RAIRO-Oper. Res. 57 (2023) 87–97
https://doi.org/10.1051/ro/2022215

RAIRO Operations Research
www.rairo-ro.org

THE WIDEST k-SET OF DISJOINT PATHS PROBLEM

Marco Antônio Ribeiro[1],
Iago Augusto Carvalho[2,*] and Armando Honório Pereira[1]

Abstract. Finding disjoint and widest paths are key problems in telecommunication networks. In this paper, we study the Widest k-set of Disjoint Paths Problem (WKDPP), an $\mathcal{NP}$-Hard optimization problem that considers both aspects. Given a digraph $G = (N, A)$, WKDPP consists of computing k arc-disjoint paths between two nodes such that the sum of its minimum capacity arcs is maximized. We present three mathematical formulations for WKDPP, a symmetry-breaking inequality set, and propose two heuristic algorithms. Computational experiments compare the proposed heuristics with another from the literature to show the effectiveness of the proposed methods.

Mathematics Subject Classification. 68R01, 90C11, 90C17, 90C47, 90C59.

Received July 25, 2022. Accepted December 18, 2022.

1. Introduction

The problem of finding multiple disjoint paths was widely studied in telecommunication networks [1–6]. Several network protocols route through multiple disjoint paths. Examples of such protocols are the *Ad hoc* On-demand Multipath Distance Vector Routing Protocol (AOMDV) [7] and the Multipath Transmission Control Protocol (MTCP) [8]. One can use disjoint paths as fault-tolerant mechanisms by enabling alternative routing paths when a route link fails [9]. In addition, multiple disjoint paths can act as load-balancing structures, distributing network packets among several routes. Thus, it is possible to increase the network throughput and avoid link congestion [10].

An important characteristic of such paths is their *bandwidth*, *i.e.*, the number of network packets able to flow through the path in a given time span [11]. We consider the bandwidth of a path p as the smallest link capacity in p. A path between two network devices s and t is the *widest* if it has the greatest bandwidth among all possible paths. The wider the path, then more packets can be sent through it. Therefore, the throughput of a network is directly correlated to the width of its routing paths [12].

The coupled version of the *Widest Pair of Disjoint Paths* (WPDPC) [13] is a $\mathcal{NP}$-Hard optimization problem that combines the notion of disjoint and widest paths. Let $G = (N, A)$ be a digraph with node set N and arc set A. Moreover, a function $f : A \mapsto \mathbb{R}_{>0}$ associates each arc $(i, j) \in A$ to a capacity c_{ij}. The WPDPC problem

Keywords. Widest paths, disjoint paths, network flows, integer programming, heuristics.

[1] Computer Science Department, Universidade Federal de Minas Gerais, Av. Pres. Antônio Carlos 6627, Belo Horizonte, MG 31270-901, Brazil.

[2] Computer Science Department, Universidade Federal de Alfenas, Av. Jovino Fernandes de Sales 2600, Alfenas, MG 37133-840, Brazil.

*Corresponding author: iago.august@gmail.com; iago.carvalho@unifal-mg.edu.br

aims to find two arc-disjoint paths between a source node $s \in N$ and a sink node $t \in N$, such that the sum of their widths is maximized.

This work studies a generalization of the WPDPC, called *Widest k-set of Disjoint Paths Problem* (WKDPP), which was also studied by Wang *et al.* [14]. Given $G = (N, A)$, s, and t, as previously defined, the WKDPP aims to find a set $H = \{p_1, p_2, \ldots, p_k\}$ of k arc-disjoint paths from s to t such that the sum of their widths is maximized, *i.e.*,

$$\max \sum_{h=1}^{k} \min_{(i,j)\in p_h} c_{ij}.$$

Since the WKDPP is a generalization of the WPDPC for k paths, it is also $\mathcal{NP}$-Hard.

The WKDPP relates to other network flow problems, such as the Origin-Destination Integer Multicommodity Flow problem [15], the Concurrent k-Splittable Flow Problem [16,17], the Unsplittable Flow Problem [18], and the Maximum Disjoint Paths with Different Colors [19]. It differs from the k-splittable Flow Problem [20,21] by forbidding two flows to pass through the same arc. It also relates to the Shortest Pair of Disjoint Paths [22] and other disjoint-path problems [23–25].

In this paper, we present three Integer Linear Programming (ILP) formulations for the WKDPP. The first model is an arc-path formulation that has a potentially exponential number of elementary $\langle s, t\rangle$ paths. The second model is a generalization of the ILP multicommodity flow formulation for the WPDPC, proposed in [13], for the case of k disjoint paths. The third model is a reformulation of the second one as another ILP multicommodity flow formulation. It uses two flow conservation constraints to compute the set of widest disjoint paths. In addition, we propose two heuristics for the WKDPP. The first heuristic rely on several executions of the Ford–Fulkerson max-flow algorithm [26]. The second one is a fix-and-optimize heuristic that removes a subset of arcs from A and optimizes the remaining model. One can extend the proposed heuristics to other $\mathcal{NP}$-Hard network flow problems, such as the k-Splittable Maximum Flow Problem [20] and the Unsplittable Flow Problem [18]. The results obtained by our heuristics are contrasted with those of the Maximum K-Path Bandwidth Algorithm (MKPB), a heuristic proposed by Wang *et al.* [14].

We organize the remainder of this paper as follows. Section 2 proposes three ILP formulations for the WKDPP. Section 3 presents the WKDPP heuristics. Section 4 reports the computational experiments of our work. Finally, the last section draw the concluding remarks of this paper.

2. Integer linear programming mathematical models

2.1. Arc-path model

The first model is an arc-path formulation. Let $\mathcal{P}$ be the potentially exponential set of all the elementary $\langle s, t\rangle$ paths in G. Besides, let $H = \{p_1, \ldots, p_k\} \subseteq \mathcal{P}$ be a set of k arc-disjoint paths that carry out flow from s to t and $u_p = \min_{(i,j)\in p} c_{ij}$ be the width of path $p \in \mathcal{P}$. We define $\delta_{ij}^p = 1$ if arc $(i, j) \in A$ belongs to the path $p \in \mathcal{P}$ and $\delta_{ij}^p = 0$ otherwise. Let x be a binary decision vector such that $x_p = 1$ if path $p \in \mathcal{P}$ belongs to the WKDPP solution and $x_p = 0$ otherwise. We define this model as follows:

$$\max \sum_{p\in\mathcal{P}} u_p x_p \tag{1}$$

s.t.

$$\sum_{p\in\mathcal{P}} \delta_{ij}^p \leqslant 1 \qquad \forall (i,j) \in A \tag{2}$$

$$\sum_{p\in\mathcal{P}} x_p = k \tag{3}$$

$$x_p \in \{0, 1\} \qquad \forall p \in \mathcal{P}. \tag{4}$$

Objective function (1) maximizes the total flow between s and t. Inequalities (2) guarantee that only one path $p \in \mathcal{P}$ uses an arc $(i,j) \in A$. Inequality (3) selects $|H|$ paths to send flow from s to t. Constraints (4) define the domain of variables x_p. This model relies on a potentially exponential number of variables x_p. Therefore, it cannot be properly solved by an ILP solver without a reformulation.

2.2. Arc-node model

Shen *et al.* [13] proposed an arc-node model for WPDPC. In this work we generalize his model for the case of k disjoint paths. We define $x_{ij}^p = 1$ if the arc $(i,j) \in A$ belongs to path $p \in H$, and $x_{ij}^p = 0$ otherwise. Besides, variable y_p stores the width of path p. We define this model as follows:

$$\max \sum_{p \in H} y_p \tag{5}$$

s.t.

$$\sum_{(j,l)\in A} x_{jl}^p - \sum_{(i,j)\in A} x_{ij}^p = \begin{cases} 1, & \text{if } j = s \\ -1, & \text{if } j = t \\ 0, & \text{otherwise} \end{cases} \qquad \forall j \in V,\ p \in H \tag{6}$$

$$\sum_{p \in H} x_{ij}^p \leqslant 1 \qquad \forall (i,j) \in A \tag{7}$$

$$y_p \leqslant c_{ij} x_{ij}^p + M\left(1 - x_{ij}^p\right) \qquad \forall (i,j) \in A, p \in H \tag{8}$$

$$x_{ij}^p \in \{0,1\} \qquad \forall (i,j) \in A, p \in H \tag{9}$$

$$y_p \geqslant 0 \qquad \forall p \in H. \tag{10}$$

The objective function (5) maximizes the sum of the path widths. Constraints (6) are the classical flow balance constraints and ensure the connectivity of the k paths from s to t. Inequalities (7) guarantee that only one $\langle s,t \rangle$ path uses the arc $(i,j) \in A$. Constraints (8) compute the width of each path. It uses a constant M equals to the highest arc capacity in G. Constraints (9) and (10) respectively define the domain of variables x_{ij}^p and y_p.

2.3. Reformulated arc-node model

The third model is a reformulation of the previous arc-node model. It describes each path as a flow subproblem. The introduction of a second flow variable $z_{ij}^p \geqslant 0$ removes the necessity of Inequalities (8), thus eliminating the big M constant. The second flow constraints, that use variables z_{ij}^p, is used to compute the width of path p. Let x_{ij}^p and y_p be as previously defined. We define this model as constraints (11)–(18).

Objective function (11), constraints (13), and (14) are the same as in the first arc-node model. Inequalities (12), together with (13), define each flow subproblem and guarantee that the flow is unsplittable and acyclic. Constraints (15) couple flow variables x_{ij}^p and z_{ij}^p. It induces variable z_{ij}^p to assume the capacity of arc (i,j) if it belongs to path $p \in H$. Finally, constraints (16)–(18) respectively define the domain of variables x_{ij}^p, y_p and z_{ij}^p.

$$\max \sum_{p \in H} y_p \tag{11}$$

s.t.

$$\sum_{(j,l)\in A} z_{jl}^p - \sum_{(i,j)\in A} z_{ij}^p = \begin{cases} y_p, & \text{if } j = s \\ -y_p, & \text{if } j = t \\ 0, & \text{otherwise} \end{cases} \qquad \forall j \in V,\ p \in H \tag{12}$$

$$\sum_{(j,l)\in A} x^p_{jl} - \sum_{(i,j)\in A} x^p_{ij} = \begin{cases} 1, & \text{if } j = s \\ -1, & \text{if } j = t \\ 0, & \text{otherwise} \end{cases} \qquad \forall j \in V,\ p \in H \tag{13}$$

$$\sum_{p\in H} x^p_{ij} \leqslant 1 \qquad \forall (i,j) \in A \tag{14}$$

$$z^p_{ij} - c_{ij} x^p_{ij} \leqslant 0 \qquad \forall (i,j) \in A, p \in H \tag{15}$$

$$x^p_{ij} \in \{0,1\} \qquad \forall (i,j) \in A, p \in H \tag{16}$$

$$y_p \geqslant 0 \qquad \forall p \in H \tag{17}$$

$$z^p_{ij} \geqslant 0 \qquad \forall (i,j) \in A, p \in H. \tag{18}$$

2.4. Symmetry-breaking constraints

Both arc-node models have a strong symmetry structure induced by the decision variables' assignment. Let $H^* = \{p^*_1, p^*_2, \ldots, p^*_k\}$ be the optimal set of k paths for a WKDPP instance whose respective widths are $\{y^*_1, y^*_2, \ldots, y^*_k\}$. Note that any permutation of those k paths leads to another optimal solution with the same objective value. Therefore, $k!$ different optimal solutions exist as permutations of those k paths.

This symmetric structure can be avoided by introducing a set of variable-ordering constraints [27], as stated by Inequalities (19). It imposes path p_i to be, at least, as *wider* as path $p_{(i+1)}$, thus avoiding the permutation of the solutions. However, it cannot prevent situations where the optimal solution has two or more paths p_i and p_j with the same width, *i.e.*, $y_i = y_j$.

$$y_{(i+1)} - y_i \leqslant 0 \qquad 1 \leqslant i < k. \tag{19}$$

3. Heuristics for the WKDPP

In this section, we propose two heuristics for the WKDPP. The first executes the Ford–Fulkerson max-flow algorithm several times cleaning arcs, leaving the graph containing exactly k disjoint paths. It relies on several executions of the Ford–Fulkerson max-flow algorithm [26]. The second one is a fix-and-optimize heuristic that removes a subset of arcs from A and optimizes the remaining model.

Given a WKDPP instance, the first algorithm heuristically removes all arcs with a capacity smaller than a minimum value computed and then solves an ILP model with the remaining arc subset. The second heuristic inspects all arcs separately, in ascending order according to their capacity. It checks if removing an arc leads to an infeasible solution. If true, then it reinserts the arc into the instance. Otherwise, the arc is definitely removed. After inspecting all arcs, the remaining arc set results exactly in k disjoint paths.

3.1. Maximum Flow-Based Algorithm

The Maximum Flow-Based Algorithm (MFBA) is a $\mathcal{O}(k|A|^2)$ heuristic for the WKDPP that runs $|A|$ times the Ford–Fulkerson maximum flow algorithm [26]. It aims at removing low-capacity arcs from A in order to compute a flow decomposition of G into k disjoint paths with maximum width. In order to reduce the complexity of this heuristic, the Ford–Fulkerson algorithm is partially executed. Instead of computing the total flow of the graph, it stops when a flow of at least k units is found.

Algorithm 1 presents the MFBA heuristic. It receives as input G, s, t, and k. Initially, line 1 generates an arc set A' as an exact copy of A. Lines 2, 3 set the capacity of all arcs in A' to 1. The algorithm uses these capacities to compute the maximum number of disjoint $\langle s,t\rangle$ paths in $G' = (N, A')$. Lines 4–8 iterate over all arcs (i,j) in A sorted in ascending order according to their capacity. Line 5 removes the arc (i,j) from A'. Then, it computes the maximum flow in A' (mf) by partially running the Ford–Fulkerson algorithm, where it tries to find k augmenting paths, resulting in a complexity of $\mathcal{O}(k|E|)$. If the resulting maximum flow is smaller than k (line 7), then arc (i,j) cannot be removed from A', otherwise the instance becomes unfeasible. Thus, line 8 reinserts the arc in A'. Thereafter, only the arcs from the k paths remain. Next, it executes a flow decomposition

Algorithm 1: Maximum Flow-Based Algorithm (MFBA).

```
   input  : G, s, t, k
   output: set containing k disjoint paths
1  A' ← A
2  foreach (i, j) ∈ A' do
3  |  c_ij ← 1
4  foreach (i, j) ∈ sorted(A) do
5  |  A' ← A' \ { (i, j) }
6  |  mf ← ford-fulkerson(A', k)
7  |  if mf < k then
8  |  |  A' ← A' ∪ { (i, j) }
9  return flow-decomposition(A')
```

Algorithm 2: Fix-and-Optimize Heuristic.

```
   input  : G, s, t, k
   output: A' ⊆ A
1  A' ← A
2  min-arc ← −∞
3  H ← flow-decomposition(A')
4  while |H| ⩾ k do
5  |  min-arc ← min_{(i,j)∈p}(c_ij) ∀p ∈ H
6  |  A' ← A' \ { (i, j) ∈ A' | c_ij ⩽ min-arc }
7  |  H ← flow-decomposition(A')
8  A' ← A \ { (i, j) ∈ A | c_ij < min-arc }
9  return A'
```

algorithm [28] on A' to compute a feasible solution for the WKDPP. Such flow decomposition returns exactly k disjoint paths.

3.2. Fix-and-Optimize Heuristic

This Fix-and-Optimize Heuristic (F&OH) is a $\mathcal{O}(k|A|^2)$ heuristic preprocessing algorithm for the WKDPP. Given a WKDPP instance, it aims at computing a reduced arc set $A' \subseteq A$ to be inserted into an ILP mathematical model, thus speeding-up its resolution. The algorithm searches for a maximum capacity value such that all arcs with lower capacity can be removed without greatly affecting the optimal solution.

Algorithm 2 states the F&OH. It receives as input G, s, t, and k. Initially, it generates an arc set A' as an exact copy of A (line 1) and creates a variable to store the maximal minimum arc capacity value such that arcs with smaller capacity can be removed while keeping k disjoint paths (line 2). Then, it performs a flow decomposition in A', considering that each arc capacity is 1, and stores the paths found in H. While there are at least k disjoint paths in A', it computes the minimum arc in H and removes every arc with capacity lesser or equal than to it from A' (lines 5–7). Finally, it generates a new arc set A' by removing from A every arc with capacity smaller than the computed min-arc value (line 8). We use the computed arc set A' as input to an ILP solver, using any arc-based model for the WKDPP.

4. Computational experiments

We performed the computational experiments on a single core of an Intel Xeon CPU E5645 with 2.4 GHz and 32 GB of RAM, running under the operating system Ubuntu Server 14.04. We used the *branch-and-bound*

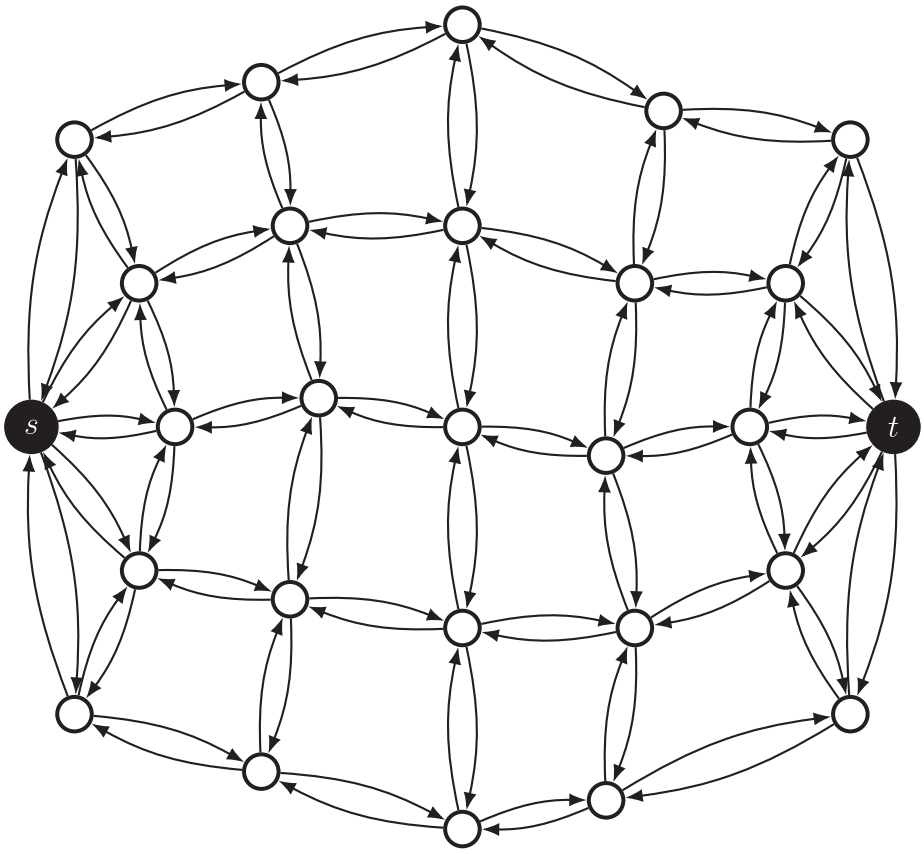

FIGURE 1. An example of a Transit Grid Graph with 27 nodes and 100 arcs.

implementation of the ILOG CPLEX version 12.6 with default parameter settings to solve the proposed models and the F&OH remaining model. To implement the heuristics, we used C++, alongside with GNU g++ 6.2 to compile. We limited the running time of all algorithms to 7200 s.

Two instance groups were used in the computational experiments. The first instances group are complete graphs with $|N| \in \{100, 200\}$. We used a normal distribution $\mathcal{N}(\mu, \sigma^2)$ to randomly generate the arcs capacities, where $\mu = 100$ and $\sigma^2 = 25$. The second instances group was generated through the Transit Grid Graph generator of G. Waissi and J. Setubal[1]. Those instances are sparse graphs, whose topology are similar to a transportation network, being previously used in the computational experiments of Truffot and Duhamel [21]. Despite being sparse graphs, these instances were built in such a way that the size of the minimum cut-set is always greater or equal to 10 (which is the maximum value of k evaluated in this work), thus ensuring that all instances have, at least, 10 edge-disjoint paths according to Menger's Theorem [29]. Figure 1 presents an example of such topology. Different from the first group, we generated the arc's capacities of such instances by means of an uniform distribution $\mathcal{U}(1, 200)$. This instance group also contains two subsets, one with $|N| = 100$ and the other with $|N| = 200$. Besides, each subset consists of 10 instances. Therefore, we evaluate in our computational experiments a total of 40 instances.

[1] www.informatik.uni-trier.de/~naeher/Professur/research/generators/maxflow/tg/index.html.

TABLE 1. Evaluation of the symmetry-breaking constraint for models ILP_1 and ILP_2.

Name	k	ILP_1			ILP_1^S			ILP_2			ILP_2^S		
		$o/f/u$	gap ± sd (%)	t (s)*	$o/f/u$	gap ± sd (%)	t (s)*	$o/f/u$	gap ± sd (%)	t (s)*	$o/f/u$	gap ± sd (%)	t (s)*
$c100$	2	10/ 0/ 0	0.00(000)	13	10/ 0/ 0	0.00(000)	7	3/ 7/ 0	2.40(022)	5373	4/ 6/ 0	1.90(024)	5906
	4	10/ 0/ 0	0.00(000)	235	10/ 0/ 0	0.00(000)	63	0/10/ 0	4.60(022)	7177	0/10/ 0	5.40(037)	7171
	6	3/ 7/ 0	3.70(033)	6292	10/ 0/ 0	0.00(000)	248	0/10/ 0	4.20(019)	7167	0/10/ 0	5.80(021)	7182
	8	0/10/ 0	9.20(033)	7179	4/ 6/ 0	2.10(023)	4782	0/10/ 0	3.50(014)	7182	0/10/ 0	8.60(023)	7183
	10	0/10/ 0	14.30(047)	7182	2/ 8/ 0	5.40(052)	6720	0/10/ 0	10.70(047)	7182	0/10/ 0	17.20(037)	7182
$c200$	2	10/ 0/ 0	0.00(000)	108	10/ 0/ 0	0.00(000)	75	0/10/ 0	8.30(033)	7155	0/10/ 0	9.70(034)	7139
	4	10/ 0/ 0	0.00(000)	2529	10/ 0/ 0	0.00(000)	708	0/10/ 0	13.10(032)	7183	0/10/ 0	18.20(052)	7183
	6	0/10/ 0	11.80(015)	7160	8/ 2/ 0	1.20(030)	3100	0/ 4/ 6	17.00(230)	7184	0/ 2/ 8	22.00(460)	7184
	8	0/10/ 0	15.20(027)	7177	2/ 8/ 0	6.40(055)	6697	0/ 3/ 7	14.80(230)	7183	0/ 1/ 9	27.00(870)	7183
	10	0/10/ 0	22.00(052)	7181	0/10/ 0	11.00(052)	7182	0/ 0/10	–**	–**	0/ 0/10	–**	–**
$g100$	2	10/ 0/ 0	0.00(000)	2	10/ 0/ 0	0.00(000)	1	4/ 6/ 0	20.00(240)	5008	4/ 6/ 0	12.10(180)	4670
	4	10/ 0/ 0	0.00(000)	506	10/ 0/ 0	0.00(000)	11	0/10/ 0	54.20(250)	7186	0/10/ 0	47.60(230)	7186
	6	1/ 9/ 0	66.40(370)	6668	10/ 0/ 0	0.00(000)	407	0/10/ 0	55.90(120)	7186	0/10/ 0	72.30(230)	7186
	8	0/10/ 0	118.40(380)	7186	8/ 2/ 0	5.60(140)	3300	0/10/ 0	60.40(180)	7187	0/10/ 0	94.50(350)	7187
	10	0/10/ 0	131.60(420)	7186	0/10/ 0	65.20(230)	7187	0/10/ 0	66.80(230)	7187	0/10/ 0	106.10(360)	7187
$g200$	2	10/ 0/ 0	0.00(000)	6	10/ 0/ 0	0.00(000)	2	0/10/ 0	40.00(180)	7186	0/10/ 0	37.00(210)	7186
	4	6/ 4/ 0	17.00(250)	5023	10/ 0/ 0	0.00(000)	187	0/10/ 0	107.90(190)	7187	0/10/ 0	103.20(190)	7187
	6	0/10/ 0	112.00(280)	7138	8/ 2/ 0	7.70(170)	2178	0/10/ 0	136.00(330)	7187	0/10/ 0	203.50(750)	7187
	8	0/10/ 0	162.70(430)	7187	0/10/ 0	83.20(360)	7187	0/10/ 0	152.10(420)	7188	0/10/ 0	250.80(1100)	7188
	10	0/10/ 0	361.60(950)	7188	0/10/ 0	246.40(740)	7188	0/10/ 0	194.90(500)	7188	0/10/ 0	307.30(700)	7188

Notes. (*)Average time in seconds with respect to optimal and feasible solutions. (**)We found no feasible solution for this instance set.

4.1. Mathematical models evaluation

Table 1 shows the computational results of both models with and without the symmetry-breaking constraints (19). For the sake of brevity, the models presented in Sections 2.2 and 2.3 are respectively referred to as ILP_1 and ILP_2. Besides, the ILP_1 with such constraints is denoted as ILP_1^S. Similarly, the ILP_2 with the symmetry-breaking constraint is denoted as ILP_2^S.

The first column of Table 1 represents the instance subset name. Each subset contains 10 randomly generated instances with the same characteristics. The instance subset names are represented as a tuple T–N, where T denotes the instance topology (c for complete graphs and g for transit grid graphs) and N denotes the number of nodes. The second column displays the number k of disjoint paths. The third, fourth and fifth columns refer to the ILP_1. The third column displays a triple $o/f/u$ with the number of instances solved at optimality (o), the number of instances with feasible solutions (f), and the number of instances without any integer solution found within 7200 s (u). The fourth column displays the average integrality gap computed by CPLEX and its standard deviation. The fifth column shows the model's average running time. The following columns show the same information for ILP_1^S, ILP_2, and ILP_2^S. We only take into account the relative gap and running time of the instances with optimal or feasible solutions. Therefore, we symbolize the gap and running time of instance subsets without feasible solutions as a dash.

4.1.1. Comparison between ILP_1 and ILP_2

The first experiment compares the execution of ILP_1 and ILP_2 on the proposed instance set. Table 1 shows that ILP_1 found smaller gaps than ILP_2 for complete graphs, both with 100 and 200 nodes. One can see that ILP_1 found optimal solutions for all complete graph instances with $k = 2$ and $k = 4$, while ILP_2 could not properly deal with instances with $k \geqslant 4$. Besides, ILP_2 was not able to find integer solutions for complete graphs with 200 nodes and $k = 10$. These results may be due to the great number of additional variables and constraints of model ILP_2, which introduced a new flow variable (z_{ij}^p) and a new set of flow constraints. Consequently, it was not able to find smaller gaps as ILP_1 within the time limit for complete graphs.

On the other hand, ILP_2 performs better than ILP_1 for sparse graphs and large k values, as measured by the computational experiments with instances $g100$ and $g200$. It found smaller gaps than ILP_1 for instances $g100$ with $k \in \{6, 8, 10\}$ and for instances $g200$ with $k = 8$ and $k = 10$. As $g100$ and $g200$ are sparse graphs, they do

not induce a great number of additional variables and constraints on model ILP_2 over ILP_1. Consequently, the strategy used in this model to avoid the big M constant showed to be partially effective.

4.1.2. Effectiveness of the symmetry-breaking constraints

The second experiment aims at comparing the effectiveness of the symmetry-breaking constraints (19) over the models ILP_1 and ILP_2. One can see from Table 1 that the introduction of such constraints significantly improves the first ILP model since ILP_1^S found a greater number of optimal solutions than ILP_1. Besides, the ILP_1^S running time is smaller than ILP_1 for all instances with $k \leqslant 6$. The introduction of constraints (19) also enabled ILP_1^S to reduce the relative integrality gap over ILP_1 for all not optimal instances.

These same results do not hold for the second ILP model. One can see that the introduction of constraints (19) negatively affects model ILP_2. The model without the symmetry-breaking constraints performs better than its constrained version for all instances when $k \geqslant 6$, achieving less than half of the ILP_2^S integrality gap in some cases. On the other hand, ILP_2^S finds smaller gaps than ILP_2 for both complete and sparse instances with $k \leqslant 4$. Such results might be due the CPLEX scheme for exploring its branch-and-bound tree. When exploring deeper levels of the tree, a great number of variables are fixed. Therefore, the solutions found at these levels have less room to be modified. As the introduction of constraints (19) highly constrain the problem, the search tree is most frequently re-initiated on ILP_2^S than in ILP_2. Therefore, ILP_2^S achieves worse results than ILP_2. The same behavior does not occur with ILP_1 and ILP_1^S, since ILP_1 has just one flow sub-problem, while ILP_2 is based on k flow sub-problems and a greater number of variables.

4.2. Heuristics evaluation

The last experiment evaluates the heuristics proposed in Section 3, namely the MFBA and the F&OH. It also contrasts the results of the above-mentioned heuristics with those of the MKPB of Wang *et al.* [14], which is the most prominent heuristic for solving WKDPP in the literature. Tables 2 and 3 show the computational results of this experiment. The F&OH uses ILP_1^S to solve the remaining model since it has the best average results in comparison to the other ILP models, as shown in Table 1.

Table 2 shows the results of the three heuristics. The first column represents the instance subset name. The second column displays the number k of disjoint paths computed. The third, fourth, and fifth columns refer to the MKPB. The third column shows the relative deviation of MKPB over the ILP_1^S primal solution, computed as $(\text{ILP} - H)/\text{ILP}$, where H denotes the value given by the heuristic and ILP denotes the ILP_1^S primal solution value. The fourth column denotes the standard deviation of the MKPB results shown in the previous column. The fifth column reports the heuristic's average running time. Next, the sixth, seventh, and eighth columns give the same information for the F&OH, and the last three columns show the same data for the MFBA heuristic.

Table 3 reports the arcs set average reduction for each instance subgroup. The first column displays the number of disjoint paths. The second and third column refers to instances $c100$. The second column shows the number of arcs in the original instance, while the third column displays the average arc set reduction for the 10 instances within this subgroup. The remaining columns show the same information for instances $c200$, $g100$, and $g200$.

One can see from Table 2 that the F&OH has greater running times, being stopped by the maximum running time of 7200 s in some cases. This is due to the resulting optimization model size. The heuristic was able to greatly reduce the arc set of the complete graph instances, as shown in Table 3. However, F&OH cannot significantly reduce the arc set of the sparse graphs, especially for large values of k. Despite its resulting arc set size, the reduction approach was effective in providing better solutions than ILP_1^S. The F&OH found optimal solutions for complete graphs with $k \leqslant 4$ and for sparse graphs with 200 nodes and $k \leqslant 4$. Besides, its running time is smaller than ILP_1^S for such instances. In addition, it improved the ILP_1^S primal solution by more than 17% in $g200$ instances with 10 disjoint paths.

The MFBA and the MKPB heuristics have a completely different behavior than the F&OH. One can see that both MFBA and MKPB running times are much smaller than that of F&OH, in such a way that the MFBA's running time never exceeds half of a second, while that of MKPB never exceeds 1.02 s. Moreover, both

TABLE 2. Evaluation of the heuristics for the WKDPP.

Name	k	MKPB [14]			F&OH			MFBA		
		dev (%)	sd (%)	time (s)	dev (%)	sd (%)	time (s)	dev (%)	sd (%)	time (s)
$c100$	2	0.1	0.03	0.05	0.0	0.00	0.11	0.1	0.02	0.02
	4	0.4	0.05	0.09	0.0	0.00	1.05	0.3	0.03	0.04
	6	0.3	0.03	0.15	0.0	0.00	91.32	0.3	0.02	0.06
	8	0.3	0.06	0.19	−0.4	0.05	751.46	0.2	0.05	0.08
	10	−1.1	0.48	0.25	−2.1	0.34	4348.96	−1.4	0.35	0.11
$c200$	2	0.3	0.02	0.29	0.0	0.00	0.39	0.3	0.03	0.12
	4	0.1	0.03	0.45	0.0	0.00	3.54	0.2	0.03	0.19
	6	0.1	0.15	0.63	−0.7	0.21	313.02	−0.2	0.18	0.25
	8	−2.4	0.54	0.74	−3.4	0.39	3357.92	−2.9	0.37	0.33
	10	−4.7	0.71	1.02	−5.6	0.41	7187.84	−5.4	0.40	0.43
$g100$	2	2.4	0.31	0.03	0.9	0.23	0.17	2.5	0.28	0.01
	4	11.3	1.12	0.05	0.6	0.19	1.90	7.5	0.58	0.02
	6	19.4	1.00	0.08	0.1	0.02	27.08	16.1	0.70	0.03
	8	33.5	1.34	0.10	0.1	0.11	2465.88	32.2	0.48	0.04
	10	63.8	2.59	0.09	−0.9	0.20	7187.51	52.1	0.88	0.04
$g200$	2	1.7	0.35	0.06	0.0	0.00	0.44	1.6	0.31	0.02
	4	9.9	0.90	0.12	0.0	0.00	5.90	6.4	0.62	0.05
	6	19.1	1.05	0.24	0.3	0.11	244.86	15.3	0.54	0.10
	8	26.0	1.97	0.37	−7.3	0.81	6598.75	16.2	0.92	0.15
	10	26.3	2.07	0.54	−17.1	1.40	7187.34	15.5	1.10	0.21

TABLE 3. Effectiveness of F&OH in reducing WKDPP instances.

k	$c100$		$c200$		$g100$		$g200$	
	Arcs	Reduction	Arcs	Reduction	Arcs	Reduction	Arcs	Reduction
2	9900	96.77%	39 800	89.24%	399	39.12%	791	40.65%
4	9900	94.00%	39 800	98.25%	399	25.96%	791	31.37%
6	9900	91.70%	39 800	97.09%	399	17.84%	791	23.43%
8	9900	88.94%	39 800	96.22%	399	8.19%	791	15.34%
10	9900	89.24%	39 800	94.75%	399	0.22%	791	7.06%

heuristics were able to improve the average ILP_1^S primal results, especially when $|N| = 200$, whose average relative deviation was −1.6% for the case of MFBA and −1.32% in the case of MKPB. Despite that, in the case of sparse graphs, neither of the heuristics was able to improve the ILP_1^S solutions. It can be seen that the average relative deviation of MFBA was 22.08% for instances $g100$ and of 11% for instances $g200$, while those of MKBP were 26.08% and 16.6%, respectively, for instances $g100$ and $g200$. Comparatively, it was observed that the average running time of MFBA was always or smaller than that of MKPB, which is expected due to the greater computational complexity of the latter [14]. Besides that, MFBA achieved a smaller average relative deviation than MKPB in 15 out of the 20 instance's subsets. Therefore, these results indicate that MFBA overcomes MKPB both in terms of running time and optimization results.

5. Conclusions

This paper addressed the WKDPP, a $\mathcal{NP}$-Hard maximum flow optimization problem that combines the notions of disjoint and widest paths. This problem finds application in the design of telecommunication networks. We presented three ILP formulations for the WKDPP, along with a strategy for breaking the symmetric structure of these models. The first model needs to enumerate all possible $\langle s, t\rangle$ of the graph. Therefore, it cannot be properly solved by an ILP solver. The others are compact arc-based models. The second model showed to be effective for solving the WKDPP into complete graph instances, while the third model gives better results for large sparse graphs. The symmetric-breaking constraints were demonstrated to be effective in reducing the computational cost of the first compact model.

We also proposed two heuristics for the WKDPP. One of the heuristics clearly outperforms the baseline heuristic for the WKDPP, with an average running time that never exceeds half of a second, and achieves better average results than the ILP models for complete graphs. The other heuristic aims at reducing the arc set size. It proved to be effective for both complete and sparse graphs, reducing the total computational time and improving the average result.

The MFBA heuristics can be extended to other $\mathcal{NP}$-Hard max flow optimization problems, such as the Unsplittable Flow Problem [18] and the k-Splittable Maximum Flow Problem [20]. One can modify the MFBA by changing the decision process if an arc can be removed or not. The Fix-and-Optimize approach can also be extended to other problems by modifying the strategy of computing the min-arc cost.

Future works regarding the WKDPP can focus on developing new heuristics for this problem. These heuristics can use the MFBA as an initial solution since it provides good results in a small computational time. Besides, one can apply a Dantzig–Wolfe decomposition over the arc-path model and build a Branch-and-Price algorithm, using the generation of the disjoint path as a sub-problem, following the strategies of Truffot and Duhamel [21]. The resulting model will be an arc-path model, similar to the first ILP formulation presented in this work.

Acknowledgements. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPq), and the Fundação de Amparo à Pesquisa do Estado de Minas Gerais – Brasil (FAPEMIG).

Conflict of interest/competing interest. The authors declare that they have no conflicts/competing interests.

Availability of data and material. Data and materials can be obtained upon request to the authors.

Code availability. Code can be obtained upon request to the authors.

References

[1] H. Kerivin and A.R. Mahjoub, Design of survivable networks: a survey. *Networks* **46** (2005) 1–21.

[2] P. Cruz, T. Gomes and D. Medhi, A heuristic for widest edge-disjoint path pair lexicographic optimization, in 2014 6th International Workshop on Reliable Networks Design and Modeling (RNDM). IEEE (2014) 9–15.

[3] A. Hou, C.Q. Wu, D. Fang, Y. Wang and M. Wang, Bandwidth scheduling for big data transfer using multiple fixed node-disjoint paths. *J. Network Comput. App.* **85** (2017) 47–55.

[4] L. Liu, J.-T. Zhou, H.-F. Xing and X.-Y. Guo, Flow splitting scheme over link-disjoint multiple paths in software-defined networking. *Concurrency Comput.: Pract. Experience* **34** (2022) e6793.

[5] M. MalekiTabar and A.M. Rahmani, A delay-constrained node-disjoint multipath routing in software-defined vehicular networks. *Peer-to-Peer Networking App.* **15** (2022) 1452–1472.

[6] S. Kettouche, M. Maimour and L. Derdouri, Disjoint multipath RPL for QoE/QoS provision in the internet of multimedia things. *Computing* **104** (2022) 1677–1699.

[7] M.K. Marina and S.R. Das, Ad hoc on-demand multipath distance vector routing. *ACM SIGMOBILE Mobile Comput. Commun. Rev.* **6** (2002) 92–93.

[8] Q. Peng, A. Walid, J. Hwang and S.H. Low, Multipath TCP: analysis, design, and implementation. *IEEE/ACM Trans. Networking* **24** (2016) 596–609.

[9] Y. Challal, A. Ouadjaout, N. Lasla, M. Bagaa and A. Hadjidj, Secure and efficient disjoint multipath construction for fault tolerant routing in wireless sensor networks. *J. Network Comput. App.* **34** (2011) 1380–1397.

[10] Y. Guo, F. Kuipers and P. Van Mieghem, Link-disjoint paths for reliable QoS routing. *Int. J. Commun. Syst.* **16** (2003) 779–798.

[11] J. Kurose and K. Ross, Computer Networking: A Top-down Approach, 7th edition. Pearson Education (2017).
[12] J. Crichigno, W. Shu and M.-Y. Wu, Throughput optimization and traffic engineering in WDM networks considering multiple metrics, in 2010 IEEE International Conference on Communications (ICC). IEEE (2010) 1–6.
[13] B.H. Shen, B. Hao and A. Sen, On multipath routing using widest pair of disjoint paths, in 2004 Workshop on High Performance Switching and Routing. HPSR. IEEE (2004) 134–140.
[14] T. Wang, C.Q. Wu, Y. Wang, A. Hou and H. Cao, Multi-path routing for maximum bandwidth with k edge-disjoint paths, in 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE (2018) 1178–1183.
[15] C. Barnhart, C.A. Hane and P.H. Vance, Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.* **48** (2000) 318–326.
[16] M. Caramia and A. Sgalambro, An exact approach for the maximum concurrent k-splittable flow problem. *Optim. Lett.* **2** (2008) 251–265.
[17] M. Caramia and A. Sgalambro, A fast heuristic algorithm for the maximum concurrent k-splittable flow problem. *Optim. Lett.* **4** (2010) 37–55.
[18] Y. Azar and O. Regev, Combinatorial algorithms for the unsplittable flow problem. *Algorithmica* **44** (2006) 49–66.
[19] J. Weiner, A.T. Ernst, X. Li, Y. Sun and K. Deb, Solving the maximum edge disjoint path problem using a modified Lagrangian particle swarm optimisation hybrid. *Eur. J. Oper. Res.* **293** (2021) 847–862.
[20] G. Baier, E. Köhler and M. Skutella, The k-splittable flow problem. *Algorithmica* **42** (2005) 231–248.
[21] J. Truffot and C. Duhamel, A branch and price algorithm for the k-splittable maximum flow problem. *Discrete Optim.* **5** (2008) 629–646.
[22] J.W. Suurballe and R.E. Tarjan, A quick method for finding shortest pairs of disjoint paths. *Networks* **14** (1984) 325–336.
[23] S.G. Kolliopoulos, *Edge-disjoint paths and unsplittable flow*. Tech. Rep., National and Kapodistrian University of Athens (2007).
[24] S.G. Kolliopoulos, *Disjoint paths and unsplittable flow* (version 2.7), Tech. Rep., National and Kapodistrian University of Athens (2016).
[25] Y. Deng, L. Guo, K. Liao and Y. Chen, On finding maximum disjoint paths with different colors: computational complexity and practical LP-based algorithms. *Theor. Comput. Sci.* **886** (2021) 157–168.
[26] L.R. Ford and D.R. Fulkerson, Maximal flow through a network. *Can. J. Math.* **8** (1956) 399–404.
[27] H.D. Sherali and J.C. Smith, Improving discrete model representations via symmetry considerations. *Manage. Sci.* **47** (2001) 1396–1407.
[28] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993).
[29] T. Böhme, F. Göring and J. Harant, Menger's theorem. *J. Graph Theory* **37** (2001) 35–36.