mail: daniil.khachai@kedgebs.com)*

Abstract: An instance of the Protected Shortest Simple Path Problem with Must-Pass Nodes (PSSPP-MPN) is specified by an edge-weighted directed graph with dedicated source, destination, and additional must-pass nodes. The goal is to find two vertex-disjoint paths, such that the former one is simple, visits all the must-pass nodes, and has the minimum transportation cost. In this paper, we show that the PSSPP-MPN is strongly NP-hard even for subsets of must-pass nodes of arbitrary fixed size and propose a novel problem-specific branch-and-bound algorithm for this problem. Results of competitive numerical evaluation against the public dataset 'Rome99' from the 9th DIMACS Implementation Challenge show that the proposed algorithm notably outperforms the state-of-the-art MIP-optimizer Gurobi both by accuracy and execution time.

Keywords: Protected Shortest Simple Path Problem with Must-Pass Nodes, Branch-and-Bound algorithm, MILP models

1. INTRODUCTION

The Protected Shortest Simple Path Problem with Must-Pass Nodes (PSSPP-MPN) is a combinatorial optimization problem that has a great influence on communication network construction (Rak (2015); Su et al. (2019)).

Generally speaking, the PSSPP-MPN can be formulated as follows.

We are given by a transportation network represented in terms of an edge-weighted digraph with selected *source* node s, *destination* node t and an additional subset of *must-pass* nodes F, respectively. It is required to find a shortest simple s-t-path visiting all the nodes from F (also called *main path*) protected by an additional *backup* s-t-path that has no common nodes with the former one, except its end-points. In communication networks, such a path plays a role of a backup route in case of any accidence on the main one (Cholda et al. (2008, 2009)).

Unlike the close Shortest Path Problem (SPP) that can be solved to optimality in polynomial time by the well-known Dijkstra's algorithm (Dijkstra (1959)), the PSSPP-MPN appears to be strongly NP-hard enclosing the classic Traveling Salesman Problem (TSP).

Related work. The PSSPP-MPN belongs to a wide family of the optimal routing combinatorial optimization problems, including the classic TSP and its generalizations (see, e.g. Gutin and Punnen (2007); Khachay and Neznakhina (2018, 2020)) and Vehicle Routing Problem (Pessoa et al. (2020); Khachai and Dubinin (2017); Khachay et al. (2021)). Among them, the most close is the Shortest Simple Path Problem with Must-Pass Nodes (SSPP-MPN), whose goal is the same as for the considered problem excluding a backup path construction. To the best of our knowledge, the SSPP-MPN was introduced in (Saksena and Kumar (1966)). Also, in the same paper there was proposed the first simple algorithm for SSPP-MPN but, as shown in (Dreyfus (1969)), it turns to be erroneous. Later, there were developed the first dynamic programming scheme and branch-and-bound algorithm (Ibaraki (1973)) relied on the flow MILP-model for the classic Shortest Path Problem. Unfortunately, these methods have a huge running times and cannot be applied to practical instances.

First compact MILP-models, which can be tackled by MIP-solvers, were proposed in (Andrade (2016)). Several efficient meta-heuristics are known for SSPP-MPN (Su et al. (2019); Gomes et al. (2017); Martins et al. (2017)). Finally, in (Kudriavtsev et al. (2021)), a first branch-and-bound algorithm was proposed, which is proven to be efficient even on large networks (Kudriavtsev et al. (2021)).

10.1016/j.ifacol.2022.09.455

The well-known shortest k-Vertex-Disjoint Paths Problem (k-DPP) turns to be another problem close to the PSSPP-MPN. An instance of the k-DPP is given by an edge-weighted graph $G = (V, E, c)$, where $c\colon E \to \mathbb{R}_+$ specifies the transportation costs, and a collection of ordered pairs $C = \{(s_i, t_i) \in V^2\}$, $i = \overline{1,k}$, where s_i and t_i are source and destination nodes respectively. The goal is to find k simple node-disjoint paths connecting s_i and t_i such that its total cost is minimal.

If k belongs to the instance, the DPP is NP-hard (Karp (1972)). Moreover, the k-DPP is NP-hard for any fixed $k \geq 2$ if the graph G is oriented (Fortune et al. (1980)). Nevertheless, there are known several efficient algorithms proposed for the case of planar graphs (Datta et al. (2018)) and for the class of undirected graphs when $k = 2$ (Björklund and Husfeldt (2014)).

Unlike the aforementioned problems, the PSSPP-MPN was not intensively researched in terms of algorithmic design. This problem was firstly introduced in paper (Gomes et al. (2015)). Later, in papers (Gomes et al. (2017); Martins et al. (2017)), several heuristics for the PSSPP-MPN were proposed and evaluated on the data from SNDLib library (SNDLib (2022)). Although, the mentioned heuristics perform quite well on this dataset, sizes of the testing instances (at most 300 nodes in a graph) are still far from the practice. In this paper, in attempt to bridge this gap, we propose two more efficient branch-and-bound algorithms.

Our contribution is two-fold. First, we prove that PSSPP-MPN is NP-hard even for any fixed positive number of must-pass nodes. Second, we describe a novel problem-specific branch-and-bound algorithm that can be applied to sufficiently large instances of the PSSPP-MPN, and prove its high performance on a real-life dataset taken from the 9th DIMACS Implementation Challenge - Shortest Paths (DIMACS (2006)) in comparison with the state-of-the-art MIP-optimizer Gurobi.

The rest of our paper is structured as follows. In Section 2, we recall mathematical formulation of the PSSPP-MPN and the corresponding MILP-model, which is used in the subsequent numerical experiments as a baseline. Further, in section 3, we establish complexity status of the PSSPP-MPN. In Section 4, we describe two versions of the proposed branch-and-bound algorithm.

Section 5 is dedicated for the numerical evaluation carried out on the dataset 'Rome99' DIMACS (2006). Finally, in Section 6, we summarize our results and discuss open questions.

2. PROBLEM STATEMENT

An arbitrary instance of the Protected Shortest Simple Path Problem with Must Pass Nodes (PSSPP-MPN) can be given by an edge-weighted directed graph $G = (V, A, c)$, where $c : A \to \mathbb{R}_+$ specifies transportation costs, an ordered pair (s, t) called *source* and *destination*, and a subset $F \subset V$ of must pass nodes that can be visited in an arbitrary order. The goal is to construct a pair of s-t-paths (p_1^*, p_2^*), such that

- p_1^* is simple and visits all the nodes from F;
- p_1^* and p_2^* are node-disjoint;
- p_1^* has mininum cost.

In the sequel, we refer to p_1^* and p_2^* as *main* and *backup* paths, respectively. Furthermore, without loss of generality, we assume that the node s has no incoming and the node t has no outgoing arcs.

In (Gomes et al. (2017)), a compact MILP-model for the problem in question, based on dual variables and *Big-M* penalty parameters was proposed. Unfortunately, this model appears to be insufficiently robust to small distortions in the input data. Therefore, for the subsequent numerical evaluations, we employ another model, which is our adaptation of the model Q_4 from (Andrade, 2016).

$$(M):\ \min \sum_{(i,j)\in A} c_{ij} x_{ij,1} \quad \text{s.t.} \tag{1}$$

$$\sum_{(i,j)\in A} x_{ij,p} - \sum_{(j,i)\in A} x_{ji,p} = \begin{cases} -1, & i = t, \\ 1, & i = s, \\ 0, & \text{otherwise} \end{cases} \quad \begin{matrix}(i \in V),\\ (p \in \{1,2\})\end{matrix} \tag{2}$$

$$\sum_{(i,j)\in A} x_{ij,1} = \sum_{(j,i)\in A} x_{ji,1} = 1 \quad (i \in F) \tag{3}$$

$$\sum_{(i,j)\in A} f_{ij,1} - \sum_{(j,i)\in A} f_{ji,1} = \begin{cases} |F|+1, & i = s \\ -1, & i \in F \cup \{t\} \\ 0, & \text{otherwise} \end{cases} \quad (i \in V) \tag{4}$$

$$x_{ij,1} \leq f_{ij,1} \leq (|F|+1) \cdot x_{ij,1} \quad ((i,j) \in A) \tag{5}$$

$$\sum_{(i,j)\in A} (x_{ij,1} + x_{ij,2}) \leq 1, \quad (i \in V \setminus \{s,t\}) \tag{6}$$

$$x_{ij,p} \in \{0,1\}, \quad (i,j \in V) \quad (p \in \{1,2\}). \tag{7}$$

Here, the variable $x_{ij,k}$ indicates whether the arc $(i,j) \in A$ belongs or not to the path p_k. We introduce flow variables $f_{ij,1}$ to ensure connectivity of the subgraph induced by the arcs (i,j), for which $x_{ij,1} > 0$. Meantime, we skip the variables $f_{ij,2}$, since the similar subgraph defined by the values $x_{ij,2}$, besides a backup path, may contain some number of circulations.

As it follows from the problem statement, the objective function specified in (1) represents the cost of a main path. Then, equation (2) specifies the degree constraints for the both paths, while (3) forces the main path to to visit each the node from F (at once). Equations (4) and (5) allows us to avoid subtours in the constructed solution and to link flow $f_{ij,1}$ and arc $x_{ij,1}$ indicator variables. Finally, equation (6) guarantees that each node can be visited at most one time.

It is easy to see that the size of the proposed model depends polynomially on the size of the input graph G. More exactly, the number of variables and constraints is $O(|V| + |A|)$.

In the sequel, we will consider a special case of the PSSPP-MPN, where $|F| = k$, for some fixed number k. To the sake of clarity, we call this problem PSSPP-k-MPN.

3. COMPUTATIONAL COMPLEXITY

The problem PSSPP-k-MPN seems to be extremely close to the classic polynomially solvable Shortest Path Problem. Nevertheless, we show that PSSPP-k-MPN remains NP-hard for any natural k.

Theorem 1. For any fixed $k \geq 1$, the PSSPP-k-MPN on digraphs is strongly NP-hard.

Theorem 1 follows from the fact that the Simple Shortest Path Problem with k Must-Pass Nodes (SSPP-k-MPN) is polynomially reducible to the problem in question. Indeed, suppose we are given by an instance of the SSPP-k-MPN specified by an edge-weighted digraph G, source and destination nodes s and t, and a subset of must-pass nodes F. In the case, when s and t are adjacent, any simple s-t-path visiting all the nodes from F can evidently be augmented with the backup path that consists of the single arc (s, t).

Otherwise, we can assign to the initial instance of the SSPP-k-MPN an auxiliary instance of the PSSPP-k-MPN as follows. Consider an extension of the graph G obtained by inclusion an additional node w and two arcs (s, w) and (w, t). Since both instances have the same family of simple s-t-paths visiting all the must-pass nodes from the subset F ans each such a path can be augmented with the backup path s-w-t, the desired reduction follows.

To be fair, we should notice that the PSSPP-k-MPN become polynomially solvable any time, when transportation costs fulfills the triangle inequality.

4. PROBLEM SPECIFIC BRANCH-AND-BOUND ALGORITHM

In this section we describe the proposed algorithm. In fact, we propose two versions of this algorithm. We call the former one *basic*, since its structure seems to be a little bit simpler. On the other hand, the more advanced version performed slightly better in numerical tests (see results in Section 5).

4.1 Basic version

Our proposed algorithm is essentially based on the Branch-and-Bound method proposed recently in paper (Kudriavtsev et al. (2021)). Its main components are preprocessing, greedy start heuristic, branching, and asynchronous local search. Consider each of them in detail.

Preprocessing. For a given input graph we iteratively cut off all its leaves. Any time, when an arbitrary must-pass node becomes a leaf, we stop our algorithm and conclude that the given PSSPP-k-MPN instance is infeasible.

After the ending of this leaves-elimination procedure, we added to the obtained graph an extra node w, whose incoming and outgoing arcs are the same as for the s and t, respectively (see Fig. 1). Denote the obtained auxiliary graph as $\bar{G}$. Notice, that an arbitrary s-w-path in $\bar{G}$ corresponds to some feasible main path p_1^* in the initial graph G, while a w-t-path corresponds to some backup one.

Greedy start heuristic. For a given value of parameter σ, consider exactly σ random permutations of the set $F = \{m_1, \ldots, m_k\}$. For a permutation π, consider the ordered sequence $s, m_{\pi(1)}, \ldots, m_{\pi(k)}, w, t$, extend F as follows: $m_{\pi(0)} = s$, $m_{\pi(k+1)} = w$, and $m_{\pi(k+2)} = t$, and set $H = \bar{G}$. Then, for each $i \in \{0, .., k+1\}$, we try to connect $m_{\pi(i)}$ with $m_{\pi(i+1)}$ by a simple path segment in the graph H.

If such a segment does not exist, we break the loop and proceed with another permutation. Otherwise, we exclude

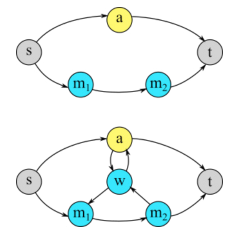

Fig. 1. An additional vertex w has incoming arcs the same as for s and outgoing arcs as for t. In this example, m_1 and m_2 are must-pass nodes

all the nodes of this segment (except $m_{\pi(i+1)}$) from the graph H and pass to the next iteration.

If we are managed to construct all the $m_{\pi(i)}$-$m_{\pi(i+1)}$-path segments, we obtain a feasible solution of our problem, where the main path is induced by a concatenation of the first $k+1$ of them and the last segment produces a backup path.

Branching. At any node (including root spawned by the initial graph G) we construct an auxiliary weighted digraph $G' = (Z, A')$ of shortest paths, such that

- $Z = F \cup \{s, w, t\}$, where w is the aforementioned extra node
- for any nodes z' and z'' in G' there is the arc (z', z'') if and only if in the digraph $\bar{G}$, for some r, there exists a path $P(r) = z', v_{i_1}, \ldots, v_{i_r}, z''$, s.t. $\{v_{i_1}, \ldots, v_{i_r}\} \cap Z = \varnothing$
- if $z' = w$ and $z'' = t$, we weight the arc $(z', z'') \in A'$ by the minimum r in such a path $P(r)$, otherwise the weight of (z', z'') is defined by the minimum cost of such paths $P(r)$.

Next, we find the shortest Hamiltonian s-t-path P' in G'. It can be done in a constant time, since $|F|$ is fixed. Then, we transform P' to the corresponding path P in the graph $\bar{G}$ and check whether P is simple or not. In the first case, P is a feasible solution, we update the UB record and cut off the current branch.

Otherwise, we perform the following actions:

- update the lower bound in the current node (also called *local lower bound (LB)*) with the cost of the found non-simple path P;
- recursively check the parent node and update its own local LB choosing the minimum among the local LBs of its child nodes;
- proceed with the branching by removing from the graph $\bar{G}$ each arc incident to a node of P visited more than once.

A branch is pruned any time, when one of the following criteria is met:

- feasible solution found
- the auxiliary graph G' has no Hamiltonian paths
- the weight of the obtained non-simple path P is greater than the current UB.

Local search. To improve the quality of the main branching procedure, we use the following asynchronous local search primal heuristic. Taking as input a non-simple path P and a node $\hat{v}$ visited by P more than once, our heuristic try to replace some segment of P containing $\hat{v}$ twice with another the shortest feasible segment containing this node with exactly ones. Notice, that if the initial segment visits some must-pass nodes, its replacement should visit these nodes as well. Any time, when the heuristic are managed to obtain a feasible solution, the UB record is also updated.

4.2 Improved version

In this section we describe a slightly more advanced version of the proposed Branch-and-Bound algorithm.

Instead of introducing an extra node w, in this version we find a desired pair of paths (p_1^*, p_2^*) directly. In addition, we introduce some supplementary heuristics, which lead to increasing of the overall performance. In the sequel, we restrict ourselves only on the updated features of the algorithm.

Greedy start heuristic. For any permutation π and a sequence $s = m_{\pi(0)}, m_{\pi(1)}, m_{\pi(k)}, t = m_{\pi(k+1)}$, any time, when we are managed to find a simple $m_{\pi(i)}$-$m_{\pi(i+1)}$-path segment, we verify whether there exists a s-t-path avoiding all the nodes form the constructed partial path connecting s and $m_{\pi(i+1)}$. In the case, where such a path is absent, we break the loop and proceed with another permutation.

If this greedy heuristic completes successfully, it provides both main and backup paths, i.e. a feasible solution of the given instance.

Branching. As the basic version, the branching is based on arc exclusion for any vertex of the graph G visited by the appropriate relaxed solution more than once. But, unlike the basic version of BnB Algorithm considered at the previous subsection, we start the processing of an arbitrary node of the search tree (including the root) with the following preliminary steps.

(i) we iteratively cut all leaves in the graph $\hat{G}$ associated with the current node (for the root, $\hat{G} = G$);
(ii) for the obtained graph, we verify the existence of a backup path from s to t that avoids all the must-pass nodes.

If we observe that some of must-pass nodes become a leaf at step (i) or there is no backup path at step (ii), we prune the considered branch.

On the other hand, if the steps (i) and (ii) are successfully passed, we construct an auxiliary weighted digraph $G' = (Z, A')$ by the similar way with the only difference: in this case, we need not to include an extra node w and set $Z = F \cup \{s, t\}$.

As for the basic version, we proceed with finding a shortest Hamiltonian path. After that, we transform it to the main path p_1 in the initial graph $\hat{G}$. By construction, there exist another s-t-path p_2 that avoids all the nodes from F. If the paths p_1 and p_2 are node-disjoint (except s and t), and p_1 is simple, we update record UB and cut off the current branch. Otherwise, we proceeding with branching by removing edges incident to nodes $(p_1 \cap p_2) \setminus \{s, t\}$.

Local search. At this stage, we added to the local graph copy $\hat{G}$ an additional node w, whose incoming arcs are the same as for t and outgoing are as for the node s. After that, we concatenate our paths $p_1 = s, v_{i_1}, \ldots, v_{i_r}, t$ and $p_2 = s, v_{j_1}, \ldots, v_{j_q}, t$, to produce the auxiliary path $\tilde{p}$ as follows:

$$\tilde{p} = s, v_{i_1}, \ldots, v_{i_r}, w, v_{j_1}, \ldots, v_{j_q}, t$$

and feed the obtained not necessarily simple path as an input to our local search heuristic. If local search are managed to transform this path to a simple one, we obtain once more feasible solution and update the UB record.

We implemented both algorithms on *Python 3.8* using *NetworkX* and *multiprocessing* packages with no other non-standard dependencies.

5. NUMERICAL EVALUATION

In this section, we describe how we carried out the numerical experiment and evaluated the performance of the proposed algorithms.

Table 1. Evaluation summary: best values are highlighted.

	feas. sol. (%)	**avg. gap (%)**	**avg. time (sec)**
k		2	
A_1	99.8	3.4	**421.7**
A_2	99.8	**2.2**	585.5
M	100.0	3.9	1526.9
M_{MIPS}	100.0	2.8	1228.7
k		4	
A_1	100.0	4.9	**739.2**
A_2	100.0	**3.2**	916.8
M	100.0	12.8	3018.0
M_{MIPS}	100.0	7.4	2704.5
k		6	
A_1	99.4	5.6	**924.2**
A_2	99.8	**5.3**	1459.1
M	99.6	15.2	3470.0
M_{MIPS}	100.0	8.2	3191.4
k		8	
A_1	99.6	6.4	1606.7
A_2	99.8	**4.2**	**1175.1**
M	97.4	16.4	3526.4
M_{MIPS}	100.0	8.8	3337.8

Experimental setup. All experiments were made using 'Rome99' dataset from the 9th Implementation Challenge — Shortest Paths (DIMACS (2006)). The used data is the directed road network around the city of Rome, Italy, and is represented by a weighted digraph G of 3353 vertices and 8870 arcs.

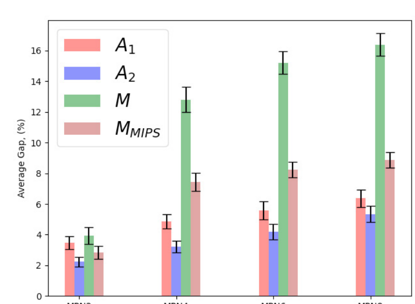

Fig. 2. Average gap with 95% confidence bounds.

For every $k \in \{2, 4, 6, 8\}$, we generated 500 instances on the same graph G with exactly k must-pass nodes. Source,

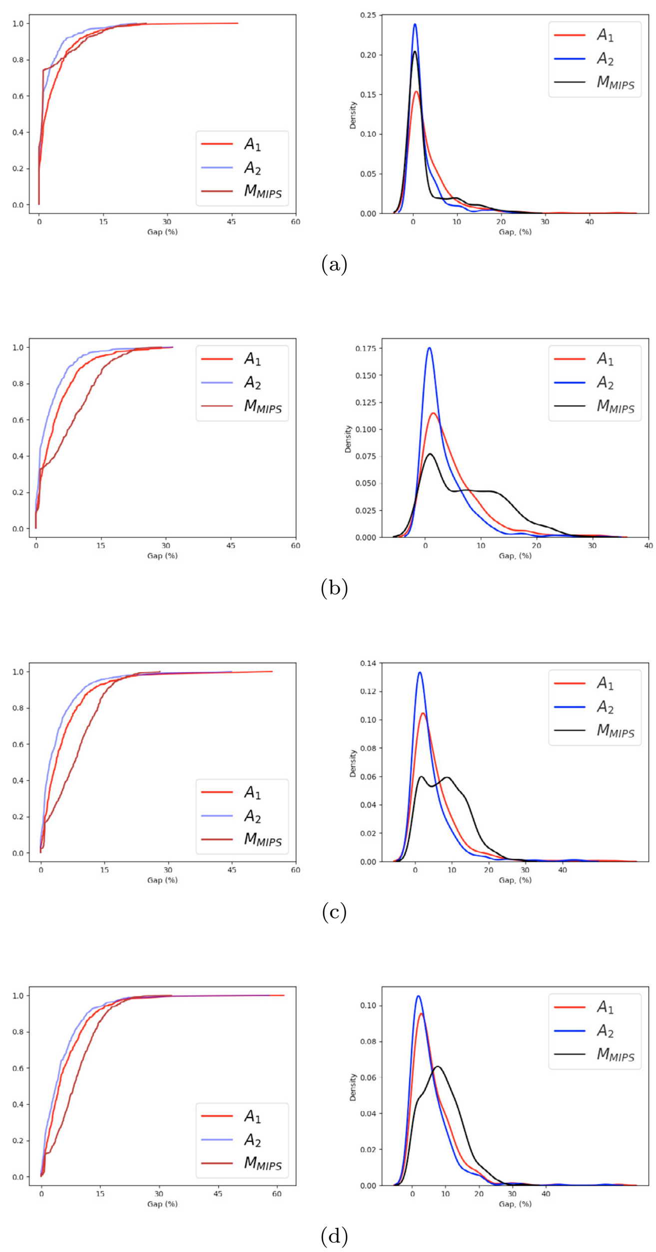

Fig. 3. Empirical distribution functions and densities of random gap obtained by BnB algorithms and Gurobi for: (a) two, (b) four, (c) six, and (d) eight must-pass nodes

destination, and must pass nodes are sampled at random. For any generated instance, we ensure that no must-pass node become a leaf in the graph G.

As baselines, for each instance, we use two runs of the state-of-the-art Gurobi MIP-optimizer (Gurobi Optimization (2021)) equipped with model (1)-(7). In the first run, we call it M, Gurobi is leaved to find a start soluion in its own while, in the second run called M_{MIPS}, we supply the solver by the MIP-start solution provided with our start heuristic.

We establish 1% gap tolerance for all the algorithms, where gap stands for an upper bound for the standard relative error $\text{gap} = (UB - LB)/LB \geq \varepsilon$. Time limits are 1800 sec for our algorithms and 3600 sec for Gurobi, respectively. Computational platform is Intel(R) CPU 4 × 2.60 Ghz 8 Gb RAM with Centos 9 Linux OS.

Results. Table 1 and Fig. 2 represent the obtained results. We denote basic version of our algorithm as A_1 while the improved version is referred to as A_2. The first column represents the ratio of found feasible solution for each method. The second column contains the average gaps for all competitive methods, where UB is the value of the best found feasible solution while LB is the best lower bound. Finally, the third column reflects the average computation time for all proposed methods.

Table 2. Probability distributions of random gaps for MPN values 2 and 4

	gap percentiles (%)					
	# of must-pass nodes					
	2			**4**		
α level (%)	A_1	A_2	M_{MIPS}	A_1	A_2	M_{MIPS}
10	0.0	0.0	0.0	0.2	0.0	0.3
20	0.1	0.0	0.0	0.8	0.3	0.9
30	0.5	0.2	0.0	1.4	0.7	1.0
40	1.0	0.5	0.6	2.3	0.9	4.4
50	1.6	0.8	0.8	3.3	1.6	6.4
60	2.6	1.0	0.9	4.4	2.6	8.8
70	4.1	2.1	1.0	5.9	3.9	11.3
80	5.8	**3.6**	5.1	8.0	**5.6**	13.5
90	9.1	5.8	10.4	11.2	8.5	16.3
100	46.4	22.8	25.0	31.5	31.5	29.0

Table 3. Probability distributions of random gaps for MPN values 6 and 8

	gap percentiles (%)					
	# of must-pass nodes					
	6			**8**		
α level (%)	A_1	A_2	M_{MIPS}	A_1	A_2	M_{MIPS}
10	0.6	0.2	0.9	0.8	0.5	1.0
20	1.0	0.7	2.1	1.7	1.0	3.7
30	1.9	1.0	4.4	2.7	1.7	5.5
40	2.6	1.5	6.2	3.6	2.7	7.0
50	3.7	2.2	8.0	4.6	3.7	8.4
60	5.0	3.5	9.4	5.8	4.8	9.9
70	6.3	4.7	11.3	7.9	6.5	11.7
80	8.5	**6.6**	13.4	10.11	**8.5**	13.7
90	12.3	9.9	15.6	12.6	11.6	16.4
100	54.4	44.8	28.0	61.7	58.0	33.0

As can be seen from Table 1, both proposed algorithms find a feasible solution more than in 99% instances. These presented results also show that our methods have a notably better approximation ratio compared to Gurobi, even if it is provided by a MIP start solution.

To perform the more accurate analysis, we estimate empirical distributions and densities for random gaps provided by all the competitive algorithms. For the sake of brevity, we report these results only for best performers. The results are summarised in Table 2-3 and illustrated in Figures 3a-3d. In particular, these data helps us, for any algorithm, to establish guaranteed accuracy by the formula

$$P\,(\text{gap} \leq \textit{threshold}) \geq \alpha.$$

For instance, with confidence level $\alpha = 80\%$ for 4 must-pass nodes, algorithm A_1 finds an approximate solution with guaranteed gap at most 8.0%, algorithm A_2 — 5.6%, while Gurobi with MIP-start solution — only 13.5%.

6. CONCLUSION

In this paper, we present two versions of novel Branch-and-Bound algorithm for the Protected Shortest Simple Path Problem with a fixed number of Must-Pass Nodes. The numerical experiments show that the proposed algorithms notably outperform the state-of-the-art MIP-solver Gurobi, even in the case, when it is supplied with the same MIP-start solution. In future work, we plan to apply the proposed methods to a significantly larger network and to extend our BnB approach to similar problems.

ACKNOWLEDGEMENTS

Yuri Ogorodnikov, Roman Rudakov, and Michael Khachay were supported by Ural Mathematical Center and the Ministry of Science and Higher Education of the Russian Federation (Agreement no. 075-02-2022-874).

REFERENCES

Andrade, R.C.d. (2016). New formulations for the elementary shortest-path problem visiting a given set of nodes. *European Journal of Operational Research*, 254(3), 755–768. doi:10.1016/j.ejor.2016.05.008.

Björklund, A. and Husfeldt, T. (2014). Shortest two disjoint paths in polynomial time. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias (eds.), *Automata, Languages, and Programming*, 211–222. Springer Berlin Heidelberg, Berlin, Heidelberg.

Cholda, P., Mykkeltveit, A., Helvik, B., Wittner, O., and Jajszczyk, A. (2008). A survey of resilience differentiation frameworks in communication networks. *Communications Surveys & Tutorials, IEEE*, 9, 32 – 55. doi:10.1109/COMST.2007.4444749.

Cholda, P., Tapolcai, J., Cinkler, T., Wajda, K., and Jajszczyk, A. (2009). Quality of resilience as a network reliability characterization tool. *IEEE Network*, 23, 11–19. doi:10.1109/MNET.2009.4804331.

Datta, S., Iyer, S., Kulkarni, R., and Mukherjee, A. (2018). Shortest k-disjoint paths via determinants. In *FSTTCS 2018s*, volume 122 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 19:1–19:21. doi:10.4230/LIPIcs.FSTTCS.2018.19.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271. doi:10.1007/BF01386390.

DIMACS (2006). 9th DIMACS Implementation Challenge – Shortest Paths. URL `http://users.diag.uniroma1.it/challenge9`. Last accessed on Jan. 14, 2022.

Dreyfus, S. (1969). An appraisal of some shortest path algorithm. *Oper. Res.*, 17, 395–412. doi:10.1287/opre.17.3.395.

Fortune, S., Hopcroft, J., and Wyllie, J. (1980). The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2), 111 – 121. doi:10.1016/0304-3975(80)90009-2.

Gomes, T., Marques, S., Martins, L., Pascoal, M., and Tipper, D. (2015). Protected shortest path visiting specified nodes. doi:10.1109/RNDM.2015.7325218.

Gomes, T., Martins, L., Ferreira, S., Pascoal, M., and Tipper, D. (2017). Algorithms for determining a node-disjoint path pair visiting specified nodes. *Optical Switching and Networking*, 23. doi:10.1016/j.osn.2016.05.002.

Gurobi Optimization, L. (2021). Gurobi optimizer reference manual. URL `https://www.gurobi.com/documentation/9.5/refman/index.html`.

Gutin, G. and Punnen, A.P. (2007). *The Traveling Salesman Problem and Its Variations*. Springer US, Boston, MA.

Ibaraki, T. (1973). Algorithms for obtaining shortest paths visiting specified nodes. *SIAM Review*, 15(2), 309–317. URL `http://www.jstor.org/stable/2028603`.

Karp, R. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, 40, 85–103. doi:10.1007/978-3-540-68279-0_8.

Khachai, M.Y. and Dubinin, R.D. (2017). Approximability of the vehicle routing problem in finite-dimensional euclidean spaces. *Proceedings of the Steklov Institute of Mathematics*, 297(1), 117–128. doi:10.1134/S0081543817050133.

Khachay, M. and Neznakhina, K. (2018). *Towards Tractability of the Euclidean Generalized Traveling Salesman Problem in Grid Clusters Defined by a Grid of Bounded Height*, volume 871 of *Communications in Computer and Information Science*, 68–77. Springer International Publishing, Cham. doi:10.1007/978-3-319-93800-4_6.

Khachay, M. and Neznakhina, K. (2020). Complexity and approximability of the Euclidean Generalized Traveling Salesman Problem in grid clusters. *Annals of Mathematics and Artificial Intelligence*, 88(1), 53–69. doi:10.1007/s10472-019-09626-w.

Khachay, M., Ogorodnikov, Y., and Khachay, D. (2021). Efficient approximation of the metric CVRP in spaces of fixed doubling dimension. *Journal of Global Optimization*, 80, 679–710. doi:10.1007/s10898-020-00990-0.

Kudriavtsev, A., Khachay, D., Ogorodnikov, Y., Ren, J., Shao, S.C., Zhang, D., and Khachay, M. (2021). The Shortest Simple Path Problem with a Fixed Number of Must-Pass Nodes: a problem-specific branch-and-bound algorithm. volume 12931 of *Lecture Notes in Computer Science*, 198–210. doi:10.1007/978-3-030-92121-7_17.

Martins, L., Gomes, T., and Tipper, D. (2017). Efficient heuristics for determining node-disjoint path pairs visiting specified nodes. *Networks*, 70(4), 292–307. doi:10.1002/net.21778.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183, 483–523. doi:10.1007/s10107-020-01523-z.

Rak, J. (2015). *Resilient Routing in Communication Networks*. Computer Communications and Networks. Springer. doi:10.1007/978-3-319-22333-9.

Saksena, J.P. and Kumar, S. (1966). The routing problem with 'k' specified nodes. *Operations Research*, 14(5), 909–913.

SNDLib (2022). Sndlib: the data source for all people working on optimization of telecommunications networks. URL `http://sndlib.zib.de/home.action`.

Su, Z., Zhang, J., and Lu, Z. (2019). A multi-stage metaheuristic algorithm for shortest simple path problem with must-pass nodes. *IEEE Access*, PP, 1–1. doi:10.1109/ACCESS.2019.2908011.