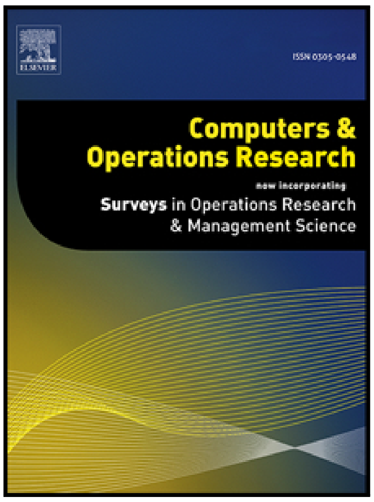

A Hybrid VNS/Tabu Search Algorithm for Solving the Vehicle Routing Problem with Drones and En Route Operations

Daniel Schermer, Mahdi Moeini, Oliver Wendt

PII: S0305-0548(19)30104-2
DOI: https://doi.org/10.1016/j.cor.2019.04.021
Reference: CAOR 4692

To appear in: *Computers and Operations Research*

Received date: 27 August 2018
Revised date: 27 March 2019
Accepted date: 27 April 2019

Highlights

- We study vehicle routing problem with drones and en route operations,
- formulate the problem as a MILP and introduce valid inequalities,
- show potentials and computational challenges of the problem,
- introduce an effective algorithm based on Variable Neighborhood Search,
- and report numerical results of our comprehensive computational experiments.

A Hybrid VNS/Tabu Search Algorithm for Solving the Vehicle Routing Problem with Drones and En Route Operations

Daniel Schermer[a], Mahdi Moeini[a,*], Oliver Wendt[a]

[a]*Chair of Business Information Systems and Operations Research (BISOR), Technische Universität Kaiserslautern, 67663 Kaiserslautern, Germany. {daniel.schermer,mahdi.moeini,wendt}@wiwi.uni-kl.de*

Abstract

With the goal of integrating drones in last-mile delivery, the *Vehicle Routing Problem with Drones* (VRPD) uses a fleet of vehicles, each of them equipped with a set of drones, for serving a set of customers with minimal makespan. In this paper, we propose an extension of the VRPD that we call the *Vehicle Routing Problem with Drones and En Route Operations* (VRPDERO). Here, in contrast to the VRPD, drones may not only be launched and retrieved at vertices but also on some discrete points that are located on each arc. We formulate the problem as a *Mixed Integer Linear Program* (MILP) and introduce some valid inequalities that enhance the performance of the MILP solvers. Furthermore, due to limited performance of the solvers in addressing large-scale instances, we propose an algorithm based on the concepts of *Variable Neighborhood Search* (VNS) and *Tabu Search* (TS). In order to evaluate the performance of the introduced algorithm as well as the solver in solving the VRPDERO instances, we carried out extensive computational experiments. According to the numerical results, the proposed valid inequalities and the heuristic have a significant contribution in solving the VRPDERO effectively. In addition, the consideration of en route operations can increase the utilization of drones and lead to an improved makespan.

Keywords: Vehicle Routing Problem, Drones, Last-Mile Delivery, Heuristics, Metaheuristics, Variable Neighborhood Search, Valid Inequalities, Tabu Search

1. Introduction

Minimum cost and fast pickup and delivery of products are important problems in logistics. In this context, for example, several variants of the classical *Vehicle Routing*

*Corresponding author

Problem (VRP) have been studied and solved using a wide range of heuristic and exact algorithms (see, e.g., (Toth and Vigo, 2002), and references therein).

Recently, there has been an interest in the integration of drones in civil applications and, particularly, in last-mile delivery. Drones have been successfully applied in many public and private sectors and are slowly becoming a proven technology (see, e.g., (Otto et al., 2018)). As a result, in an effort to lower costs and reduce delivery times, several optimization problems have been proposed in the academic literature that integrate drones into last-mile delivery (see, e.g. (Murray and Chu, 2015; Wang et al., 2016; Agatz et al., 2018)). In these problems, which might be considered as variants of the VRP or *Traveling Salesman Problem* (TSP), the shared idea of a tandem that comprises of traditional vehicles and drones is commonly accepted. Hence, a drone might be launched from a vehicle, makes an autonomous flight, delivers a parcel to an identified customer, and is then retrieved by the vehicle at a different location. Generally, we might assume that a drone can move faster between two locations than a vehicle as drones are not exposed to the restriction of following the road network structure or congestion. Additionally, drones are lightweight, which causes them to consume less energy for the movement between two points. However, their carrying capacity is more restricted than that of a vehicle. Moreover, as drones rely on relatively small batteries for powering their flight, the range of operation is more tightly bounded than that of a vehicle powered by a fossil fuel or a much larger battery (in case of electric cars). Consequently, due to the complementary features of vehicles and drones, the latter might have a supplementary role in last-mile delivery. Classically, it is commonly assumed that the launch and retrieval of drones is restricted to the locations of customers and the depot (Murray and Chu, 2015; Agatz et al., 2018). However, several researchers have suggested the relaxation of this assumption and promoted the idea of launching and retrieving drones at arbitrary locations (Poikonen et al., 2017; Marinelli et al., 2018). In particular, Marinelli et al. (2018) propose *en route* operations, i.e., the possibility to construct *arc-based* truck-drone operations. Indeed, as one of many companies that are interested in the integration of drones in last-mile delivery, Amazon Technologies Inc. has patented the possibility of incorporating drone docking stations into existing structures such as cell towers, light and power poles, and buildings that might be used as intermediate docks for supporting en route operations (Gentry et al., 2016).

In this article, we propose a new problem and we call it the *Vehicle Routing Problem with Drones and En Route Operations* (VRPDERO), which is an extension of the *Vehicle Routing Problem with Drones* (VRPD) proposed by Wang et al. (2016). In fact, compared to the VRPD, the VRPDERO assumes that drones might also be launched and retrieved at some discrete locations on each arc. The contributions of our work are

as follows:
(1) We formulate the VRPDERO as a *Mixed Integer Linear Program* (MILP). Consequently, the problem might be solved by any standard MILP solver, e.g., Gurobi Optimizer or IBM CPLEX. Furthermore, by considering the definition and mathematical structure of the VRPDERO, we introduce some sets of valid inequalities that can be beneficial in enhancing the performance of the MILP solvers. Through our numerical results, we show that our proposed valid inequalities have a significant impact on the solver's performance in finding optimal solutions to small instances.
(2) In the VRPDERO, the MILP formulation can only be used on small-scale problems to obtain optima in a reasonable amount of time. Therefore, in order to address larger instances of the VRPDERO, we propose a heuristic that combines components from *Variable Neighborhood Search* (VNS) and *Tabu Search* (TS). Furthermore, our algorithm incorporates a problem-specific drone insertion operator as local search method that is designed according to the structure and characteristics of the VRPDERO. This operator is embedded within a scalable *divide and conquer* approach. We carried out extensive computational experiments on VRPDERO instances and we show that our algorithm provides high-quality solutions in reasonable computation time.
(3) Across our numerical study, we demonstrate that, with respect to the VRPD, the consideration of en route operations can be beneficial for improving the utilization of drones, thus improving the makespan. Albeit, the introduction of en route operations significantly increases the computational complexity of the problem.

The remainder of this paper is organized as follows. We begin in Section 2 by providing a brief overview on the existing academic literature related to the application of drones in last-mile delivery. Then, Section 3 presents the notation, the mathematical model, and the proposed valid inequalities for the VRPDERO. In Section 4, we introduce a heuristic approach that is based on the concepts of VNS and TS. This section also contains a detailed description of the drone insertion procedure and the associated divide and conquer approach. Detailed results on the numerical studies, which were performed using both the MILP formulation and the heuristic approach, are presented in Section 5. Finally, Section 6 is devoted to the concluding remarks and future research implications.

2. Related Research Works

Otto et al. (2018) provide a comprehensive overview on the use of drones in civil applications. However, for the sake of completeness, we present a brief overview of some of the research works that are most relevant to this paper.

Murray and Chu (2015) introduced two new $\mathcal{NP}$-hard problems, called the *Flying Sidekick Traveling Salesman Problem* (FSTSP) and the *Parallel Drone Scheduling TSP*

(PDSTSP), that are related to delivery applications, where a vehicle (or truck) is working in tandem with a drone. In the FSTSP, a set of customers is given, each of whom must be visited exactly once by either a drone or a vehicle. The vehicle-drone-tandem starts from a common *distribution center* (DC) and must return to the same DC at the end of the tour. At any customer location, the drone may be launched from the vehicle, makes an autonomous delivery to another customer location and is then retrieved by the vehicle at a different location. In the PDSTSP, it is assumed that the DC is in close proximity to most of the customers. In this case, it can be beneficial to let the drone make its deliveries independently from the vehicle, while the vehicle serves customers that are located far away from the DC. Whereas the vehicle continues on a predefined tour, the drone will continuously return to the DC to pick up a new parcel after each delivery. Both problems, i.e., the FSTSP and PDSTSP, have the objective of minimizing the makespan. However, the latter has significantly reduced *synchronization* requirements (Drexl, 2012). Murray and Chu (2015) proposed two simple heuristic methods for solving the FSTSP and PDSTSP, respectively, as well as a MILP to solve either problem. They show that a significant makespan reduction is possible through the introduction of drones and highlight that the FSTSP is more difficult to solve.

The *Traveling Salesman Problem with Drone* (TSP-D), introduced by Agatz et al. (2018), is another problem that shares most features with the FSTSP. Here, a key difference to the FSTSP is the relaxation that the TSP-D permits a drone to be launched and retrieved at the same vertex. Agatz et al. (2018) provide theoretical insights and propose a MILP formulation as well as several new heuristics for solving the TSP-D. They confirm that significant savings are possible and conclude by suggesting the extension of their model to consider multiple vehicles (or drones) as well as adding different possibilities of recharging the drones.

Wang et al. (2016) introduced the VRPD which might be considered as a generalization of the FSTSP and TSP-D. In the VRPD, we assume that a depot, a set of customers, and a fleet of vehicles are given such that each vehicle is equipped with a predefined number of drones. The objective is to minimize the makespan, i.e., the time required to serve all customers such that the given fleet (i.e., vehicles and drones) has returned to the depot at the end of the mission. Wang et al. provided several theoretical insights. In particular, they introduced some upper bounds on the potential reductions in makespan by employing drones (compared to a case where no drones are given). The upper bounds are obtained by investigating the structure of optimal solutions and depend on the relative velocity of the drones compared to the vehicles and the number of drones carried by each vehicle. Poikonen et al. (2017) refined the work of Wang et al. (2016) by further considering the impact on the previously introduced bounds when integrating limited battery life, different distance metrics, and operational expenditures of deploying drones and vehicles in the objective function. Furthermore, Poikonen et al. (2017) proposed extending the VRPD model similar to the *close-enough traveling salesman problem* (CETSP). More precisely, the authors suggest the possibility of launching and recovering drones from arbitrary locations instead of being restricted to the locations of customers and the depot. They conclude their work by proposing that some heuristics and benchmark instances should be developed as a next step in researching the VRPD from a computational point of view.

In (Schermer et al., 2018a), based on the work by Di Puglia Pugliese and Guerriero (2017), we introduced a MILP formulation for the VRPD along with several valid inequalities. In particular, we provided a heuristic approach based on *Variable Neighborhood Search* (VNS) and two heuristic drone insertion operators for solving the VRPD. Through an extensive numerical study, we showed the effectiveness of our proposed valid inequalities and heuristic for solving small- and large-scale instances of the VRPD. Finally, we could demonstrate that a vehicle-drone-tandem allows for significantly reduced makespans compared to classic vehicle delivery.

In (Marinelli et al., 2018), the authors allowed drones to be launched and retrieved from arbitrary locations that are located on the path of the vehicle. More precisely, they propose a novel variant of the TSP-D in which the drones can be launched and retrieved on arcs by the vehicle, which they call an *en route operation*. They state that this procedure can be beneficial in increasing the utilization of drones and in overcoming their limited range of operation. In order to solve the extended TSP-D, Marinelli et al. (2018) propose a heuristic which consists of three distinct steps: First, they find a routing for the vehicle using the well-known Lin-Kernighan heuristic (Lin and Kernighan, 1973). Second, given the routing of the vehicle, they build a list of all feasible drone operations that yield a reduced cost. Third, the initial list is improved through the consideration of en route operation and the moves are inserted according to their reduced cost. Through their numerical study, they show that en route operations can yield improvements of $0 \ldots 10\%$ over the non-en route case with regards to the makespan.

3. The Vehicle Routing Problem with Drones and En Route Operations

In this paper, we introduce and study the VRPDERO, which might be considered an extension to the VRPD. We recall that in the VRPD, the drone sorties, i.e., the operation of launching from a vehicle, serving a customer, and returning to the vehicle, might only be done at the vertices (i.e., customer locations and the depot). However, in the VRPDERO, the operations might also start and end at some discrete points which are located on the arcs that are traversed by the vehicles. In other words, the VRPDERO admits *en route operations*. In fact, en route operations might be useful in two cases as highlighted in Figure 1. On the one hand, if a drone is much faster than the vehicle, then it might be able to perform multiple operations during a single transition of the vehicle. On the other hand, if a drone has a low endurance, then it might be possible to perform operations, that would otherwise not be feasible. Hence, VRPDERO can be considered as a natural extension of the VRPD.

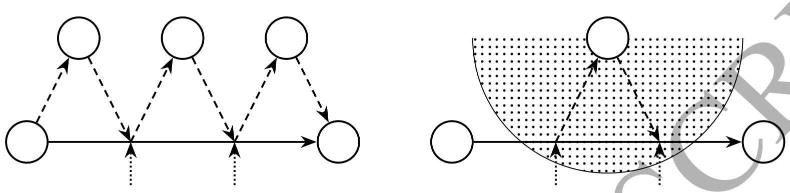

Figure 1: The paths of the vehicle and drone are shown using solid and dashed lines, respectively. The dotted arrows on the solid arcs indicate locations at which en route operations are possible. On the left-hand side, one fast-moving drone is able to perform multiple operations during a single transition of the vehicle. On the right-hand side, we assume that the customer might only be served by drone, if the drone is launched and retrieved within the feasible region (indicated by the dotted pattern). Here, through en route operations, a drone is able to perform an operation that would otherwise not be possible.

3.1. Notation and Mathematical Model

Suppose that a set of customer locations, each with an equal demand, and a fleet of homogeneous vehicles, each carrying a specified number of identical drones, are given. In the VRPDERO, we look for minimizing the time required to serve all customers using the vehicles and the drones such that, by the end of the mission, all vehicles and drones must be at the depot. Furthermore, we make the following assumptions (Murray and Chu, 2015; Poikonen et al., 2017; Schermer et al., 2018a,b; Wang et al., 2016):

- A drone can serve exactly one customer per operation and has a limited endurance of $\mathcal{E}$ distance units. After returning to the vehicle, the battery of the drone is recharged instantaneously.
- While the vehicles are restricted to the road network, the drones might be able to use a different trajectory for traveling between locations. In addition, without loss of generality, the average velocity of each vehicle is assumed to be equal to 1 and the relative velocity of each drone is assumed to be $\alpha \in \mathbb{R}^+$ times the velocity of the vehicle. Due to the absence of congestion or necessary stops (e.g., traffic lights and stop signs) in the route of the drone, we assume that $\alpha \geq 1$.
- The service time required to serve a customer by vehicle or drone and the time required to launch and retrieve a drone are assumed to be negligible.

- Drones may only be launched and retrieved either at vertices (i.e., at the depot and any other customer location) or at predefined discrete points which are located on arcs that are traversed by vehicles. Furthermore, a drone may not be picked up at the same location from which it was launched. Finally, a drone must always return to the same vehicle from which it was launched.

In order to present the mathematical model of the VRPDERO, we use the following notation which is similar to that of Di Puglia Pugliese and Guerriero (2017) and Schermer et al. (2018a).

Assume that a complete and symmetric graph $\mathcal{G} = (V, A)$ is given, where V is the set of vertices and A is the set of arcs. The set V contains n vertices associated with the customers, named $V_N = \{1, \dots, n\}$ and (in order to simplify the notation and the mathematical formulation) two extra vertices 0 and $n+1$ that both represent the same depot location at the start and end of each tour, respectively. Hence, $V = \{0\} \cup V_N \cup \{n+1\}$, where $0 \equiv n+1$. Each vertex $V_i \in V$ is defined by a tuple (x_i, y_i) in the Euclidean space. We refer to $V_L = V \setminus \{n+1\}$ as the subset of vertices in V from which drones may be launched. Furthermore, $V_R = V \setminus \{0\}$ denotes the subset of vertices where the drones may be retrieved.

In order to define the discrete points on the arcs, let $S = \{0, 1, \dots, s_n\}$ be a set of predefined integer numbers, where $s_n \in \mathbb{Z}^+$, $|S| \geq 2$. Given two vertices $V_i \in V_L, V_j \in V_R$, the location of a discrete point $\vec{V_{ij}^s}$ that is located at step $s \in S$ on arc $\vec{A}_{ij} \in A$ (connecting $\vec{V_i}$ to $\vec{V_j}$) can be calculated through formula (1):

$$\vec{V_{ij}^s} = \vec{V_i} + \frac{s}{|S|-1}(\vec{V_j} - \vec{V_i}) = \vec{V_i} + \frac{s}{|S|-1}\vec{A_{ij}} \tag{1}$$

We assume that for $s = 0$ and $s = s_n$, $\vec{V_{ij}^s}$ coincides with $\vec{V_i}$ and $\vec{V_j}$, respectively. Further, we use the sets $S_L = S \setminus \{s_n\}$ and $S_R = S \setminus \{0\}$ for defining the discrete points on each arc from which a drone might be launched and retrieved.

The parameters d_{ij} and $\overline{d}_{ij}$ define the required distance to travel from vertex i to vertex j by vehicle and drone, respectively. As $\mathcal{G}$ is assumed to be symmetric, $d_{ij} = d_{ji}$ and $\overline{d}_{ij} = \overline{d}_{ji} : \forall i, j \in V$. Furthermore, by definition, $d_{ii} = \overline{d}_{ii} = 0 : \forall i \in V$. Additionally, we introduce $\overline{d}_{ij}^{w,s}$ as the distance between V_{ij}^s and V_w. As the drone may not be limited to the road network, we can assume that $\overline{d}_{ij} \leq d_{ij} : \forall i, j \in V$.

We use the parameters v and $\overline{v}$, where $v = 1$ and $\overline{v} = \alpha \cdot v$, to indicate the velocity of the vehicle and drone, respectively. We assume that both, the vehicle and the drone, move at a constant velocity. Hence, the time required to traverse edge (i, j) is defined as $t_{ij} = d_{ij}$ and $\overline{t}_{ij} = \overline{d}_{ij}/\alpha$ for the vehicles and drones, respectively. As we assumed that $\alpha \geq 1$ and $\overline{d}_{ij} \leq d_{ij} : \forall i, j \in V$, we can conclude that $\overline{t}_{ij} \leq t_{ij} : \forall i, j \in V$. Analogously, the time required to reach (or return from) a customer $w \in V_N$ from (or to) a discrete point V_{ij}^s, where $V_i \in V_L, V_j \in V_R$ and $s \in S$, by a drone is given by $\bar{t}_{ij}^{w,s} = \bar{d}_{ij}^{w,s}/\alpha$.

As in the VRPD (Wang et al., 2016; Poikonen et al., 2017), we suppose that we have a given fleet that consists of a limited set $K = \{1, \ldots, K_N\}$ $(K_N \in \mathbb{Z}^+, |K| \geq 1)$ of vehicles such that each vehicle $k \in K$ holds an equal number of $D \in \mathbb{Z}^{\geq 0}$ identical drones. Each drone may travel a maximum distance of $\mathcal{E}$ units per operation, where a *drone-operation* is characterized by a tuple (i, s_l, w, s_r, j) or, alternatively, two tuples: (i, w, s_l) for deliveries and (w, j, s_r) for retrievals. More precisely, the drone is launched from a vehicle at $s_l \in S_L$ en route from $i \in V_L$ to serve a customer $w \in V_N$. Afterwards, the drone is retrieved by the vehicle, from which the drone was launched, at $s_r \in S_R$ en route to $j \in V_R$.

We use the following decision variables for modeling the VRPDERO (see also Figure 2):

$x_{ij}^k \in \{0,1\}$ $_{\forall k \in K, i \in V_L, j \in V_R}$	:	Variables that state whether arc (i,j) belongs to the route of vehicle k.
$y_{iw}^{ks_l} \in \{0,1\}$ $_{\forall k \in K, i \in V_L, w \in V_N, s_l \in S_L}$	:	Variables that indicate if a drone that belongs to vehicle k is launched at s_l en route from vertex i to serve w.
$\tilde{y}_{wj}^{ks_r} \in \{0,1\}$ $_{\forall k \in K, w \in V_N, j \in V_R s_r \in S_R}$	:	Variables that specify if a drone that belongs to vehicle k is retrieved at s_r en route to vertex j after a delivery to w.
$s_{iwj}^k \in \{0,1\}$ $_{\forall k \in K, i \in V_L, w \in V_N, j \in V_R}$	:	Variables that state if a drone that belongs to vehicle k performs a delivery to w that occurs on arc (i,j) at any feasible combination of steps $s_l \in S_L$ and $s_r \in S_R$.
$u_i^k \in \mathbb{Z}^{\geq 0}$ $_{\forall k \in K, i \in V}$	:	Integer variables that indicate the position of vertex i in the route of vehicle k (if customer i is served by vehicle k).
$p_{ij}^k \in \{0,1\}$ $_{\forall k \in K, \forall i \in V_L, j \in V_R}$	:	Variables that specify whether i is served before (but not necessarily adjacent to) j by vehicle k.
$z_{aiwje}^k \in \{0,1\}$ $_{\forall k \in K, a, i, w, j, e \in V}$	:	Variables that state whether any drone delivery that targets w en route from (or at) a and ends en route to (or at) e is performed circumjacent to (i,j), i.e., launched before the vehicle k has reached i and retrieved after the vehicle has reached j.
$a_i^k \in \mathbb{R}^{\geq 0}$ $_{\forall k \in K, i \in V}$	:	Continuous variables that indicate the earliest time at which vertex i is reached by the vehicle and any drone recovered directly at i. If i is served by a drone, this variable simply marks the earliest time at which i is served.
$a_j^{ks} \in \mathbb{R}^{\geq 0}$ $_{\forall k \in K, j \in V_R, s \in S_R}$	:	Continuous variables that indicate the earliest time at which step s en route to j is reached by the vehicle k and any drone that must be recovered here.

$\tau \in \mathbb{R}^{\geq 0}$: Continuous variable that indicates the makespan.

The MILP formulation of the VRPDERO is given in (2) - (25).

$$\min \quad \tau \tag{2}$$

$$\text{s.t.:} \quad a_{n+1}^k \leq \tau \ : \ \forall k \in K, \tag{3}$$

$$\sum_{j \in V_N} x_{0j}^k \leq 1 \ : \ \forall k \in K, \tag{4}$$

$$\sum_{j \in V_N} x_{0j}^k - \sum_{i \in V_N} x_{i,n+1}^k = 0 \ : \ \forall k \in K, \tag{5}$$

$$\sum_{i \in V_L} x_{ih}^k - \sum_{j \in V_R} x_{hj}^k = 0 \ : \ \forall k \in K, h \in V_N, \tag{6}$$

$$\sum_{k \in K} \sum_{i \in V_L} x_{iw}^k + \sum_{k \in K} \sum_{i \in V_L} \sum_{s_l \in S_L} y_{iw}^{ks_l} = 1 \ : \ \forall w \in V_N, \tag{7}$$

$$\sum_{w \in V_N} \sum_{s_l \in S_L} y_{iw}^{ks_l} \leq M \sum_{j \in V_R} x_{ij}^k \ : \ \forall k \in K, i \in V_L, \tag{8}$$

$$\sum_{w \in V_N} \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_l} \leq M \sum_{i \in V_L} x_{ij}^k \ : \ \forall k \in K, j \in V_R, \tag{9}$$

$$\sum_{i \in V_L} \sum_{s_l \in S_L} y_{iw}^{ks_l} - \sum_{j \in V_R} \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_r} = 0 \ : \ \forall k \in K, w \in V_N, \tag{10}$$

$$u_i^k - u_j^k + 1 \leq (n+1)(1 - x_{ij}^k) \ : \ \forall k \in K, i \in V_L, j \in V_R, \tag{11}$$

$$u_i^k - u_w^k + 1 \leq (n+1)(1 - y_{iw}^{ks_l}) \ : \ \forall k \in K, i \in V_L, w \in V_N, s_l \in S_L, \tag{12}$$

$$u_w^k - u_j^k + 1 \leq (n+1)(1 - \tilde{y}_{wj}^{ks_r}) \ : \ \forall k \in K, j \in V_R, w \in V_N, s_r \in S_R, \tag{13}$$

$$u_i^k - u_j^k + 1 \leq (n+1)(1 - p_{ij}^k) \ : \ \forall k \in K, i \in V_L, j \in V_R, \tag{14}$$

$$u_i^k - u_j^k \geq 1 - (n+1)p_{ij}^k \ : \ \forall k \in K, i \in V_L, j \in V_R, \tag{15}$$

$$\begin{aligned} &4z_{aiwje}^k - (p_{ai}^k + p_{je}^k) \\ &\leq \sum_{s_l \in S_L} y_{aw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{we}^{ks_r} \ : \ \forall k \in K, a \in V_L, i, w, j \in V_N, e \in V_R, \end{aligned} \tag{16}$$

$$\begin{aligned} &3 + z_{aiwje}^k - (p_{ai}^k + p_{je}^k) \\ &\geq \sum_{s_l \in S_L} y_{aw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{we}^{ks_r} \ : \ \forall k \in K, a \in V_L, i, w, j \in V_N, e \in V_R, \end{aligned} \tag{17}$$

$$a_j^{ks_n} = a_j^k \ : \ \forall k \in K, j \in V_R, \tag{18}$$

$$M(x_{ij}^k - 1) + a_i^k + \frac{t_{ij}}{|S| - 1} \leq a_j^{k1} \ : \ \forall k \in K, i \in V_L, j \in V_R, \tag{19}$$

$$M(x_{ij}^k - 1) + a_j^{ks} + \frac{t_{ij}}{|S| - 1} \leq a_j^{k,s+1} \ : \ \forall k \in K, i \in V_L, j \in V_R, s \in S_L \cap S_R, \tag{20}$$

$$M(x_{ij} + y_{iw}^{k0} - 2) + a_i^k + \bar{t}_{iw} \leq a_w^k \ : \ \forall k \in K, i \in V_L, w \in V_N, j \in V_R, s_l \in S_L, \tag{21}$$

$$M(x_{ij} + y_{iw}^{ks} - 2) + a_j^{ks} + \bar{t}_{ij}^{ws} \leq a_w^k \ : \ \forall k \in K, i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R, \tag{22}$$

$$M(x_{ij} + \tilde{y}_{wj}^{ks_r} - 2) + a_w^k + \bar{t}_{ij}^{ws_r} \leq a_j^{ks_r} \ : \ \forall k \in K, i \in V_L, w \in V_N, j \in V_R, s_r \in S_R, \tag{23}$$

$$\begin{aligned} &\overline{d}_{ij}^{ws_l}(x_{ij} + y_{iw}^{ks_l} - 1) \\ &+\overline{d}_{gh}^{ws_r}(x_{gh} + \tilde{y}_{gh}^{ks_r} - 1) \leq \mathcal{E} \ : \ \forall k \in K, i, g \in V_L, w \in V_N, \\ &\qquad j, h \in V_R, s_l \in S_L, s_r \in S_R, \end{aligned} \tag{24}$$

$$2 + s_{iwj}^k \geq x_{ij}^k + \sum_{s_l \in S_L} y_{iw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_r} \ : \ \forall k \in K, i \in V_L, w \in V_N, j \in V_R, \tag{25}$$

$$3s_{iwj}^k \leq x_{ij}^k + \sum_{s_l \in S_L} y_{iw}^{ks_l} + \sum_{s_r \in S_R} \tilde{y}_{wj}^{ks_r} \ : \ \forall k \in K, i \in V_L, w \in V_N, j \in V_R, \tag{26}$$

$$\begin{aligned} &\sum_{a \in V_L} \sum_{w \in V_N} \sum_{e \in V_R} z_{aiwje}^k \\ &+ \sum_{w \in V_N} \sum_{\substack{s_l \in S_L, \\ s_l \leq s}} y_{iw}^{ks_l} + \sum_{w \in V_N} \sum_{\substack{s_r \in S_R, \\ s_r > s}} \tilde{y}_{wj}^{ks_r} \\ &- \sum_{w \in V_N} s_{iwj}^k + M(x_{ij}^k - 1) \leq D \ : \ \forall k \in K, i \in V_L, j \in V_R, s \in S_L. \end{aligned} \tag{27}$$

In the VRPDERO, the objective is to minimize the makespan τ. Based on the arrival time at the depot, i.e., a_{n+1}^k, constraints (3) provide lower bounds on the makespan for each vehicle k (and its assigned drones). The set of constraints (4) - (6) form the preservation of flow for the vehicles. More precisely, constraints (4) state that each vehicle $k \in K$ may depart from the depot at most once. For any vehicle that has departed, constraints (5) guarantee that the vehicle must return exactly once. Constraints (6) ensure the preservation of flow for each vehicle. According to constraints (7), each customer must be served exactly once by either a vehicle or a drone. Synchronization between the route of a vehicle and drone operations is required through constraints (8) - (9). More precisely, these constraints guarantee that drone operations can only occur at discrete points associated with vertices that are visited by the respective vehicle. For each customer that is served by a drone, constraints (10) state that the flow of drones must be preserved. Constraints (11) - (13) are Miller-Tucker-Zemlin type constraints (Miller et al., 1960) that link the variables u_i^k to the decision variables x_{ij}^k, $y_{iw}^{ks_l}$, and $\tilde{y}_{iw}^{ks_l}$. In a similar manner, constraints (14) - (15) guarantee the right precedence relations through variables p_{ij}^k with regards to u_i^k and u_j^k. Constraints (16) - (17) determine if circumjacent deliveries occur (see also Figure 1). For a circumjacent delivery to occur, a drone must be launched before a vehicle k has reached i and retrieved after the same vehicle has reached j. Constraints (18) - (23) determine valid bounds on the temporal variables a_i^k and a_j^{ks}. Here, constraints (18) state that the arrival time en route to j at $s_r = s_n$ equals the time a_j^k at the vertex itself. Furthermore, for any arc that is used by the vehicle, constraints (19) and (20) provide lower bounds at each discrete step. In a similar manner, constraints (21) - (22) determine arrival times for any customer that is served by a drone. Finally, constraints (23) guarantee valid bounds whenever a drone is retrieved by a vehicle. Constraints (24) impose a maximum distance (endurance) constraint on every possible drone operation. For drone operations that involve a single arc (i, j), constraints (25) - (26) force the variables s_{iwj}^k to take value 1. Finally, constraints (27) guarantee that during the transition of a vehicle on arc (i, j), at each discrete step, a maximum of D drones may be in operation simultaneously. To this end, we account for the following:

- circumjacent deliveries,
- the launch of any drone that is performed on the same arc up to step s,
- and any outstanding future recovery of a drone on the same arc.

To make this accounting correct, we must prevent counting the same operation twice (once as launch up to s and once as a future recovery). To this end, we subtract any operation s_{iwj}^k that starts and ends on this arc (i, j) (see also Figure 2 and Table 1).

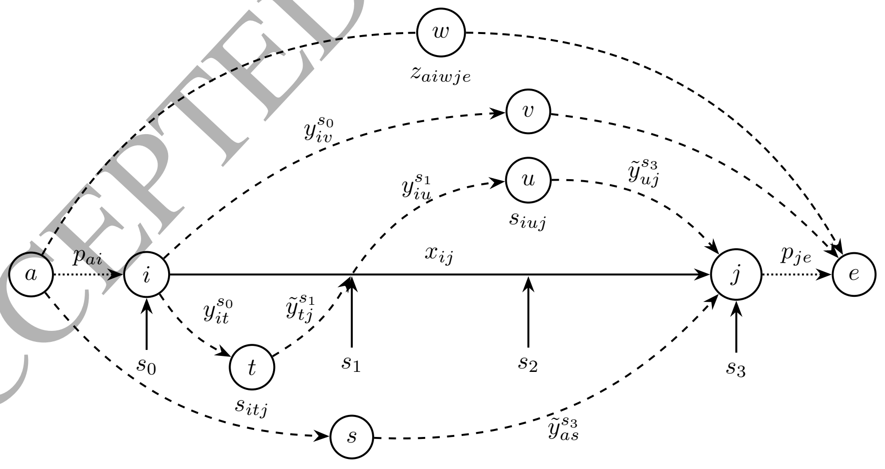

Figure 2: A visualization of some of the decision variables for a given vehicle k. The solid and dotted lines refer to the path of the vehicle. The dashed lines indicate deliveries made by drones, which are assigned to the vehicle.

s	$\sum z$	$\sum y$	$\sum \tilde{y}$	$-\sum s_{iwj}$	LHS according to (27)
s_0	1	2	3	-2	4
s_1	1	3	2	-2	4
s_2	1	3	2	-2	4

Table 1: This table illustrates how the number of operations is accounted for based on constraint (27) and Figure 2. The first column shows the step s on arc (i, j). The remaining columns show the number of circumjacent deliveries surrounding (i, j) ($\sum z$), launches up to s_i on (i, j) ($\sum y$), future recoveries after s_i on (i, j) ($\sum \tilde{y}$) and the negative number of operations that involve the single arc (i, j) ($\sum s_{iwj}$). The last column shows the left hand side (LHS) according to (27) , i.e., the summation over columns 2 to 5 in the respective row of the table. In this case, the vehicle needs to be equipped with $D \geq 4$ drones to perform the operations as illustrated in Figure 2.

3.2. Valid Inequalities

In solving a challenging combinatorial optimization problem, a well-known approach for enhancing the performance of a MILP solver consists in using suitable valid inequalities. In this context, Schermer et al. (2018a) provides several sets of effective valid inequalities for solving the VRPD. Due to their success, we adapt them to the VRPDERO and improve them for providing sharper lower bounds that strengthen the relaxation of the formulation and restrict the solution space of the problem while excluding none of the optimal solutions.

Proposition 1 *In the MILP (2) - (27), τ is bounded by the time spent traveling by each vehicle.*

Proof: Let τ^k be the value which indicates the total time that the vehicle $k \in K$ spends traveling on arcs. It can be computed as follows:

$$\tau^k = \sum_{i \in V_L} \sum_{j \in V_R} x_{ij}^k t_{ij} \ : \ \forall k \in K. \tag{28}$$

In any VRPDERO solution, a_{n+1}^k, i.e., the earliest possible arrival time of the vehicle k and all drones that are assigned to it at depot, is bounded by the arrival of the vehicle itself. For each vehicle k, this value is a lower bound on τ according to constraints (3). Indeed, since it might happen that the vehicle k waits at some vertices (or at discrete points on arcs) for the arrival of a drone, we can use τ^k as a lower bound on a_{n+1}^k. Consequently, we can obtain the following valid inequalities:

$$\sum_{i \in V_L} \sum_{j \in V_R} x_{ij}^k t_{ij} \leq \tau \ : \ \forall k \in K, \tag{29}$$

which completes the proof of Proposition 1. ■

In the MILP (2) - (27), drone operations are characterized implicitly. More precisely, the variables $y_{iw}^{ks_l}$ (respectively, $\tilde{y}_{wj}^{ks_l}$) state that a drone operation starts at s_l en route from i to serve a customer w (respectively, ends at s_r en route to j after a delivery to w). However, only through the consideration of the binary variables x_{ij} we do know which vertices are visited after i and before j. This information is crucial for the determination of the position of discrete steps and, therefore, travel times of drones. Hence, for the general case of $s_n > 2$, we propose the inclusion of auxiliary variables and constraints to the MILP (2) - (27) as it is suggested by the following Lemma.

Lemma 1 *Assume that y^{ks}_{iwj}, where $k \in K, i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R$, are binary variables stating whether a drone that belongs to the vehicle k travels from s on (i,j) to w or in reverse. Then, the following inequalities (30) - (31) are valid for the MILP (2) - (27):*

$$2y^{ks}_{iwj} \leq x^k_{ij} + y^{ks}_{iw} + \tilde{y}^{ks}_{wj} \ : \ \forall i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R, \tag{30}$$

$$1 + y^{ks}_{iwj} \geq x^k_{ij} + y^{ks}_{iw} + \tilde{y}^{ks}_{wj} \ : \ \forall i \in V_L, w \in V_N, j \in V_R, s \in S_L \cap S_R. \tag{31}$$

Proof: These inequalities guarantee that y^{ks}_{iwj} is forced to value 1 if two conditions apply. First, arc (i,j) must be a part of the vehicle k's route. Second, either a delivery to w must be initiated by a drone or a drone must retrieved after a delivery to w en route at s on arc (i,j).

These constraints are valid with respect to MILP (2) - (27), because a drone launched to customer w can not be retrieved at the same location from which it was launched (see Section 3.1 and MILP (2) - (27)). Therefore, the sum over y^{ks}_{iw} and $\tilde{y}^{ks}_{wj}$ can be at most one. Furthermore, an optimal solution vector x^* to MILP (2) - (27) is guaranteed to be feasible subset of an optimal solution vector to MILP (2) - (27), (30) - (31). Thus, the value of the optimal solution does not change. This completes the proof of Lemma 1. ■

Proposition 2 *In the MILP (2) - (27), τ is bounded by the average time spent traveling by drones.*

Proof: We distinguish two cases: whether $|S| = 2$ ($s_n = 1$), i.e., there is no en route operation, and $|S| \geq 3$ ($s_n \geq 2$), i.e., in presence of en route operations.

- **(i)** For $|S| = 2$, we have the following case:

Since $|S| = 2$, hence, $S = \{0, 1\}$ and there are no en route operations. We define τ^k_i as the total time that the i-th drone, which is assigned to vehicle $k \in K$, spends traveling on arcs of the underlying graph. Then, we can compute the total time spent traveling by the D drones as follows:

$$\sum_{i=1}^{D} \tau^k_i = (\sum_{i \in V_L} \sum_{w \in V_N} \bar{t}_{iw} y^{ks_0}_{iw} + \sum_{w \in V_N} \sum_{j \in V_R} \bar{t}_{wj} \tilde{y}^{ks_n}_{wj}) \ : \ \forall k \in K. \tag{32}$$

Let $\tau^k_{max} = max(\tau^k_1, \dots, \tau^k_D)$, i.e., τ^k_{max} is the maximum time that a single drone, associated to a vehicle $k \in K$, spends traveling. Since a drone might also have to wait at a vertex or an arc in order to be retrieved by the vehicle, the makespan τ, at which all drones and the vehicle have returned to the depot, cannot be smaller than τ^k_{max}, that is:

$$\tau^k_{max} \leq \tau \ : \ \forall k \in K. \tag{33}$$

As we do not explicitly account for the travel time of each drone in MILP (2) - (27), instead, we consider $\overline{\tau}^k_D$, i.e., the average arrival time of the set of drones D associated to vehicle k. Then, in the given set of D drones, $\overline{\tau}^k_D$ must always be less than or equal to the earliest arrival time τ^k_{max}, that is:

$$\overline{\tau}^k_D = \frac{1}{D}\sum_{i=1}^{D} \tau^k_i \leq \tau^k_{max} \leq \tau \ : \ \forall k \in K. \tag{34}$$

If we divide both sides of (32) by D, i.e., the number of drones, and take into account the inequalities (34), we obtain the following valid inequalities that provide a lower bound on the makespan τ:

$$\frac{1}{D}(\sum_{i\in V_L}\sum_{w\in V_N} \bar{t}_{iw} y^{ks_0}_{iw} + \sum_{w\in V_N}\sum_{j\in V_R} \bar{t}_{wj} \tilde{y}^{ks_n}_{wj}) \leq \tau \ : \ \forall k \in K. \tag{35}$$

- **(ii)** For $|S| \geq 3$, the more general case applies:

Here, the relaxation provided by (35) becomes weak because we only account for operations that occur directly on vertices, i.e., at $s \in \{0, s_n\} \subset S$. Through the introduction of the binary variables y^{ks}_{iwj} and the inequalities (30) and (31) in Lemma 1, we can improve the bounds given in (35). More precisely, for any drone operation that either starts or ends at a discrete step $s \in S_L \cap S_R$ and does not coincide with a vertex, we account for the time spent travelling by all drones as follows for each vehicle:

$$\sum_{i\in V_L}\sum_{w\in V_N}\sum_{j\in V_R}\sum_{s\in S_L\cap S_R} \bar{t}^{ws}_{ij} y^{ks}_{iwj} \ : \ \forall k \in K. \tag{36}$$

By analogously following the structure of proof shown for $|S| = 2$, we can consider (36) as an additional term to (35) that tracks the total time spent performing en route drone operations. This yields the following more general valid inequalities:

$$\frac{1}{D}\sum_{w\in V_N}(\sum_{i\in V_L} \bar{t}_{iw} y^{ks_0}_{iw} + \sum_{j\in V_R} \bar{t}_{wj} \tilde{y}^{ks_n}_{wj} + \sum_{i\in V_L}\sum_{j\in V_R}\sum_{s\in S_L\cap S_R} \bar{t}^{ws}_{ij} y^{ks}_{iwj}) \leq \tau \ : \ \forall k \in K, \tag{37}$$

which is the desired conclusion of Proposition 2. ■

4. Variable Neighborhood Search for Solving the VRPDERO

As we will see in more detail in Section 5, only small-sized problem instances can be solved to optimality by a commercial solver software within reasonable run time. In order to overcome this issue, in this Section, we introduce an algorithm which exhibits features of *Variable Neighborhood Search* (VNS) and *Tabu Search* (TS) for solving the VRPDERO.

Variable Neighborhood Search (VNS) is a metaheuristic framework that was introduced by Mladenović and Hansen (1997) for solving (combinatorial) optimization problems. Given an incumbent solution x, the strategy of VNS consists in systematically visiting increasingly distant k-th neighborhoods of the current incumbent solution x, i.e., $\mathcal{N}_k(x)$. The new neighborhoods are then explored by a local search method in order to identify the local optima. A new solution is accepted if and only if an improvement is made. Several extensions to the basic VNS have been proposed and an overview of the proposed methods is given by Gendreau and Potvin (2010) and Hansen et al. (2017).

Tabu Search (TS) was introduced by Glover (1986) as a metaheuristic for guiding and controlling heuristics. The basic idea is to identify certain moves as forbidden, i.e., *tabu*, to prevent cycling. Glover (1986) differentiates *strong inhibition*, i.e., if a move is identified as tabu then it will never be chosen in the future (e.g., found in branch-and-bound methods) from *weak inhibition*, which consists of a short-term memory.

The basic structure of our proposed procedure is shown in Algorithm 1. A detailed explanation of all components is given in the subsequent sections. Indeed, this framework structurally reflects the basic VNS by Mladenović and Hansen (1997) that consists of an *initialization*, a *shaking* step that moves the current solution x to the k-th Neighborhood $\mathcal{N}_k(x)$, and a *local search* procedure. For adapting VNS to the VRPDERO, we propose decomposing the problem into two natural components: *route generation* and *drone operation insertion*. In our case, the local search procedure is a problem specific operator that, given a routing x (that consists of a route for each vehicle), generates a valid VRPDERO solution y through the insertion of drone operations (line 9 of Algorithm 1). Furthermore, in order to avoid cycling, a tabu list is implemented in the algorithm (lines 6-7 of Algorithm 1).

Finally, the *neighborhood change* operator shown in Algorithm 2 handles value updates during each iteration. In case of an improvement (or an equally good objective value), we store the incumbent solution y^* and the routing x associated with this solution. Furthermore, as we will see in Sections 4.2 and 4.3, the local search operator is a deterministic procedure. Therefore, we add x to a tabu list that will prevent future evaluations of the same routing. Note that, in principe, different routings x may lead to the same VRPDERO solution y after the drone insertion procedure. Therefore, an equally good objective value also triggers a change in the incumbent value to enable an effective exploration of the search space.

Algorithm 1 Adaptation of Basic VNS

```
1: $x \leftarrow$ Initialization() {refer to Section 4.1}
2: while The termination criteria is not met do
3:     $k \leftarrow 1$
4:     while $k < k_{\max}$ do
5:         $x' \leftarrow$ Shake$(x, k)$
6:         if TabuList.contains$(x')$ then
7:             Continue
8:         end if
9:         $y \leftarrow$ LocalSearch$(x')$ {refer to Sections 4.2 - 4.3}
10:        $x, y, k \leftarrow$ NeighborhoodChange$(x', y, k)$ {refer to Algorithm 2}
11:    end while
12: end while
13: return $y$
```

Algorithm 2 Neighborhood Change

```
1: if $f(y) \leq f(y^*)$ then
2:     $y^* \leftarrow y$
3:     $x \leftarrow x'$
4:     $k \leftarrow 1$
5:     TabuList.add$(x)$
6: else
7:     $k \leftarrow k + 1$
8: end if
```

In what follows, the integrated components of Algorithm 1 will be described in more detail. More precisely, in Section 4.1 we demonstrate the procedure that is used for the *initialization*, *local search*, and *shaking*. Section 4.2 is devoted to the presentation of the drone insertion procedure. As the drone insertion procedure is a deterministic search operator, we embed it within a *divide and conquer* approach that is well-suited for parallel computing architectures. This approach is presented in Section 4.3. We conclude this part in Section 4.4 with further remarks on the heuristic procedure and on some potential pitfalls that might be encountered.

4.1. Route generation

During the initialization procedure, first, a set of routes is generated through the parallel savings heuristic (Clarke and Wright, 1964). As vehicles are by definition not associated with a capacity in the VRPDERO, we generate the routes in such a way that each route may contain at most $\lceil |V_N|/|K| \rceil$ customers.

Then, for a limited number of iterations, a local search procedure is applied. Within this procedure, we employ the *2-opt*, *string relocation* (SR), and *string exchange* (SE) operators (Lin and Kernighan, 1973; van Breedam, 1994). More precisely, the 2-opt operator will invert a *randomly* chosen subsequence of a route, SR involves a relocation of a *random* customer from one route to another route and SE invokes an exchange of two *random* customers from two distinct routes. When a vertex is moved through SR or SE, we insert it at the position that will add the least additional length to the affected route. A 2-opt move is only applied, if the move will decrease the length of the original route. Similarly, SR and SE are only applied if the move leads to a reduction of the maximum length of the routes that are involved in the move.

For shaking, we also use the 2-opt, SR, and SE operators that we described above. More precisely, for k iterations, first, a random move is selected. Then, one route (or two routes, in case of SR or SE) is selected and the move applied. However, as opposed to local search, we also accept an increase in the route length after the application of any move.

4.2. Drone Insertion

So far, we did not consider drone operations during the construction of the routes, local search, or shaking. At this point, we are interested in a procedure that transforms a set of routes into a VRPDERO solution through introduction of drone operations. The drone operations should be inserted within each route in a manner, such that the makespan is minimized. For the insertion of drones in an existing route, we apply the following procedure for each route π represented by the vector x (see Algorithm 1). Assume that each route $\pi = \{0, \dots, n+1\}$ is given as a permutation of vertices. Set $V_L = \pi \setminus \{n+1\}$, $V_R = \pi \setminus \{0\}$, and $V_N = V_L \cap V_R$. Then, we perform a greedy search strategy that iteratively looks for a drone operation (i, s_l, w, s_r, j) (see Section 3.1 for its definition) that satisfies the following criteria:

1. A significant reduction in the length of the vehicle's route is achieved through the assignment of a customer, which is currently served by the vehicle, to a drone.
2. The *temporal delay*, that is incurred at the retrieval location through the synchronization requirement, is minimized.

The first criterion can be evaluated very quickly and is guided by the following formula, where d_{ij} is the distance between nodes i and j (see Section 3.1), w is the customer associated with the drone operation, and w_p and w_s are the direct predecessor and successor of w in the route π:

$$\gamma(w) \coloneqq d_{w_p,w_s} - d_{w_p,w} - d_{w,w_s} \tag{38}$$

Clearly, the first criterion only depends on the customer w associated with the drone operation (i, s_l, w, s_r, j). Therefore, the second criterion integrates the launch and retrieval locations (i, s_l) and (s_r, j), respectively. More precisely, the second criterion considers the temporal difference in arrival time between the vehicle and drone at the retrieval location (that is defined by (s_r, j) in the current operation) after the assignment of customer w to a drone. In order to calculate this value, given the starting time defined at (i, s_l), we calculate the arrival time of the vehicle and drone, denoted by $a_j^{s_r}$ and $\overline{a}_j^{s_r}$, respectively. The temporal delay is then calculated as an absolute value as follows:

$$\lambda(i, s_l, w, s_r, j) = |a_j^{s_r} - \overline{a}_j^{s_r}| \tag{39}$$

Based on this formula, a large value indicates that either the drone must wait for the vehicle or the vehicle must wait for the drone for a long time. In the first case, if a drone arrives much earlier than a vehicle, it is unlikely that a good utilization will be achieved as the drone spends much time waiting. In this case, it can be beneficial to look for an earlier retrieval location such that the temporal delay is reduced and, in this sense, fast synchronization achieved. In the second case, if a vehicle arrives much earlier than a drone, it must wait for the drone to arrive before it can continue its tour. In this case, it can be worthwhile to investigate if the drone can not be retrieved at a later point in the tour such that time spent waiting by the vehicle is turned into time spent moving.

Given an initial route, these criteria are then embedded in an algorithm as follows. First, we look for $w^* = \arg\max_w \gamma(w)$. Note that w^* must be feasible, i.e., it does not serve as a launch or retrieval location for any previously chosen operation. Afterwards, the operation $(i, s_l, w, s_r, j)^* = \arg\min_{(i,s_l,w,s_r,j)} \lambda(i, s_l, w, s_r, j)$ is identified (for the domain of (i, s_l, w, s_r, j), refer to Section 3.1). This operation must be feasible with respect to the drone's endurance, the number of drones in operation at any point, and any other links between the route of the vehicle and drone operations that are implied by MILP (2) - (27). As we will see in the following section, an operation may also be prohibited, i.e., tabu. If the operation is feasible and not tabu, we add it to the list of operations and iterate until no more feasible or improving operation can be found.

4.3. Divide and Conquer

The drone insertion procedure that was introduced in the previous section is summarized in Algorithm 3. This procedure is a very fast and deterministic operator that performs a greedy search that is guided by the criteria γ and λ. While the results returned by this procedure are solid in many cases, it may be of interest to perform a more thorough and in-depth exploration of possible drone operations. To this end, we propose a search space, called $\mathcal{N}(1)$ search space, where the neighborhood relationship is

Algorithm 3 Drone Insertion Heuristic

```
1: while The termination criteria is not met do
2:     for Customer $w$ with maximum $\gamma(w)$ in the route $\pi$ do
3:         for Operation $(i, s_l, w, s_r, j)$ with minimum $\lambda(i, s_l, w, s_r, j)$ that is feasible do
4:             if The operation is not tabu (see Section 4.3) then
5:                 Insert $(i, s_l, w, s_r, j)$ and remove $w$ from $\pi$
6:             end if
7:         end for
8:     end for
9: end while
```

a single drone operation, i.e., two neighboring solutions differ through exactly one drone operation. For the exploration of the $\mathcal{N}(1)$ search space, we embed the drone insertion procedure in a *divide and conquer* (D&C) approach (for a general description of a D&C algorithm, see, e.g., (Cormen et al., 2009) and references therein).

In order to describe our D&C approach, consider the rooted tree shown in Figure 3. In this tree, the root node is associated with the route π of a single vehicle where no drone operations occur. The children of each node then refer to operations that are inserted into π in a sequential manner. The depth of a node directly corresponds to the number of operations that have been inserted.

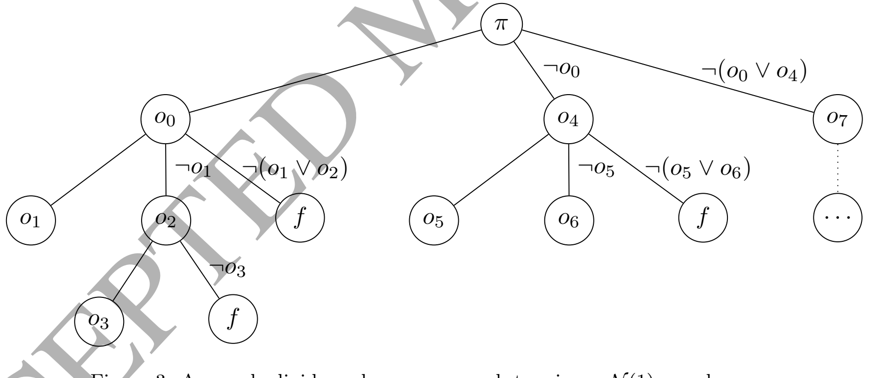

Figure 3: A sample divide and conquer search tree in an $\mathcal{N}(1)$ search space.

Within the divide and conquer approach, first, we apply the drone insertion procedure (Algorithm 3) which generates the leftmost branch. This branch indicates that o_0 is the first operation inserted in the route, afterwards o_1 is inserted, and then, no more operation can be found. As the leaf node o_1 has no children, we say that this node has been fully *fathomed*. After the generation of the initial branch, we follow a divide and conquer approach and explore the tree in a *depth-first* manner. More precisely, after the initial (leftmost) branch has been generated, we start with node o_1, that has been fathomed, and move to its parent node. Then, we generate a new branch where the operation at depth 2 of the tree is prohibited to be o_1, i.e., $\neg o_1$, which generates the node o_2 and its child o_3 through Algorithm 3. This procedure is iterated according to Algorithm 4 until a termination criteria is met or the tree fully explored.

Algorithm 4 Depth-First Search in the $\mathcal{N}(1)$ Search Space

```
1:  Root $r \leftarrow \pi$
2:  $r$.addBranch() {refer to Algorithm 3}
3:  Node $n$ = r.getLeaf()
4:  **while** Termination criteria not met **do**
5:      **if** $n$.degree == 0 **then**
6:          $n$.fathomed ← true
7:      **end if**
8:      Node $p = n$.getParent()
9:      **if** ($p$.getRightMostChild.Operation == {}) **then**
10:         $p$.fathomed ← true
11:     **end if**
12:     **while** $p$.fathomed == true **do**
13:         $p = p$.getParent()
14:     **end while**
15:     $n = p$.addRightMostChild
16:     $n$.addTabus($n$.Siblings)
17:     $n$.addOperations($n$.Ancestors)
18:     $n$.addBranch() {refer to Algorithm 3}
19: **end while**
```

4.4. Comments

In essence, this divide and conquer procedure is well-suited to implementations that utilize parallel-computing architectures (see, e.g., (Kumar and Rao, 1987; Rao and Kumar, 1987)). More precisely, after the initial leftmost branch has been generated, branching can start on multiple nodes. This procedure can then be iterated whenever unexplored nodes are available, yielding a *parallel depth-first search* procedure. However, for the purpose of our preliminary study, we chose a more simple sequential depth-first search approach.

As no nodes are pruned in our procedure, in principle, we greedily search the complete operation space where the neighboring relation between two nodes is a single operation. This enumeration is easily possible for small instances that can be associated with small (in terms of vertices) routes at the root node. However, for large initial routes and an increase in the maximum permitted steps s_n, the size of the tree may become prohibitively large. In such a case, for providing a heuristic exploration of the tree, it is easily possible to introduce some criteria that limit the effort spent on the search. Possible criteria that limit the overall size of the tree include, e.g., limits on the number of nodes that have been processed in the tree, or a limit on the degree of each node before we consider it to be fathomed. As an alternative to restricting the size of the tree, it is also possible to reduce the time that is spent processing each node in the tree. More precisely, additional logic might be added to Algorithm 3 such that only a subset of drone operations is considered in the first place. Given the initial route π, let i_{id}, w_{id}, and j_{id} refer to the indices of vertices i, w, and j in π. Then, one possibility is to only consider those operations, where $|w_{id} - i_{id}|$ and $|j_{id} - w_{id}|$ do not exceed a given threshold. Typically, in optimal solutions, retrieval of drones occurs almost immediately after the launch such that a high utilization is guaranteed. Finally, a more general criteria might simply limit the run time spent on the exploration of the tree.

It is also important to consider that the $\mathcal{N}(1)$ search space may prevent the (globally) optimal solution to be found in some cases, if the feasibility of an operation is considered during each move in Algorithm 3. To illustrate this case, consider Figure 4, which shows an initial route (left). Assume that we have a single drone with limited endurance and the optimal drone operations are (a, s_0, b, s_1, d) and (a, s_1, c, s_2, d) (right). However, given the initial route and the $\mathcal{N}(1)$ search space, it is not possible to reach the optimal solution without visiting an intermediate solution that is infeasible. Similarly, the insertion of an operation that is feasible by itself can render existing operations, that were deemed feasible previously, infeasible. Therefore, careful attention is required when the feasibility of a solution is checked and a full exploration is only guaranteed if the feasibility is verified on a node-level in the rooted tree instead of a per-move basis in Algorithm 3. However, such delayed validation of operational feasibility (i.e., on a node instead of per-move basis) will increase the size of the tree.

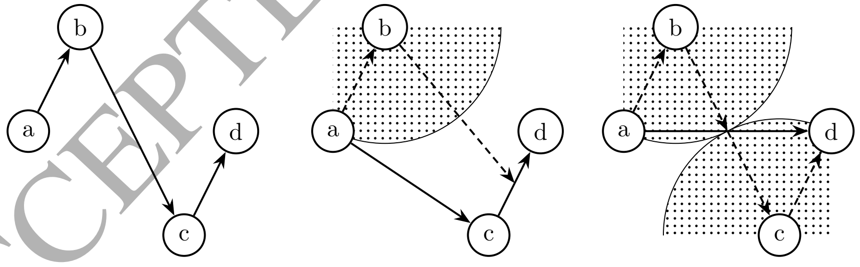

Figure 4: An initial route (left) and the optimal solution that features two drone operations (right). In this case, the optimal solution can only be constructed by generating an intermediately infeasible solution in the $\mathcal{N}(1)$-neighborhood (center). In this case, we assume that $s_n = 2$, i.e., we have $S = \{0, 1, 2\}$ which implies an available discrete point at the center of each arc.

Overall, the increase in discrete steps has significant influence on number of possible drone operations and consequently, the size of the tree and the potential degree of its nodes. In the MILP (2) - (27), the number of possible drone operations, which are reflected by combinations of variables $y_{iw}^{ks_l}$ and $\tilde{y}_{wj}^{ks_r}$, grows in the order of $\mathcal{O}(|S|^2)$ with the amount of discrete steps. As the route is assumed to be fixed when solving the drone insertion, given s_n, a route $\pi = \{0, a, \ldots, b, n+1\}$ with $\pi \setminus \{0, n+1\} \in V_N \setminus \{w\}$, and a vertex $w \in V_N$ that is set to be served by drone, the number of potential operations can be calculated according to formula (40), in which the first underbrace accounts for the drone operations that are performed on single arcs and the second underbrace accounts for all drone operations that involve multiple arcs. As an example, Figure 5 visualizes the number of possible operations for a route $\pi^k = \{g, h, j\}$ with just three vertices and $s_n = 2$. In this case, for $s_n = 1$, we have three possible operations that are highlighted in red. For $s_n = 2$, the number of possible operations already increases to 10 and, clearly, this amount grows not just with an increase in s_n but also with respect to the value of $|\pi^k|$.

$$\sum_{i=1}^{|\pi|-1} \underbrace{\Big(\sum_{s_l \in S_L} \sum_{\substack{s_r \in S_R \\ s_r > s_l}} 1 \Big)}_{\text{single arc}} + \sum_{i=1}^{|\pi|-2} \underbrace{\Big((|\pi| - 1 - i) \sum_{s_l \in S_L} \sum_{s_r \in S_R} 1 \Big)}_{\text{multiple arcs}} \tag{40}$$

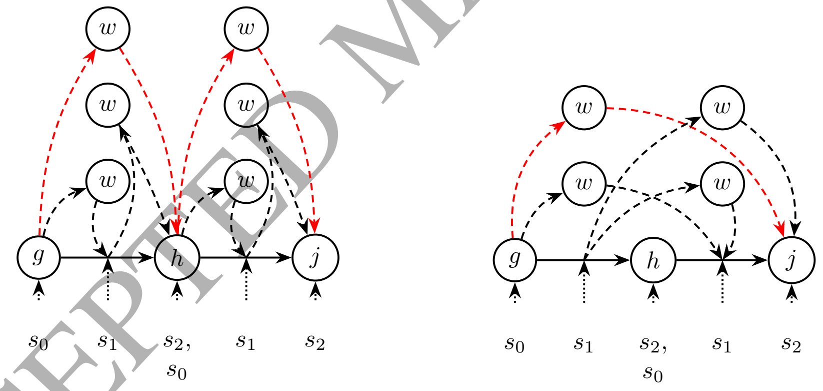

Figure 5: All possible drone operations for a given route $\pi^k = \{g, h, j\}$, a customer w that is served by drone and $s_n = 2$. The left-hand and right-hand side represent the first and second under brace in formula (40), respectively. The operations that are possible for $s_n = 1$ are highlighted in red.

5. Computational Experiments and their Numerical Results

We carried out two main classes of computational experiments, on small- and large-scale instances, and we present the numerical results in this section. More precisely, in Section 5.1 we discuss the performance of Gurobi Optimizer in solving MILP (2) - (27)

and compare its performance to our proposed heuristic. Afterwards, Section 5.2 shows that the heuristic can also be used to solve larger instances.

All experiments were performed on single compute nodes in an Intel® Xeon® Gold 6126 cluster, where each node operated at 2.6 GHz with 16 GB of RAM (hyper-threading disabled). We implemented our algorithms in Java SE 8 and for solving MILPs, we used Gurobi Optimizer 8.1.0 (Gurobi Optimization, 2018). The solver was allowed to utilize all cores of the processors, whereas our heuristic ran single-threaded. Since there is no commonly accepted benchmark for the VRPD, we performed our experiments on the benchmark instances used by Agatz et al. (2018). The instances are specified by a graph $\mathcal{G}(V, E)$ (that determines the location of the depot and customers), the relative velocity α, and the endurance $\mathcal{E}$. In particular, through a *relative endurance* parameter $\mathcal{E}_r$, the endurance $\mathcal{E}$ can be calculated as follows, where $max(E)$ refers to the edge with maximum weight in the graph $\mathcal{G}$:

$$\mathcal{E} \coloneqq \mathcal{E}_r \cdot max(E) \quad : \quad \mathcal{E}_r \in [0, 2]. \tag{41}$$

In this case, $\mathcal{E}_r = 0$ guarantees that no drone operation is possible while $\mathcal{E}_r = 2$ marks any operation as feasible. Furthermore, the vehicles and drones are assumed to both follow the Euclidean distance metric. For different numbers of vertices, Agatz et al. (2018) provide 10 instances with a uniform distribution of customers. Each instance is associated with a value $\alpha \in \{1, 2, 3\}$ and a value $\mathcal{E}_r \in \{0.2, 0.4, 0.6, 1.0, 1.5, 2.0\}$ yielding a total of $10 \cdot 3 \cdot 6 = 180$ different instances for a given size, i.e., $|V|$. Furthermore, by definition of (Wang et al., 2016; Poikonen et al., 2017), the number of vehicles and drones per vehicle are parameters in the VRPD (and therefore, also in the VRPDERO). As a meaningful selection, we limit ourselves to $|K| \in \{1, 2\}$ and $|D| \in \{1, 2\}$, i.e., we either have one or two vehicles that are equipped with either one or two drones each. In addition, to investigate the benefits of en route operations, we consider $s_n \in \{1, 2\}$, i.e., we either have 0, or 1 discrete steps on each arc, which can be used for en route operations. Figure 1 shows a sample VRPDERO solution.

5.1. Small Instances

For the purpose of studying the performance of the MILP solver and a comparison to our heuristic, we used instances with $|V| \in \{8, 9, 10\}$ vertices. For each size, we have ten different instances each of which is associated with three different values for the relative velocity α and six different values for the relative endurance $\mathcal{E}_r$. In total, this yields $3 \cdot 180 = 540$ instances. Based on the additional VRPDERO-specific problem parameters, i.e., the number of vehicles, drones, and discrete steps, we considered a total of $2^3 \cdot 540 = 4320$ problem instances in our experiments. Sections 5.1.1 and 5.1.2 are

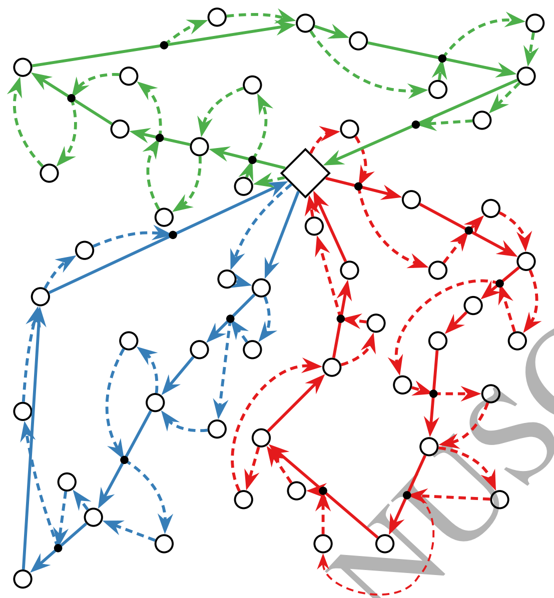

Figure 6: A sample VRPDERO solution on a problem instance with 50 customers, $|K| = 3$, $D = 1$, and $s_n = 2$. The solid and dashed lines represent the paths of the vehicles and drones, respectively.

dedicated to detailed numerical results that were obtained by solving these instances through the MILP solver Gurobi and our heuristic.

5.1.1. Results obtained through the MILP Formulation

In this section, we provide the results that were obtained by solving the MILP introduced in Section 3.1 using the solver Gurobi Optimizer 8.1.0 (Gurobi Optimization, 2018). More precisely, we solved these 4320 instances in two different ways:

1. We solved the MILP (2) - (27) while we included the VIEQ (29) - (35) that provide bounds on the makespan. For $s_n = 1$, we only included valid inequalities (29) and (35) in the model. For the general case, where $s_n \geq 2$, we also took the variables y^k_{iwj} into consideration, enabling us to add constraints (30) - (31) to the model. Through these additional variables and constraints, the valid inequalities (35) were replaced with the more general valid inequalities (37).
2. In MILP (2) - (27), the cardinality of rows required for constraints (16) - (17) that bind the values of the z^k_{aiwje} variables is very large. Furthermore, given the underlying set of these variables, the typical subset of variables in the solution vector with binary value 1 is very sparse and thus, they rarely influence constraints (27). Therefore, these constraints might be a good candidate for a row-generation

procedure. Following this logic, we also considered a case where these constraints were treated as *lazy constraints*. More precisely, whenever the solver detects a new candidate incumbent integer-feasible solution, we determine if any of constraints (16) - (17) are violated. If this is the case, the respective constraints are *lazily* added to the model, i.e., not before needed or only as necessary, and then, the solver's internal branch-and-cut procedure continues. For this case, we also included the valid inequalities according to the previous case (as we previously described).

When solving the VRPDERO as MILP, we limited the run time of the solver to 10 minutes. Note that, in principle, we could have also solved MILP (2) - (27) without the presence of any VIEQ. However, for all but trivial cases, this will lead to MIP gaps that are close or equal to 100 % after run time. As this is not very revealing, we decided to exclude them from the final experiments and analysis.

For the analysis of our results, we introduce the following metric. The value Δ indicates the relative change in makespan through the introduction of drones and is calculated as follows:

$$\Delta \coloneqq 100\% - \frac{\text{objective value after run time}}{\textit{optimal} \text{ objective value with } D = 0} \tag{42}$$

Note that the optimal objective value for $D = 0$ can be obtained within a few seconds by solving MILP (2) - (27).

The results are visualized in Figures 7 and 8; however, for the sake of completeness, the detailed numerical results are available in the appendix (refer to Tables A1 - A6). Overall, the problem parameters such as, e.g., the relative endurance parameter $\mathcal{E}_r$, the number of vehicles $|K|$, the number of drones D, and the instance size have a significant influence on the run time of the solver. A small relative endurance (i.e., $\mathcal{E}_r \in \{0.2, 0.4, 0.6\}$) will often allow the solver to find the optimal solution within the allocated run time. Moreover, a change in the discrete steps might increase the run time significantly. As an example, even for the smallest instance ($n = 8$), the introduction of a discrete step on each arc can cause a nearly *tenfold* increase in run time to find the optimal solution.

In general, treating constraints (16) - (17) as lazy constraints has some benefits in the sense that most instances can be solved faster to proven optimality or with smaller gaps within the run time limit. Significant speed ups are gained when there is no en route possibility, i.e., $s_n = 1$. However, this procedure quickly approaches the quality of the non-lazy case as the instance size increases and, as a result, the run time limit is steadily approached by both approaches.

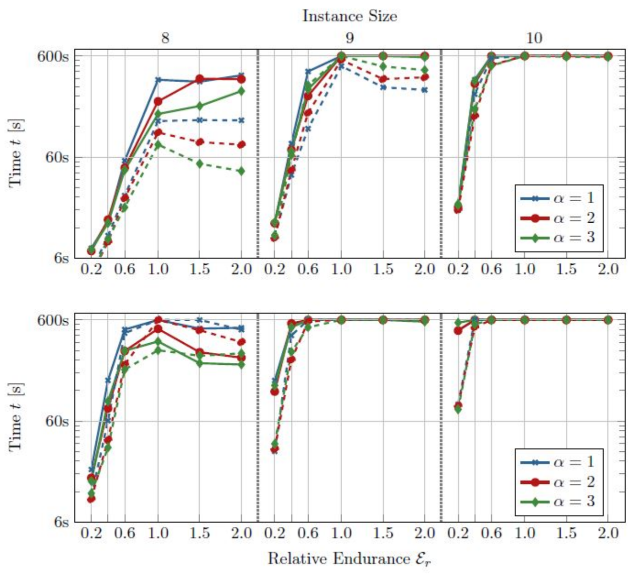

Figure 7: The average time t (shown on a logarithmic scale) required by the solver for different instance sizes, relative endurances $\mathcal{E}_r$, and relative velocities α. The lines are shown for $s_n = 1$ (upper figure) and $s_n = 2$ (lower figure) and a single vehicle equipped with a single drone (based on Tables A1 - A6). The solid lines represent the basic MILP and the dotted lines represent the case where lazy constraints were used.

Figure 7 visualizes the run time of the solver for both approaches that were described at the beginning of this section. Furthermore, Figure 8 highlights the remaining MIP gap after run time. These figures show the cases of a single vehicle that is equipped with a single drone for $s_n = 1$ and $s_n = 2$, respectively.

A peak can often be detected for the run time and MIP gap at $\mathcal{E}_r = 1$. This behavior is interesting as $\mathcal{E}_r = 1$ implies a much more constrained solution space than $\mathcal{E}_r \in \{1.5, 2.0\}$. Even though we have investigated this behavior deeply, as of right now, we have no in-depth explanation for the solver's behavior.

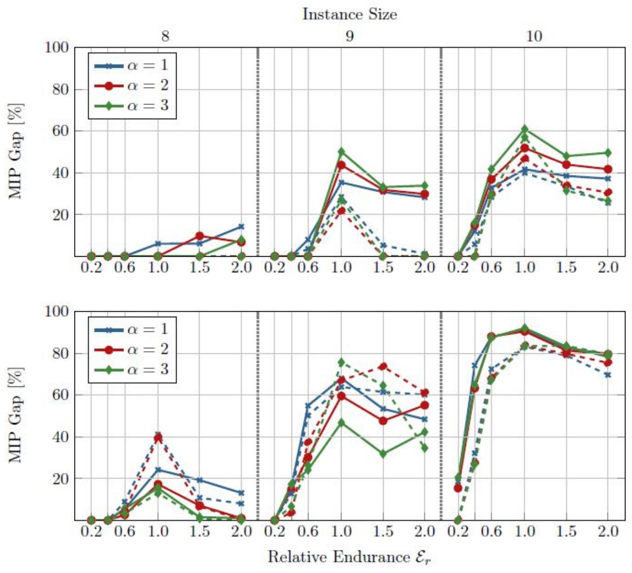

Figure 8: The average remaining relative MIP gap after run time returned by the solver for different instance sizes, relative endurances $\mathcal{E}_r$, and relative velocities α. The lines are shown for $s_n = 1$ (upper figure) and $s_n = 2$ (lower figure) and a single vehicle equipped with a single drone (based on Tables A1 - A6). The solid lines represent the basic MILP and the dotted lines represent the case where lazy constraints were used.

Similar observations can be made as the number of drones per vehicle D or the number of vehicles $|K|$ is increased. Typically, a single vehicle equipped with two drones can be tackled easier by the solver than the case of two vehicles that are equipped with one drone each. Moreover, the case of two vehicles equipped with two drones each has proven itself to be the most difficult case. In all of these cases, generally, the average run time and MIP gaps increase tremendously compared to the basic case of $|K| = D = 1$ (see Tables A1 - A6). Clearly, the number of columns and rows of MILP (2) - (27) are independent of the number of drones, i.e., D (only the right hand side in constraints (27) is affected). Opposed to this, an increase in the number of vehicles increases both, the number of variables and constraints, significantly.

To sum up, based on the recorded Δ values in Tables A1 - A6, significant makespan-savings are possible through the introduction of drones. These values vary significantly with the drone parameters and hence, care should be taken when designing a drone fleet in practice. Moreover, the introduction of a discrete step on each arc typically reduces the makespan by an additional 2 - 3 % with respect to the non-en route cases. Compared to this, a second drone allows for significantly increased savings that typically fall in the range of 5 - 15 %. In both cases, the increased savings depend strongly on the relative velocity and endurance of drones. This is also illustrated in Figure 9.

5.1.2. Results obtained through the Heuristic

In this section, we provide the numerical results that were obtained by solving the small instances through the heuristic introduced in Section 4. We begin by explaining the parameters that were used for the implementation of the heuristic. For the initialization procedure described in Section 4.1, after the initial generation of routes, $|V|^3$ iterations were spent on local search. We ran the heuristic five times, each time using a different random seed, for at most 10 minutes. If the incumbent solution remained unchanged for one minute, the heuristic terminated ahead of time. The value $k_{\max}$, that determines the maximum depth of the neighborhood within our VNS procedure, was set to $k_{\max} = 6$.

For the determination of feasible moves in the drone insertion heuristic (Algorithm 3), we only considered the limit on the number of drones currently in operation and the endurance with respect to the current operation. Recall that, as mentioned in Section 4.4, the insertion of a feasible operation can render previously feasibly inserted operations infeasible. The feasibility of a solution (defined by multiple operations) was then determined whenever a new branch was added to the rooted tree (Section 4.3). For the depth-first search of this tree, we impose a maximum degree of 30 on all nodes (including the root), before considering it to be fathomed. Alternatively, in each iteration of Algorithm 1, we canceled the exploration of the tree after 5000 calls of Algorithm 3.

The detailed results can be found in Tables A7 - A9 (see the Appendix). With respect to the results achieved by the MILP solver, overall, our heuristic demonstrated desirable behavior. In what follows, we will compare the results of our heuristic to the best results achieved by the MILP solver through either approach (best results from Tables A1 - A3 and Tables A4 - A6). For the 4320 different problem instances and 5 runs per heuristic, the heuristic often found a solution that was as good as or better than the one provided by the solver in 18569 out of 21600, i.e., 85.96 % of all cases. Given that not every instance was solved to optimality by Gurobi after run time, the results returned by the heuristic were on average 3.6 % better in terms of the objective value. Therefore, it is fair to conclude, as backed up by the detailed results, that for the cases where the heuristic performed worse than the solver, the solution quality did not differ much. The average run time for the heuristic was 106.56 seconds, i.e., on average the final solution was detected within $106.56 - 60 = 44.76$ seconds. Note that our neighborhood change (see Algorithm 2) is also triggered when the same objective value is found through a different routing (for an explanation, see Section 4). Therefore, the same solution (that is constructed through a different routing) might often be found earlier. As was the case for the MILP, run time behavior was significantly influenced by the discrete steps, the number of drones, and endurance of drones. Clearly, all of these parameters influence the size of the tree explored during divide and conquer.

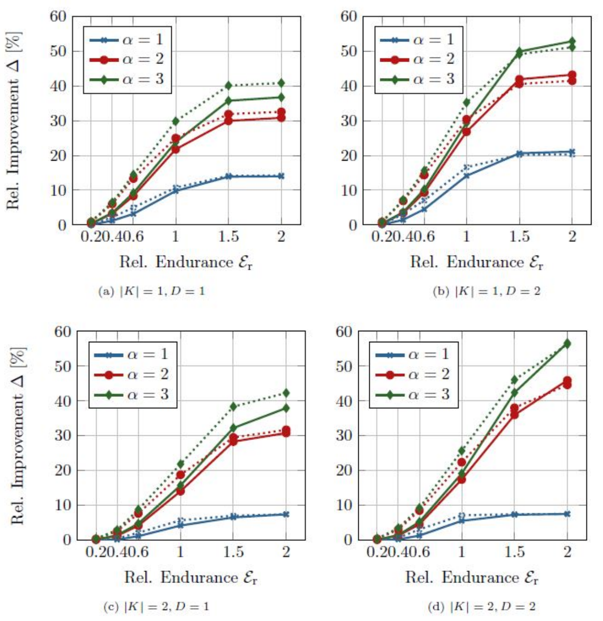

Figure 9: The average relative makespan reduction Δ depending on the number of vehicles $|K|$, the number of drones per vehicle D, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on Tables A7) - (A9). Lines shown for $s_n = 1$ (solid) and $s_n = 2$ (dotted).

Figure 9 visualizes the relative change in the makespan Δ (defined by formula (42)) for the cases that were recorded for the heuristic. These figures provide an insight in the savings through allowing an en route operation and the influence of the relative endurance $\mathcal{E}_r$ and relative velocity α. A look at this figure confirms the observations that were already mentioned in the previous section. More precisely, the introduction of a discrete step increases the savings by typically 2 - 3 %. The introduction of a second drone is often much more effective, albeit, the relative endurance and velocity must be sufficiently be large. Overall, the relative improvement by increasing α or $\mathcal{E}_r$ diminishes as these values are increased. Furthermore, the benefits of fast drones are enhanced as they possess greater endurance (as is the other way around). These observations also apply to the cases where multiple vehicles are present. For these small instances, in the presence of two vehicles, the average number of customers served by each vehicle is very small. Therefore, drones often seem very effective in these cases. Care must be taken when comparing the cases of one vehicle to those where two vehicles are present as the denominator when calculating Δ also changes.

5.2. Large Instances

In this section, we show that our heuristic is also suitable for tackling larger VR-PDERO instances. To this end, we tested on reasonably-sized instances with either 20 or 50 vertices that are within the scope of those that were used for related problems (see, e.g., Agatz et al. (2018); Bouman et al. (2018); Murray and Chu (2015)). On these instances, we tested according to the specifications given at the beginning of Section 5. Deviating from that, we only used $\mathcal{E}_r \in \{0.2, 0.4, 0.6, 1.0\}$ as any larger value on the relative endurance had no significant impact (see also Figure 9). Hence, in total, we tested on 1920 medium-sized instances. On each of these instances, we ran our heuristic five times with the time restrictions that were introduced previously.

With respect to the description of the parameters that were used for small instances (see Section 5.1.2), we adjusted some of them as follows. In order, to perform a more focused exploration of the search space, the maximum depth of the neighborhood has been reduced to $k_{\max} = 3$. Furthermore, to achieve a better tradeoff between diversification (route generation) and intensification (exploration of the rooted tree), some of the parameters for the divide and conquer approach were adjusted. More precisely, a maximum degree of 10 on the root node and 5 on all other nodes was imposed, before we considered them to be fathomed. Alternatively, we canceled the search after 10^4 calls of Algorithm 3.

The detailed results are available in Table A10 of the appendix. In general, our heuristic still performs well on these instances. On average, the final solution is found within just below two minutes of run time (after 111 seconds). As there are no other results to compare ourselves to, for this section, we focus on the performance differences of our heuristic in solving these instances with regards to en route operations. We have seen in Section 5.1 that en route operations significantly increase the computational complexity of the problem. Therefore, we are interested to know if our heuristic is able to process larger instances as well.

Figure 10 compares the average objective values achieved through the heuristic while en route operations are allowed ($s_n = 2$) to the cases where they are not allowed ($s_n = 1$). More precisely, for each combination of vehicles and drones, this figure contains a plot for $|V| \in \{20, 50\}$ and the different combinations of drone parameters.

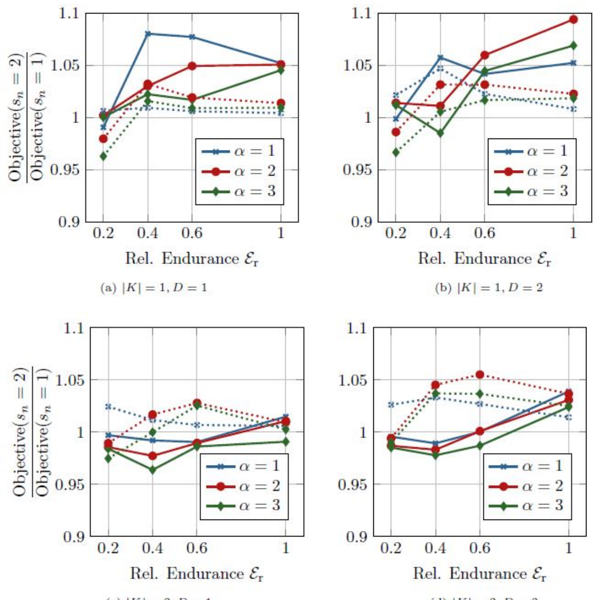

Figure 10: The average makespan shown as relative values for the en route and basic case depending on the number of vehicles $|K|$, the drones per vehicle D, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on Table A10)). Lines shown for $|V| = 20$ (solid) and $|V| = 50$ (dotted).

Overall, the much more difficult case of en route operations is still handled well by our heuristic. As we already observed in the previous section, enabling en route operations can in some cases allow saving in the order of 2 - 3 %. Typically, such savings are achieved if the relative velocity is large and the relative endurance small. However, it is also evident that the solutions found for the en route case are in some cases of slightly worse quality than those found for the basic case. As was already evident on small instances, the consideration of en-route operations increases the size of the search space tremendously. As a result, much more effort has to be spent during the divide and conquer approach to identify the best drone operations in the en route case, as the rooted tree might become significantly larger. For these cases, an implementation that provides a parallel depth-first search or prunes parts of the search tree might yield a significant speed-up.

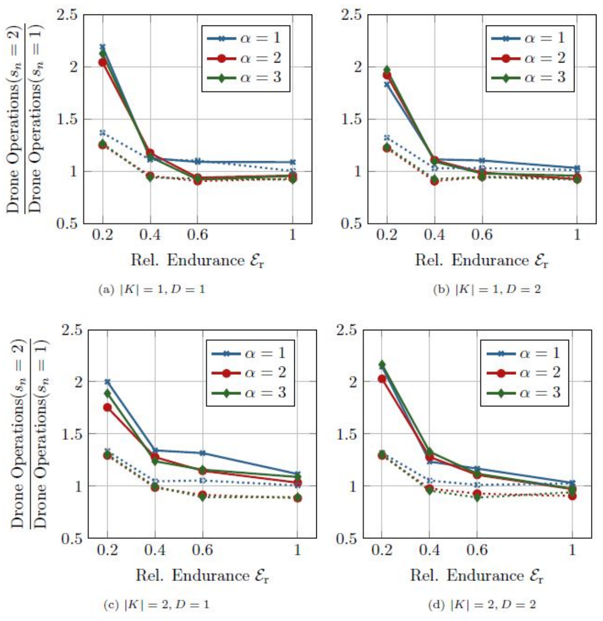

Figure 11: The average number of drone operations shown as relative values for the en route and basic case depending on the number of vehicles $|K|$, the drones per vehicle D, the relative endurance $\mathcal{E}_r$, and the relative velocity α (based on Table A10)). Lines shown for $|V| = 20$ (solid) and $|V| = 50$ (dotted).

In a similar manner, Figure 11 gives some insights into the structural differences when drone operations are allowed. More precisely, this figure visualizes the increase (decrease) of drone operations. This figure (and the respective plots within it) highlight that solutions differ notably at a small relative endurance values (i.e., $\mathcal{E}_r \in \{0.2, 0.4\}$), where the number of operations can be larger in the en route case. Notably, there is no large variation in these plots with a change in the relative velocity α. However, as we have seen in Figure 10, an increase in α can nevertheless reduce the makespan significantly.

6. Conclusion

In this paper, we have introduced the VRPDERO, which consists of a combination of multiple trucks and drones in a last-mile delivery setting. The significant difference between the VRPDERO and VRPD is that the former allows drones to be launched and retrieved on arcs by the vehicle; more precisely, drones are allowed to be launched and retrieved on certain discrete steps on each arc. We formulated the VRPDERO as a mixed integer linear programm. Then, we proposed valid inequalities in order to enhance the performance of the solver and showed the effectiveness of the valid inequalities through computational experiments. However, we are only able to solve some small-scale problems to optimality through Gurobi Optimizer within reasonable run time. Therefore, we introduced a heuristic, which is based on the concepts of Variable Neighborhood Search and Tabu Search. Within this heuristic, as a local search operator, we introduced a drone insertion procedure that was embedded within a divide and conquer approach. We provided an extensive computational study that considered the impact of several problem parameters on, e.g., the makespan and run time.

Overall, through our numerical results, we are able to show the potential of a reduced makespan and a higher utilization of drones if we consider the possibility of launching and retrieving drones en route. Notably, en route operations seem most beneficial when either the drone endurance is small or the relative velocity is sufficiently large. In the former case, the drone is able to perform operations that would otherwise not be feasible. In the latter case, the drone is able to perform multiple operations in a most efficient way. However, we have also seen that the introduction of en route operations increases the computational complexity of the problem significantly. As a next step, it might be worth to refine our drone insertion operators. Naturally, it might be interesting to investigate the performance of a parallel depth-first search procedure. Moreover, a procedure to prune parts of the search tree might significantly increase the performance, in particular in solving larger instances. Finding alternative and possibly compact MILP formulations might also be an interesting research direction.

In this paper, we only considered a basic model with possibility of en route operations and assumed that the battery is recharged instantaneously after each delivery and discharged only if the drone is moving. Furthermore, possible effects of service times were neglected. In a practical setting, we might also be interested to investigate more realistic battery life considerations, recharge models, and the impact of potentially stochastic service times. For the moment, we leave them as future research plans.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions that helped us in improving the quality of this paper.

References

Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization Approaches for the Traveling Salesman Problem with Drone. Transportation Science 52, 965–981. doi:10.1287/trsc.2017.0791.

Bouman, P., Agatz, N., Schmidt, M., 2018. Dynamic programming approaches for the traveling salesman problem with drone. Networks 72, 528–542. doi:10.1002/net.21864.

van Breedam, A., 1994. An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a selection of problems with Vehicle-related, Customer-related, and Time-related Constraints Contents. Ph.d. dissertation. University of Antwerp.

Clarke, G., Wright, J.W., 1964. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research 12, 568–581. doi:10.1287/opre.12.4.568.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Clifford, S., 2009. Introduction to Algorithms. Third ed., MIT Press.

Di Puglia Pugliese, L., Guerriero, F., 2017. Last-Mile Deliveries by Using Drones and Classical Vehicles. Optimization and Decision Science 217, 557–565. doi:10.1007/978-3-319-67308-0_56.

Drexl, M., 2012. Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints. Transportation Science 46, 297–316. doi:10.1287/trsc.1110.0400.

Gendreau, M., Potvin, J.Y., 2010. Handbook of Metaheuristics. International Series in Operations Research & Management Science. 2 ed., Springer US, Boston, MA. doi:10.1007/978-1-4419-1665-5.

Gentry, N.K., Hsieh, R., Nguyen, L.K., 2016. Multi-use UAV docking station systems and methods.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13, 533–549. doi:10.1016/0305-0548(86)90048-1.

Gurobi Optimization, 2018. Gurobi Optimizer Reference Manual.

Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S., 2017. Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization 5, 423–454. doi:10.1007/s13675-016-0075-x.

Kumar, V., Rao, V.N., 1987. Parallel depth first search. Part II. Analysis. International Journal of Parallel Programming 16, 501–519. doi:10.1007/BF01389001.

Lin, S., Kernighan, B.W., 1973. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research 21, 498–516. doi:10.1287/opre.21.2.498.

Marinelli, M., Caggiani, L., Ottomanelli, M., Dell'Orco, M., 2018. En route truck–drone parcel delivery for optimal vehicle routing strategies. IET Intelligent Transport Systems 12, 253–261. doi:10.1049/iet-its.2017.0227.

Miller, C.E., Tucker, A.W., Zemlin, R.A., 1960. Integer Programming Formulation of Traveling Salesman Problems. Journal of the ACM 7, 326–329. doi:10.1145/321043.321046.

Mladenović, N., Hansen, P., 1997. Variable Neighborhood Search. Computers & Operations Research 24, 1097–1100. doi:10.1016/S0305-0548(97)00031-2.

Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies 54, 86–109. doi:10.1016/j.trc.2015.03.005.

Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. Networks 72, 411–458. doi:10.1002/net.21818.

Poikonen, S., Wang, X., Golden, B., 2017. The vehicle routing problem with drones: Extended models and connections. Networks 70, 34–43. doi:10.1002/net.21746.

Rao, N.N., Kumar, V., 1987. Parallel depth first search. Part I. Implementation. International Journal of Parallel Programming 16, 479–499. doi:10.1007/2FBF01389000.

Schermer, D., Moeini, M., Wendt, O., 2018a. A Variable Neighborhood Search Algorithm for Solving the Vehicle Routing Problem with Drones (Technical Report).

Schermer, D., Moeini, M., Wendt, O., 2018b. Algorithms for Solving the Vehicle Routing Problem with Drones. Lecture Notes in Artificial Intelligence 10751, 352–361. doi:10.1007/978-3-319-75417-8_33.

Toth, P., Vigo, D., 2002. The vehicle routing problem. SIAM monographs on discrete mathematics and applications, Society for Industrial and Applied Mathematics, Philadelphia, Pa.

Wang, X., Poikonen, S., Golden, B., 2016. The vehicle routing problem with drones: Several worst-case results. Optimization Letters 11, 679–697. doi:10.1007/s11590-016-1035-3.

Appendix

This appendix contains the detailed numerical results. All experiments were performed on the uniformly distributed instances used by Agatz et al. (2018).

For each respective instance size and problem parameters, the values provided in the tables are shown as average values over 10 instances. Tables A1 - A6 show the results that were obtained through Gurobi Solver. More precisely, Tables A1 - A3 show the results that were obtained by solving MILP (2) - (27). For $s_n = 1$, valid inequalities (29) and (35) were added to the model. For $s_n = 2$, we also added the variables y_{iwj}^k, enabling us to include constraints (30) - (31) to the model. In this case, the valid inequalities (35) was replaced with the more general valid inequalities (37). Tables A4 - A3 show the results that were obtained by treating constraints (16) - (17) as lazy constraints. Apart from the problem parameters such as the number of vehicles, number of drones, discrete steps, relative velocity and relative endurance, the columns show the average savings Δ (see formula 42), the average run time t and the average remaining MIP gap after run time.

Furthermore, Tables A7 - A9 and A10 show the results that were obtained by solving the small and large instances through the proposed heuristic, respectively. These tables also show the average savings Δ, the average run time t, and the average number of drone operations (Ops) that were present in the solution vector. As no guaranteed-global optimal solution for the non-drone case is available for larger instances, Table A10 shows the objective value (Obj.), provided by the heuristic, as real numbers.

Table A1: Results obtained through solving the MILP on instances with 8 vertices.

			$\lvert K\rvert = 1$						$\lvert K\rvert = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.1%	7.5	0.0%	0.1%	7.9	0.0%	0.0%	23.9	0.0%	0.0%	21.0	0.0%
		0.4	0.9%	13.8	0.0%	1.0%	13.1	0.0%	0.0%	51.7	0.0%	0.0%	66.0	0.0%
	1	0.6	3.0%	55.1	0.0%	3.5%	48.4	0.0%	0.7%	553.0	23.4%	0.8%	459.4	9.0%
		1	9.4%	348.6	6.0%	12.6%	423.5	14.8%	3.0%	603.9	46.3%	5.0%	601.0	49.8%
		1.5	13.1%	333.9	6.1%	19.9%	546.3	33.4%	6.3%	603.4	44.6%	6.6%	601.8	51.8%
		2	13.1%	385.9	14.2%	20.4%	556.5	35.8%	7.0%	603.5	43.3%	7.2%	602.0	61.0%
		0.2	0.5%	7.1	0.0%	0.5%	8.4	0.0%	0.4%	27.0	0.0%	0.4%	23.0	0.0%
		0.4	2.2%	14.4	0.0%	2.5%	11.2	0.0%	1.5%	45.1	0.0%	1.7%	58.3	0.0%
1	2	0.6	5.3%	46.9	0.0%	6.0%	38.7	0.0%	3.9%	450.4	18.9%	4.7%	386.9	13.2%
		1	20.2%	213.2	0.0%	24.2%	174.2	0.0%	12.0%	601.3	39.2%	13.4%	493.5	24.9%
		1.5	31.9%	357.4	9.7%	42.8%	188.4	5.8%	31.4%	474.2	32.5%	36.1%	340.5	8.2%
		2	33.2%	353.6	6.7%	45.6%	175.5	0.0%	37.2%	469.7	23.8%	47.5%	385.4	23.9%
		0.2	0.6%	7.3	0.0%	0.6%	6.9	0.0%	0.5%	21.2	0.0%	0.5%	24.2	0.0%
		0.4	2.5%	13.3	0.0%	2.9%	9.9	0.0%	1.8%	68.0	0.0%	1.9%	50.7	0.0%
	3	0.6	5.6%	44.5	0.0%	6.2%	41.5	0.0%	4.1%	427.2	18.1%	4.5%	400.9	6.8%
		1	22.5%	160.0	0.0%	26.3%	168.9	5.1%	16.0%	520.9	28.0%	17.6%	507.8	25.8%
		1.5	41.1%	191.9	0.0%	49.5%	104.6	0.0%	34.9%	396.8	28.2%	42.3%	358.0	12.3%
		2	41.8%	269.7	7.9%	53.8%	132.2	0.0%	43.4%	382.0	13.3%	63.2%	263.5	0.0%
Average			13.7%	156.9	2.8%	17.7%	147.6	5.3%	11.3%	351.3	20.0%	14.1%	313.6	15.9%
		0.2	0.2%	19.9	0.0%	0.3%	15.0	0.0%	0.0%	162.5	0.0%	0.0%	163.0	2.6%
		0.4	1.7%	151.5	0.0%	2.2%	91.8	0.0%	0.4%	546.8	28.7%	0.3%	521.6	25.4%
	1	0.6	4.3%	481.6	5.3%	5.6%	521.5	5.5%	1.5%	554.9	41.4%	1.2%	551.1	46.1%
		1	7.6%	604.0	24.2%	13.0%	604.1	36.5%	5.8%	601.5	47.1%	6.1%	602.6	52.8%
		1.5	12.6%	492.6	19.2%	19.8%	600.4	41.2%	6.8%	602.8	44.6%	7.2%	603.9	54.2%
		2	13.3%	501.5	13.1%	20.5%	600.2	30.2%	7.2%	603.0	41.0%	7.2%	603.6	61.7%
		0.2	0.7%	16.4	0.0%	0.7%	13.9	0.0%	0.5%	165.3	0.0%	0.5%	114.2	0.0%
		0.4	4.4%	79.8	0.0%	5.3%	83.0	0.0%	-0.5%	471.1	24.3%	2.1%	444.2	32.1%
2	2	0.6	11.0%	295.6	2.6%	11.3%	278.1	0.9%	7.3%	488.7	26.7%	7.3%	509.6	32.8%
		1	22.0%	490.0	17.3%	27.6%	321.9	16.6%	17.3%	530.1	27.7%	20.2%	480.8	18.6%
		1.5	33.3%	287.3	7.0%	44.8%	224.6	4.5%	34.0%	339.3	12.5%	37.3%	358.1	15.4%
		2	35.0%	254.0	1.0%	47.3%	262.2	1.5%	38.2%	399.4	5.3%	48.1%	433.3	8.9%
		0.2	0.8%	15.2	0.0%	0.8%	14.4	0.0%	0.5%	164.4	0.2%	0.5%	145.0	0.0%
		0.4	4.7%	94.0	0.0%	5.8%	76.0	0.0%	2.3%	426.5	17.5%	0.6%	470.5	25.5%
	3	0.6	10.1%	297.4	5.6%	11.1%	243.3	2.9%	8.0%	509.1	21.5%	7.0%	500.7	32.3%
		1	28.1%	367.4	15.4%	32.9%	280.1	10.2%	20.5%	531.9	22.4%	23.0%	436.1	22.6%
		1.5	43.2%	223.9	1.5%	52.7%	83.1	0.0%	41.7%	287.0	9.0%	45.3%	246.4	3.3%
		2	43.9%	217.1	1.0%	56.8%	104.1	0.0%	48.5%	282.1	4.1%	63.4%	235.7	0.0%
Average			15.4%	271.6	6.3%	19.9%	245.4	8.3%	13.3%	425.9	20.8%	15.4%	412.2	24.1%

Table A2: Results obtained through solving the MILP on instances with 9 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
				$D = 1$			$D = 2$			D=1			D=2	
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.0%	13.4	0.0%	0.0%	13.0	0.0%	0.0%	72.6	0.0%	0.0%	76.3	0.0%
		0.4	1.5%	81.5	0.0%	1.8%	61.2	0.0%	0.2%	493.9	17.7%	0.3%	475.7	23.5%
	1	0.6	3.4%	421.3	8.0%	5.2%	321.3	6.2%	1.4%	603.0	36.3%	1.5%	602.5	32.8%
		1	8.9%	604.2	35.3%	13.1%	603.8	48.0%	1.1%	601.3	54.9%	0.8%	600.9	67.9%
		1.5	13.7%	601.9	30.8%	20.0%	602.4	46.6%	5.8%	602.8	52.0%	7.5%	604.1	67.3%
		2	14.3%	601.8	28.2%	20.8%	602.0	43.9%	6.1%	602.3	50.9%	8.1%	604.3	66.3%
		0.2	0.1%	13.3	0.0%	0.1%	13.3	0.0%	0.1%	77.9	0.0%	0.1%	71.5	0.0%
		0.4	4.5%	70.7	0.0%	4.8%	57.2	0.0%	1.5%	450.6	19.9%	1.3%	448.7	28.0%
1	2	0.6	10.5%	241.6	0.0%	11.9%	230.5	0.0%	4.2%	601.9	32.1%	4.6%	601.2	38.3%
		1	19.8%	603.4	43.7%	24.7%	601.5	50.2%	9.0%	600.8	66.4%	11.7%	601.0	76.9%
		1.5	30.0%	602.8	31.8%	41.3%	540.8	19.3%	24.2%	600.7	56.7%	35.0%	600.8	55.0%
		2	30.9%	607.1	29.8%	43.7%	563.0	14.3%	26.1%	600.7	55.3%	43.3%	600.7	49.6%
		0.2	0.1%	13.6	0.0%	0.1%	13.4	0.0%	0.1%	79.0	0.0%	0.1%	75.6	0.0%
		0.4	4.6%	68.1	0.0%	5.0%	51.2	0.0%	0.5%	447.6	27.1%	0.9%	433.8	21.7%
	3	0.6	11.7%	280.8	0.0%	13.3%	259.0	0.0%	4.8%	602.0	34.8%	6.0%	602.2	44.3%
		1	24.0%	601.1	50.1%	26.0%	601.5	47.1%	11.6%	600.8	66.5%	9.8%	600.9	80.5%
		1.5	34.7%	603.6	33.1%	52.3%	424.3	5.3%	30.9%	597.0	49.5%	41.6%	553.9	33.0%
		2	34.4%	583.6	33.8%	53.7%	406.5	4.3%	34.0%	600.7	48.2%	53.4%	539.7	23.4%
Average			13.7%	365.5	17.9%	18.8%	331.4	15.8%	9.0%	490.9	37.1%	12.6%	483.0	39.4%
		0.2	0.1%	151.1	0.0%	0.1%	114.4	0.0%	-2.2%	546.2	33.6%	-5.6%	516.2	30.5%
		0.4	-1.4%	527.7	15.8%	1.8%	508.8	14.6%	-62.7%	601.2	64.8%	-53.5%	600.9	64.6%
	1	0.6	1.1%	600.8	54.9%	4.8%	601.1	47.3%	-2.2%	601.3	87.5%	-13.8%	601.5	89.3%
		1	2.2%	600.9	68.0%	12.7%	600.9	68.7%	-2.9%	602.0	92.8%	-6.6%	601.8	94.0%
		1.5	11.3%	600.9	53.3%	18.9%	602.2	65.4%	1.8%	601.1	82.5%	6.4%	601.2	86.9%
		2	10.4%	601.5	48.4%	20.9%	602.3	56.7%	5.9%	600.8	85.2%	8.1%	600.9	74.8%
		0.2	0.5%	117.3	0.0%	0.5%	127.3	0.0%	-0.1%	556.6	32.3%	-0.3%	538.8	28.1%
		0.4	2.5%	554.0	14.7%	6.9%	465.0	4.3%	-33.1%	601.1	56.2%	-49.3%	601.1	59.3%
2	2	0.6	10.5%	601.0	30.2%	11.0%	593.8	27.4%	0.2%	601.7	86.3%	-11.3%	601.4	83.1%
		1	13.5%	601.3	59.5%	26.8%	577.5	38.6%	-2.7%	602.3	93.5%	2.9%	602.0	93.6%
		1.5	25.1%	601.1	47.6%	38.2%	602.9	35.7%	16.3%	601.2	82.4%	23.3%	601.0	76.8%
		2	22.3%	603.9	55.1%	41.2%	603.1	30.6%	24.6%	600.8	74.9%	45.6%	601.3	26.6%
		0.2	0.6%	135.0	0.0%	0.6%	131.3	0.0%	-41.5%	526.7	28.7%	-1.8%	558.4	36.1%
		0.4	2.9%	506.3	17.2%	-1.0%	436.9	17.8%	-60.1%	600.9	65.3%	-59.4%	601.0	74.0%
	3	0.6	14.6%	599.4	24.1%	16.2%	597.7	21.0%	-1.2%	601.4	83.0%	-11.2%	601.4	82.6%
		1	20.9%	601.9	46.7%	35.3%	560.5	25.5%	-0.7%	602.3	94.2%	6.5%	601.8	92.8%
		1.5	34.1%	601.9	31.8%	53.1%	497.6	9.9%	21.5%	601.2	75.9%	32.8%	580.9	60.7%
		2	31.6%	576.7	42.3%	53.9%	589.9	14.3%	35.6%	601.0	56.2%	59.0%	573.2	12.0%
Average			11.2%	508.2	33.7%	19.0%	489.6	26.5%	-5.8%	591.7	70.9%	-1.0%	587.9	64.7%

Table A3: Results obtained through solving the MILP on instances with 10 vertices.

			$\lvert K\rvert = 1$						$\lvert K\rvert = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.1%	19.7	0.0%	0.1%	17.0	0.0%	0.1%	305.7	0.0%	0.1%	280.3	0.0%
		0.4	-0.8%	349.0	12.0%	-2.0%	323.2	15.6%	-0.9%	603.1	36.6%	-16.5%	602.0	54.3%
	1	0.6	-2.3%	601.9	32.8%	-3.8%	601.0	36.6%	-9.7%	601.1	63.5%	-1.0%	600.9	62.3%
		1	5.7%	600.7	41.6%	7.5%	600.9	57.0%	1.1%	601.7	68.6%	-0.3%	602.4	77.0%
		1.5	9.3%	600.8	38.5%	15.1%	600.8	49.5%	3.5%	601.7	66.9%	3.5%	601.2	73.4%
		2	9.5%	601.0	37.2%	16.3%	600.9	47.2%	4.8%	601.8	64.4%	5.9%	601.7	73.1%
		0.2	0.5%	19.1	0.0%	0.5%	20.3	0.0%	0.1%	327.4	1.2%	0.1%	278.1	0.0%
		0.4	-1.2%	319.5	14.9%	0.3%	299.8	15.4%	-4.9%	602.1	51.8%	-25.3%	591.2	64.5%
1	2	0.6	-4.1%	598.9	36.8%	-4.7%	567.7	38.2%	-7.7%	601.0	72.5%	-3.1%	600.9	75.9%
		1	15.9%	601.2	51.9%	21.6%	601.2	63.7%	6.8%	602.9	75.0%	9.1%	601.8	83.5%
		1.5	22.4%	601.2	43.9%	34.8%	600.9	51.1%	13.3%	601.5	71.4%	22.9%	601.4	78.2%
		2	23.1%	600.8	41.7%	35.4%	601.3	50.1%	19.0%	601.6	67.5%	28.1%	601.7	72.9%
		0.2	0.5%	20.6	0.0%	0.5%	20.7	0.0%	0.1%	276.4	0.0%	0.1%	270.2	0.0%
		0.4	-0.9%	340.2	16.1%	0.2%	307.1	15.9%	-15.7%	601.2	53.6%	-22.1%	601.1	53.2%
	3	0.6	-2.0%	593.2	41.7%	-0.5%	577.2	33.3%	-8.2%	601.0	78.1%	1.0%	599.5	72.3%
		1	16.3%	600.7	60.9%	21.1%	600.8	70.4%	4.4%	601.8	81.1%	8.0%	602.3	87.0%
		1.5	28.3%	600.9	47.9%	41.1%	601.1	56.3%	17.7%	601.5	75.5%	26.7%	601.5	81.8%
		2	27.6%	601.1	49.5%	49.1%	601.1	42.2%	19.4%	601.3	73.5%	36.4%	601.9	74.4%
Average			8.3%	461.9	31.7%	12.9%	452.4	35.7%	2.4%	551.9	55.6%	4.1%	546.7	60.2%
		0.2	-2.7%	474.5	18.5%	-11.4%	483.8	21.8%	-125.0%	602.0	85.1%	-86.3%	602.2	79.6%
		0.4	-36.2%	600.9	74.1%	-9.8%	601.1	58.9%	-54.6%	602.5	86.9%	-59.0%	602.3	87.3%
	1	0.6	-9.3%	601.1	87.7%	-9.9%	601.5	88.3%	-26.6%	602.9	96.2%	-43.0%	603.1	97.2%
		1	-9.2%	601.4	91.1%	-4.2%	601.6	92.1%	-33.2%	603.6	95.8%	-24.4%	603.0	96.5%
		1.5	5.1%	601.4	82.4%	11.6%	601.1	82.1%	-7.9%	601.8	91.3%	-1.5%	602.4	91.8%
		2	6.2%	600.9	79.6%	13.0%	601.1	79.3%	-11.6%	602.2	90.6%	1.1%	602.1	90.7%
		0.2	-5.1%	470.4	15.4%	-7.6%	487.1	18.0%	-95.6%	601.5	81.5%	-80.6%	601.3	79.5%
		0.4	-25.5%	601.1	63.3%	-20.7%	601.2	58.0%	-50.9%	602.3	87.3%	-63.5%	603.2	87.2%
2	2	0.6	-5.4%	601.4	88.0%	-1.3%	601.1	85.1%	-38.8%	603.5	96.0%	-24.6%	604.1	96.7%
		1	5.9%	601.4	90.4%	6.2%	601.2	91.1%	-41.5%	603.3	96.6%	-40.2%	604.9	97.3%
		1.5	18.1%	601.2	81.3%	27.6%	601.2	79.5%	-4.4%	601.6	91.9%	20.9%	602.2	91.2%
		2	17.0%	601.0	79.5%	30.9%	601.1	77.7%	1.2%	601.7	90.6%	25.8%	601.6	89.3%
		0.2	-11.2%	564.2	20.5%	-3.8%	480.0	20.1%	-92.4%	601.4	81.1%	-96.8%	601.6	81.7%
		0.4	-32.5%	601.0	64.7%	-32.9%	600.9	64.5%	-52.8%	602.0	90.8%	-93.8%	602.0	89.1%
	3	0.6	-6.6%	601.1	87.5%	0.2%	601.1	82.5%	-45.3%	602.6	96.7%	-22.1%	603.6	96.5%
		1	-1.9%	601.6	92.0%	7.4%	601.5	91.5%	-35.2%	602.8	96.5%	-56.0%	602.7	97.1%
		1.5	14.7%	601.1	83.1%	39.8%	601.7	72.0%	-5.5%	601.9	92.7%	17.5%	601.4	92.5%
		2	24.5%	601.0	78.2%	44.6%	601.3	67.7%	9.9%	602.1	90.5%	37.0%	601.6	88.6%
Average			-3.0%	585.4	71.3%	4.4%	581.6	68.4%	-39.5%	602.3	91.0%	-32.8%	602.5	90.5%

Table A4: Results obtained through solving the MILP with lazy constraints on instances with 8 vertices.

			$\lvert K\rvert = 1$						$\lvert K\rvert = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
1	1	0.2	0.1%	4.4	0.0%	0.1%	4.5	0.0%	0.0%	24.3	0.0%	0.0%	29.3	0.0%
		0.4	0.9%	10.0	0.0%	1.0%	10.6	0.0%	0.0%	52.1	0.0%	0.0%	55.9	0.0%
		0.6	3.0%	24.9	0.0%	3.5%	26.8	0.0%	0.8%	300.3	0.0%	0.8%	256.0	0.0%
		1	9.4%	136.2	0.0%	12.7%	209.3	0.0%	5.4%	600.5	31.2%	4.7%	600.6	36.7%
		1.5	13.1%	139.2	0.0%	19.9%	257.5	0.0%	7.1%	517.7	6.0%	7.2%	520.1	9.4%
		2	13.1%	138.5	0.0%	20.5%	263.9	0.0%	7.2%	514.7	5.7%	7.2%	586.9	14.0%
	2	0.2	0.5%	4.1	0.0%	0.5%	4.6	0.0%	0.4%	23.1	0.0%	0.4%	23.6	0.0%
		0.4	2.2%	8.6	0.0%	2.5%	8.9	0.0%	1.5%	50.3	0.0%	1.7%	44.1	0.0%
		0.6	5.3%	23.5	0.0%	6.0%	21.9	0.0%	3.9%	184.8	0.0%	4.7%	153.2	0.0%
		1	20.2%	104.9	0.0%	24.2%	116.7	0.0%	13.5%	592.1	35.2%	15.7%	501.9	17.4%
		1.5	31.9%	84.3	0.0%	42.8%	41.7	0.0%	33.0%	141.9	0.0%	36.1%	183.7	1.4%
		2	33.5%	79.4	0.0%	45.6%	77.0	0.0%	38.0%	148.5	0.0%	48.1%	161.4	0.0%
	3	0.2	0.6%	4.1	0.0%	0.6%	4.6	0.0%	0.5%	24.9	0.0%	0.5%	22.4	0.0%
		0.4	2.5%	9.3	0.0%	2.9%	10.0	0.0%	1.8%	43.9	0.0%	1.9%	40.4	0.0%
		0.6	5.6%	19.1	0.0%	6.2%	20.9	0.0%	4.1%	190.9	3.1%	5.1%	196.9	0.0%
		1	22.5%	79.7	0.0%	26.3%	127.7	0.0%	16.4%	516.0	25.5%	16.6%	510.3	24.2%
		1.5	41.1%	51.4	0.0%	49.5%	21.1	0.0%	38.0%	140.0	0.0%	42.3%	129.7	0.0%
		2	41.8%	43.5	0.0%	53.8%	21.8	0.0%	44.7%	95.0	0.0%	63.2%	69.5	0.0%
Average			13.7%	53.6	0.0%	17.7%	69.4	0.0%	12.0%	231.2	5.9%	14.2%	227.0	5.7%
2	1	0.2	0.2%	10.8	0.0%	0.3%	11.3	0.0%	0.0%	83.1	0.0%	0.0%	82.5	0.0%
		0.4	1.7%	60.5	0.0%	2.2%	50.5	0.0%	-2.5%	467.2	20.0%	-6.0%	436.0	37.6%
		0.6	3.4%	443.6	8.9%	5.2%	408.5	9.2%	1.7%	553.1	30.8%	1.5%	547.5	31.8%
		1	4.7%	602.4	41.1%	10.7%	602.3	48.7%	4.3%	602.3	84.0%	3.1%	602.9	85.8%
		1.5	13.3%	599.2	10.8%	20.2%	598.6	30.9%	5.6%	600.9	83.4%	6.1%	600.8	79.8%
		2	13.4%	480.9	8.0%	20.4%	600.6	18.8%	5.4%	601.0	78.9%	7.1%	600.8	83.8%
	2	0.2	0.7%	10.1	0.0%	0.7%	11.2	0.0%	0.5%	64.0	0.0%	0.5%	58.0	0.0%
		0.4	4.4%	39.2	0.0%	5.3%	29.6	0.0%	-2.1%	443.7	22.5%	-4.2%	347.7	24.4%
		0.6	10.4%	217.8	4.1%	11.3%	122.2	0.0%	7.1%	383.9	13.3%	-2.4%	455.6	31.3%
		1	17.5%	603.2	39.8%	26.1%	333.0	15.4%	10.4%	601.0	80.2%	14.6%	563.9	74.2%
		1.5	33.3%	474.4	6.6%	44.9%	377.0	0.0%	18.8%	601.3	83.9%	30.7%	601.7	75.4%
		2	35.0%	362.5	0.3%	47.3%	315.6	0.9%	31.8%	601.2	50.3%	45.2%	600.8	68.4%
	3	0.2	0.8%	11.6	0.0%	0.8%	11.3	0.0%	0.5%	54.1	0.0%	0.5%	49.9	0.0%
		0.4	4.7%	32.8	0.0%	5.8%	27.5	0.0%	1.6%	302.9	5.7%	-4.3%	381.7	26.7%
		0.6	11.2%	193.5	3.3%	11.4%	107.5	0.0%	6.4%	395.9	17.9%	8.4%	449.6	15.4%
		1	28.5%	300.7	13.0%	31.5%	284.4	11.8%	12.6%	554.8	64.2%	18.3%	517.8	59.8%
		1.5	43.2%	265.2	0.8%	52.7%	109.8	0.0%	31.1%	477.0	60.7%	39.7%	506.8	53.5%
		2	44.1%	280.3	0.0%	56.8%	233.0	0.0%	39.7%	516.9	57.4%	56.6%	600.4	57.1%
Average			15.0%	277.1	7.6%	19.6%	235.2	7.5%	9.6%	439.1	41.9%	12.0%	444.7	44.7%

Table A5: Results obtained through solving the MILP with lazy constraints on instances with 9 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
		0.2	0.0%	10.1	0.0%	0.0%	9.8	0.0%	0.0%	116.8	0.0%	0.0%	118.3	0.0%
		0.4	1.5%	62.7	0.0%	1.8%	59.6	0.0%	0.3%	385.2	11.7%	0.4%	390.4	11.5%
	1	0.6	3.4%	308.2	3.6%	5.4%	254.2	4.1%	1.4%	600.5	35.4%	0.7%	600.9	34.2%
		1	8.9%	600.3	28.4%	11.7%	600.5	42.2%	0.0%	601.5	60.5%	-1.8%	602.2	70.9%
		1.5	15.6%	472.3	5.3%	21.7%	593.1	15.4%	6.9%	600.6	53.0%	5.3%	600.6	70.9%
		2	15.9%	435.2	1.2%	22.7%	598.0	12.1%	8.0%	600.4	35.1%	8.1%	600.6	44.3%
		0.2	0.1%	9.5	0.0%	0.1%	9.9	0.0%	0.1%	113.5	0.0%	0.1%	124.2	0.0%
		0.4	4.5%	44.6	0.0%	4.8%	44.5	0.0%	1.8%	347.4	9.4%	1.8%	350.4	5.9%
1	2	0.6	10.5%	164.3	0.0%	11.9%	149.0	0.0%	4.1%	601.0	42.4%	3.7%	590.6	30.1%
		1	20.8%	548.2	21.7%	25.5%	554.8	22.0%	5.9%	602.2	68.5%	6.8%	603.6	76.9%
		1.5	31.0%	354.2	0.0%	42.9%	300.9	2.6%	28.3%	583.1	30.7%	35.2%	526.3	25.9%
		2	31.6%	369.2	0.0%	44.4%	306.5	0.0%	32.9%	526.2	15.9%	47.5%	515.7	13.2%
		0.2	0.1%	9.8	0.0%	0.1%	9.9	0.0%	0.1%	119.9	0.0%	0.1%	145.7	0.0%
		0.4	4.6%	39.9	0.0%	5.0%	47.2	0.0%	2.0%	376.5	4.6%	2.0%	379.8	6.7%
	3	0.6	11.7%	114.6	0.0%	13.3%	118.0	0.0%	5.8%	585.7	34.2%	5.8%	537.1	31.2%
		1	21.6%	476.3	27.0%	25.8%	512.2	35.2%	-5.2%	602.7	74.7%	8.6%	601.4	83.1%
		1.5	35.9%	293.8	0.0%	52.3%	70.9	0.0%	34.7%	517.8	36.3%	42.6%	427.5	16.0%
		2	36.2%	276.7	0.0%	53.7%	164.2	0.0%	43.0%	376.6	0.4%	55.0%	324.5	11.0%
Average			14.1%	255.0	4.8%	19.1%	244.6	7.4%	9.5%	458.8	28.5%	12.3%	446.6	29.5%
		0.2	0.1%	30.0	0.0%	0.1%	38.1	0.0%	0.1%	403.2	7.6%	0.0%	405.2	11.4%
		0.4	1.8%	421.3	12.7%	2.8%	340.5	9.4%	-25.2%	600.6	53.1%	-31.3%	570.6	51.3%
	1	0.6	-3.9%	604.4	50.2%	0.8%	603.1	46.9%	-8.9%	601.0	75.9%	-26.9%	601.5	80.6%
		1	-0.3%	608.0	63.8%	6.9%	604.5	70.7%	-0.2%	602.7	90.3%	2.3%	624.0	92.1%
		1.5	5.5%	601.6	61.4%	13.3%	601.2	73.3%	-0.8%	602.0	89.1%	1.6%	601.6	89.9%
		2	5.5%	600.6	60.1%	15.2%	600.5	65.5%	1.8%	601.2	86.1%	3.3%	600.7	87.7%
		0.2	0.5%	31.7	0.0%	0.5%	31.4	0.0%	0.1%	380.1	8.4%	0.3%	356.3	8.4%
		0.4	6.3%	242.1	3.6%	7.1%	229.0	1.9%	-38.0%	534.7	53.0%	-21.0%	562.4	44.7%
2	2	0.6	7.4%	578.1	37.4%	11.2%	567.6	27.1%	-13.9%	601.1	74.2%	-12.2%	601.3	80.5%
		1	9.8%	603.9	67.0%	23.3%	570.5	48.8%	6.9%	642.0	90.3%	11.0%	623.2	86.1%
		1.5	12.0%	601.0	73.6%	28.8%	600.9	60.2%	8.8%	601.1	89.2%	14.7%	601.9	89.5%
		2	16.7%	600.7	61.1%	36.0%	600.9	56.6%	8.4%	600.9	87.4%	20.5%	600.5	87.1%
		0.2	0.6%	35.6	0.0%	0.6%	37.2	0.0%	0.4%	398.1	7.5%	0.4%	310.7	2.9%
		0.4	6.2%	289.7	6.6%	8.0%	182.8	0.0%	-18.0%	579.1	44.3%	-22.2%	569.6	52.4%
	3	0.6	10.9%	507.9	26.3%	12.8%	485.2	23.4%	-2.7%	601.2	74.3%	-15.1%	601.7	75.3%
		1	9.3%	607.9	75.6%	29.4%	531.4	37.7%	6.6%	626.3	91.0%	11.0%	636.9	92.2%
		1.5	22.3%	602.4	64.5%	49.2%	600.7	40.8%	12.8%	601.1	89.4%	16.5%	601.6	91.0%
		2	32.0%	600.4	34.6%	53.6%	582.7	33.4%	17.5%	601.3	86.5%	31.1%	602.8	87.9%
Average			7.9%	453.7	38.8%	16.6%	433.8	33.1%	-2.5%	565.4	66.5%	-0.9%	559.6	67.3%

Table A6: Results obtained through solving the MILP with lazy constraints on instances with 10 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
				$D = 1$			$D = 2$			D=1			D=2	
s_n	α	$\mathcal{E}_r$	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap	Δ	t	Gap
1	1	0.2	0.1%	17.9	0.0%	0.1%	17.5	0.0%	0.1%	485.5	11.3%	0.1%	478.3	8.5%
		0.4	1.5%	251.6	5.7%	1.7%	246.0	5.6%	-0.8%	601.3	31.4%	-0.7%	601.5	38.0%
		0.6	1.1%	570.4	28.4%	2.6%	565.3	31.8%	-2.2%	601.2	58.4%	-2.0%	602.1	64.7%
		1	4.6%	601.0	39.9%	4.5%	601.0	55.1%	-2.0%	604.0	64.8%	0.0%	602.3	74.9%
		1.5	7.8%	600.6	33.5%	14.6%	600.5	48.3%	1.2%	601.7	65.7%	0.5%	601.4	74.4%
		2	10.7%	600.5	25.6%	15.5%	600.6	40.1%	1.1%	601.3	64.1%	4.8%	601.1	73.1%
	2	0.2	0.5%	18.0	0.0%	0.5%	16.9	0.0%	0.1%	475.6	10.3%	0.1%	489.5	10.8%
		0.4	3.3%	152.5	0.0%	3.5%	216.4	0.0%	0.1%	601.4	35.9%	-0.8%	601.9	35.4%
		0.6	5.9%	488.2	29.4%	7.2%	476.0	21.5%	-0.3%	601.7	64.9%	0.9%	601.7	65.9%
		1	13.7%	601.0	46.7%	19.3%	601.3	47.8%	6.5%	602.1	72.3%	9.8%	601.2	79.5%
		1.5	22.0%	600.5	33.9%	31.3%	588.6	36.4%	14.1%	601.7	69.6%	21.8%	601.7	75.1%
		2	22.6%	590.4	30.5%	36.0%	556.7	39.4%	16.2%	600.7	67.6%	26.3%	602.0	71.9%
	3	0.2	0.5%	20.1	0.0%	0.5%	17.6	0.0%	0.1%	489.2	10.2%	0.1%	471.7	10.8%
		0.4	3.4%	177.3	0.0%	3.6%	221.5	2.0%	-0.6%	600.9	37.3%	-0.3%	593.2	36.0%
		0.6	8.3%	481.6	29.3%	6.7%	478.2	24.5%	-1.9%	603.2	70.7%	1.1%	572.1	60.2%
		1	6.3%	601.0	57.1%	21.0%	601.7	53.3%	5.7%	602.7	78.2%	-5.2%	601.6	85.5%
		1.5	29.3%	587.9	31.4%	46.8%	519.3	28.3%	15.1%	601.6	72.5%	26.8%	601.1	75.1%
		2	31.4%	584.3	26.5%	47.7%	560.8	29.7%	23.3%	601.5	69.9%	36.3%	601.7	73.3%
Average			9.6%	419.1	23.2%	14.6%	415.9	25.8%	4.2%	582.1	53.1%	6.6%	579.2	56.3%
2	1	0.2	0.2%	87.7	0.0%	0.2%	109.3	0.0%	-0.7%	594.5	31.9%	-2.7%	599.8	30.1%
		0.4	-2.0%	600.6	32.1%	-1.7%	585.5	31.3%	-68.4%	600.7	89.1%	-87.4%	600.8	84.2%
		0.6	-11.4%	601.1	72.4%	-7.5%	601.1	69.9%	-15.2%	615.0	94.3%	-10.8%	601.3	92.4%
		1	-9.4%	600.8	83.1%	-1.0%	602.1	84.2%	-9.9%	601.4	95.3%	-11.0%	601.9	95.7%
		1.5	-0.9%	601.3	78.9%	7.6%	601.4	82.5%	-3.7%	601.7	90.8%	-0.1%	601.7	91.8%
		2	1.2%	601.0	69.6%	10.5%	601.4	76.4%	-3.4%	601.2	90.1%	3.5%	601.5	90.2%
	2	0.2	1.5%	84.8	0.0%	1.5%	85.2	0.0%	-0.3%	582.8	24.0%	-8.3%	580.1	25.8%
		0.4	1.9%	509.7	27.3%	1.6%	505.1	28.1%	-56.6%	601.3	84.3%	-50.4%	601.1	80.3%
		0.6	-2.8%	600.9	68.2%	5.1%	601.9	57.9%	-16.0%	601.5	94.4%	-2.0%	601.1	93.8%
		1	0.4%	601.8	83.6%	11.2%	601.1	81.4%	-3.6%	601.6	95.7%	2.4%	601.6	95.0%
		1.5	6.9%	601.2	79.7%	17.9%	601.2	79.9%	5.1%	601.4	91.4%	17.3%	602.0	91.2%
		2	11.6%	601.6	75.4%	23.5%	600.8	78.6%	6.5%	601.1	90.2%	20.2%	601.2	89.5%
	3	0.2	1.6%	78.1	0.0%	1.6%	88.8	0.0%	-2.7%	601.5	28.5%	-4.8%	601.6	35.0%
		0.4	2.1%	544.9	26.9%	4.6%	465.7	18.4%	-63.4%	601.2	79.9%	-56.6%	601.5	78.2%
		0.6	-3.6%	602.1	67.2%	6.4%	601.3	60.3%	-11.0%	601.4	93.2%	-13.2%	624.9	93.4%
		1	6.3%	601.0	83.4%	18.2%	601.5	69.7%	2.1%	601.8	94.6%	1.0%	601.5	94.8%
		1.5	7.5%	601.0	83.5%	31.3%	603.3	72.5%	6.6%	602.9	92.0%	20.9%	602.1	91.7%
		2	11.3%	600.8	79.5%	29.4%	602.1	77.9%	15.9%	601.3	89.9%	33.5%	603.2	88.3%
Average			1.3%	506.2	56.0%	8.9%	503.3	53.8%	-11.5%	600.8	80.5%	-7.6%	601.6	80.0%

Table A7: Results obtained through the heuristic on instances with 8 vertices.

			$\|K\| = 1$						$\|K\| = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops
		0.2	0.1%	61.6	0.4	0.1%	61.6	0.4	-0.1%	61.4	0.2	-0.1%	61.4	0.2
		0.4	0.9%	61.8	0.9	1.0%	62.1	1.1	-0.1%	61.5	0.7	-0.1%	61.5	0.7
	1	0.6	2.9%	62.2	1.4	3.5%	67.6	2.2	0.7%	61.5	1.4	0.8%	62.3	1.7
		1	9.4%	63.6	1.8	12.7%	85.3	2.9	3.9%	61.8	1.9	5.5%	63.6	2.6
		1.5	13.1%	64.2	1.9	19.9%	202.3	3.2	6.1%	62.0	2.2	7.1%	63.9	3.4
		2	13.1%	64.5	1.9	20.5%	267.3	3.0	7.2%	62.7	2.0	7.2%	77.3	3.5
		0.2	0.5%	61.7	0.4	0.5%	61.7	0.4	0.3%	61.4	0.2	0.3%	61.4	0.2
		0.4	2.2%	61.8	0.9	2.2%	62.1	1.1	1.5%	61.5	0.7	1.5%	61.6	0.7
2	2	0.6	5.3%	62.3	1.4	5.9%	75.3	2.2	3.5%	61.5	1.6	3.9%	62.0	1.8
		1	20.0%	63.6	2.3	24.2%	103.9	3.5	11.6%	61.9	2.5	14.4%	64.2	3.1
		1.5	31.7%	64.6	2.8	42.8%	238.7	4.2	30.6%	62.3	3.2	36.0%	68.3	4.2
		2	33.2%	65.0	2.9	45.6%	290.0	4.1	35.9%	62.2	3.1	47.4%	67.7	4.4
		0.2	0.6%	61.7	0.4	0.6%	61.7	0.4	0.4%	61.5	0.2	0.4%	61.4	0.2
		0.4	2.5%	61.8	0.9	2.5%	62.1	1.1	1.8%	61.5	0.7	1.8%	61.6	0.7
	3	0.6	5.5%	62.2	1.3	6.2%	75.9	2.3	3.7%	61.5	1.6	4.1%	62.0	1.8
		1	22.1%	63.6	2.4	26.3%	107.6	3.6	14.1%	62.0	2.5	16.0%	64.1	2.9
		1.5	39.7%	64.6	3.1	49.3%	259.4	4.3	35.0%	62.3	3.3	41.7%	69.0	4.2
		2	41.0%	65.0	3.2	53.8%	215.1	4.3	42.6%	62.2	3.3	56.1%	67.2	4.5
Average			13.5%	63.1	1.7	17.7%	131.1	2.5	11.0%	61.8	1.7	13.6%	64.5	2.3
		0.2	0.1%	61.9	0.6	0.1%	62.2	0.7	-0.2%	61.5	0.6	-0.2%	61.6	0.6
		0.4	1.7%	62.9	1.2	2.5%	76.2	2.1	0.3%	61.9	1.4	0.4%	62.7	1.7
	1	0.6	4.4%	65.9	1.7	5.7%	124.8	2.8	1.9%	62.6	1.9	2.5%	65.5	2.5
		1	10.0%	77.3	2.2	16.7%	182.5	3.3	5.8%	63.0	2.2	6.9%	66.9	3.1
		1.5	13.4%	88.5	2.0	20.1%	179.8	3.5	6.7%	63.0	2.1	7.1%	67.5	3.4
		2	13.4%	87.4	2.0	20.2%	186.8	3.4	7.2%	64.5	2.0	7.2%	146.8	3.8
		0.2	0.7%	61.8	0.7	0.7%	62.1	0.9	0.5%	61.5	0.6	0.5%	61.6	0.6
		0.4	4.3%	63.1	1.4	5.3%	82.1	2.3	2.3%	62.0	1.5	3.1%	62.9	1.7
2	2	0.6	10.7%	66.4	2.3	11.3%	157.7	3.5	7.5%	63.1	2.7	7.8%	67.4	2.9
		1	23.4%	87.9	3.3	27.9%	189.0	4.2	16.7%	64.0	3.4	20.2%	97.8	3.8
		1.5	33.6%	106.2	3.2	43.2%	127.0	4.3	32.0%	64.0	3.4	37.7%	80.8	4.0
		2	35.0%	117.5	3.2	45.3%	133.9	4.2	35.7%	64.1	3.3	44.2%	91.7	4.3
		0.2	0.8%	61.8	0.7	0.8%	62.1	0.9	0.5%	61.5	0.6	0.5%	61.6	0.6
		0.4	4.7%	63.3	1.4	5.7%	85.0	2.3	2.4%	62.0	1.5	3.3%	62.7	1.8
	3	0.6	11.0%	67.7	2.4	11.4%	166.4	3.4	8.8%	63.1	2.6	8.7%	67.5	3.0
		1	28.3%	92.9	3.4	32.7%	218.5	4.3	19.3%	64.1	3.3	22.7%	103.4	3.9
		1.5	43.1%	110.5	4.0	50.7%	159.5	4.5	39.0%	64.2	3.6	45.1%	82.1	4.2
		2	44.0%	115.6	3.9	53.9%	151.0	4.5	43.3%	64.4	4.0	53.8%	84.8	4.4
Average			15.7%	81.0	2.2	19.7%	133.7	3.1	12.8%	63.0	2.3	15.1%	77.5	2.8

Table A8: Results obtained through the heuristic on instances with 9 vertices.

			$\lvert K\rvert = 1$						$\lvert K\rvert = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops
	1	0.2	0.0%	61.7	0.1	0.0%	61.7	0.1	0.0%	61.6	0.1	0.0%	61.6	0.1
		0.4	1.4%	62.2	1.1	1.8%	62.7	1.3	0.2%	61.6	0.6	0.4%	61.7	0.7
		0.6	3.4%	63.6	1.5	5.4%	65.8	2.2	1.6%	61.8	1.4	1.8%	62.7	1.7
		1	10.4%	64.8	2.0	15.3%	196.0	3.5	4.2%	62.5	2.3	6.0%	64.9	3.1
		1.5	15.8%	68.2	2.0	22.5%	317.4	3.7	7.5%	62.8	2.4	8.0%	71.1	3.7
		2	16.1%	69.0	2.0	23.1%	303.2	3.7	7.9%	63.2	2.2	8.1%	123.6	3.8
1	2	0.2	0.1%	61.7	0.2	0.1%	61.7	0.2	0.1%	61.6	0.1	0.1%	61.6	0.1
		0.4	4.4%	62.0	1.1	4.7%	62.7	1.4	1.7%	61.6	0.7	1.7%	61.7	0.7
		0.6	10.5%	63.1	1.4	11.7%	64.9	2.3	5.4%	61.8	1.5	5.8%	62.6	1.8
		1	23.1%	64.6	2.8	27.7%	166.6	4.0	14.0%	62.6	2.8	17.4%	69.3	3.6
		1.5	30.7%	67.2	3.3	43.1%	345.5	4.8	28.0%	63.2	3.5	37.1%	71.6	4.7
		2	31.4%	68.8	3.5	43.3%	232.0	4.9	28.6%	63.1	3.6	45.7%	73.7	5.1
	3	0.2	0.1%	61.7	0.2	0.1%	61.7	0.2	0.1%	61.6	0.1	0.1%	61.6	0.1
		0.4	4.6%	62.1	1.1	4.9%	62.7	1.4	1.8%	61.6	0.7	1.8%	61.7	0.7
		0.6	11.7%	63.1	1.6	13.2%	65.2	2.5	6.0%	61.8	1.6	6.6%	62.7	1.7
		1	25.4%	64.8	2.9	30.0%	183.9	4.2	15.1%	62.6	2.9	19.1%	70.5	3.5
		1.5	34.6%	69.9	3.5	51.2%	335.3	4.9	31.9%	63.1	3.7	42.9%	71.6	4.7
		2	35.6%	70.4	3.7	53.3%	273.8	5.2	36.3%	63.3	3.9	53.9%	72.2	5.0
Average			14.4%	64.9	1.9	19.5%	162.4	2.8	10.6%	62.3	1.9	14.3%	69.2	2.5
	1	0.2	0.0%	61.9	0.4	0.0%	62.0	0.5	-0.2%	61.7	0.3	-0.2%	61.7	0.3
		0.4	2.6%	64.6	1.4	3.9%	73.7	2.0	0.5%	62.2	1.4	0.6%	63.3	1.7
		0.6	6.0%	76.0	2.0	8.9%	320.9	3.2	2.7%	64.0	2.1	4.1%	68.3	2.9
		1	11.8%	151.8	2.2	17.7%	220.3	3.9	5.8%	64.5	2.5	7.7%	79.7	3.6
		1.5	16.1%	246.5	2.3	21.8%	185.9	3.9	7.5%	64.8	2.3	8.1%	78.3	3.9
		2	16.3%	221.8	2.3	22.1%	205.9	3.9	7.9%	65.9	2.3	8.1%	240.1	4.0
2	2	0.2	0.5%	61.9	0.5	0.5%	62.0	0.6	0.4%	61.7	0.3	0.4%	61.7	0.3
		0.4	6.6%	64.2	1.6	7.8%	75.4	2.4	3.5%	62.1	1.7	3.7%	63.4	1.9
		0.6	15.2%	73.0	2.6	17.1%	216.3	3.8	8.3%	64.3	2.8	9.1%	69.2	3.5
		1	27.3%	141.3	3.4	32.7%	296.4	4.7	19.8%	66.1	3.9	23.7%	117.7	4.5
		1.5	33.0%	209.4	4.1	40.7%	158.5	4.7	28.4%	66.5	4.1	38.4%	107.1	4.8
		2	33.5%	203.0	4.1	40.9%	158.1	4.9	29.4%	67.4	3.7	45.0%	91.0	5.0
	3	0.2	0.6%	61.9	0.5	0.6%	62.0	0.6	0.5%	61.7	0.3	0.5%	61.7	0.3
		0.4	7.1%	64.3	1.5	8.1%	73.6	2.3	4.3%	62.3	1.7	4.5%	63.6	2.0
		0.6	16.6%	72.7	2.7	19.0%	242.7	3.7	9.2%	64.6	2.9	9.9%	77.3	3.1
		1	31.9%	164.6	3.8	37.6%	294.8	4.9	23.9%	67.2	4.0	28.1%	168.7	4.7
		1.5	39.9%	244.5	4.8	49.2%	190.1	4.9	37.1%	66.8	4.5	44.8%	96.9	4.9
		2	40.6%	228.6	4.7	50.9%	191.0	5.0	42.0%	66.6	4.6	55.0%	86.0	5.0
Average			17.0%	134.0	2.5	21.1%	171.6	3.3	12.8%	64.5	2.5	16.2%	92.0	3.1

Table A9: Results obtained through the heuristic on instances with 10 vertices.

			$\lvert K\rvert = 1$						$\lvert K\rvert = 2$					
			$D = 1$			$D = 2$			D=1			D=2		
s_n	α	$\mathcal{E}_r$	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops	Δ	t	Ops
1	1	0.2	0.1%	62.0	0.4	0.1%	62.0	0.4	-0.1%	61.7	0.2	-0.1%	61.6	0.2
		0.4	1.5%	63.0	1.1	1.7%	66.8	1.7	0.3%	61.8	1.0	0.3%	62.4	1.2
		0.6	3.3%	64.5	1.9	4.8%	123.4	2.7	1.0%	62.4	1.9	1.4%	63.5	2.4
		1	9.5%	73.6	2.3	14.4%	506.7	3.8	4.5%	63.2	2.6	5.1%	70.2	3.8
		1.5	12.8%	87.5	2.2	19.4%	328.2	4.1	6.0%	63.3	2.6	6.8%	78.5	4.0
		2	12.8%	88.6	2.3	19.7%	366.3	4.0	7.2%	65.1	2.5	7.3%	402.1	4.1
	2	0.2	0.5%	62.0	0.4	0.5%	62.0	0.4	-0.1%	61.7	0.2	-0.1%	61.6	0.2
		0.4	3.3%	63.0	1.2	3.5%	67.0	1.8	0.6%	61.9	1.0	0.7%	62.4	1.1
		0.6	9.3%	64.4	2.0	10.4%	144.9	3.0	3.6%	62.7	2.3	4.0%	64.2	2.5
		1	22.2%	76.0	3.2	28.5%	398.1	4.9	16.6%	63.7	3.4	20.4%	110.8	4.3
		1.5	27.4%	88.2	3.6	39.6%	361.8	5.2	26.3%	63.8	3.9	34.8%	112.9	5.5
		2	27.8%	86.0	3.7	40.5%	374.2	5.4	27.7%	63.9	4.0	44.6%	126.2	5.5
	3	0.2	0.5%	62.1	0.4	0.5%	62.1	0.4	-0.1%	61.7	0.2	-0.1%	61.7	0.2
		0.4	3.4%	62.9	1.2	3.6%	67.4	1.8	0.6%	61.9	1.1	0.8%	62.4	1.1
		0.6	10.4%	64.5	2.0	11.3%	143.8	3.1	4.6%	62.8	2.3	5.0%	64.2	2.6
		1	23.9%	78.8	3.3	31.7%	452.2	5.0	18.0%	63.7	3.7	22.3%	84.9	4.5
		1.5	32.8%	96.9	4.0	49.0%	392.2	5.3	29.7%	63.6	4.1	42.4%	116.5	5.4
		2	33.5%	97.0	4.0	51.2%	358.9	5.4	34.9%	63.7	4.3	59.8%	91.9	5.9
Average			13.0%	74.5	2.2	18.4%	241.0	3.2	10.1%	62.9	2.3	14.2%	97.7	3.0
2	1	0.2	0.1%	62.9	0.6	0.2%	63.4	0.8	-0.2%	61.9	0.6	-0.1%	62.1	0.6
		0.4	2.5%	69.2	1.5	3.3%	218.6	2.7	0.6%	63.4	1.7	1.0%	65.7	2.4
		0.6	4.7%	162.9	2.0	6.6%	345.3	3.6	1.8%	64.8	2.5	2.7%	74.2	3.2
		1	10.5%	212.2	2.8	15.4%	262.5	4.1	5.4%	68.1	2.7	6.8%	91.3	4.4
		1.5	12.9%	225.1	2.4	18.6%	214.0	4.2	6.9%	66.7	2.4	7.1%	101.1	4.2
		2	12.9%	228.0	2.4	18.6%	232.5	4.4	7.2%	116.7	2.5	7.3%	551.0	4.3
	2	0.2	1.5%	62.8	0.7	1.5%	63.3	0.9	0.2%	61.9	0.7	0.3%	62.1	0.8
		0.4	7.4%	67.1	2.1	7.7%	244.2	3.1	2.0%	63.5	2.5	2.4%	68.2	2.8
		0.6	14.1%	142.7	3.0	14.8%	300.5	4.3	7.2%	66.6	3.3	8.6%	127.6	4.3
		1	24.2%	187.7	4.4	30.6%	218.7	5.3	19.8%	81.0	4.4	23.3%	247.1	5.3
		1.5	29.0%	176.2	4.3	37.6%	221.4	5.0	28.0%	77.3	4.4	37.9%	130.3	5.5
		2	29.1%	177.8	4.2	38.1%	231.0	5.0	29.9%	91.7	4.6	44.5%	196.6	5.7
	3	0.2	1.6%	62.8	0.7	1.5%	63.3	0.9	0.2%	61.9	0.7	0.3%	62.1	0.7
		0.4	7.7%	69.8	2.1	7.8%	238.8	3.2	2.1%	63.5	2.4	2.6%	69.7	2.9
		0.6	15.9%	153.0	3.3	16.5%	290.2	4.4	8.4%	67.5	3.1	9.2%	153.9	4.2
		1	29.2%	220.4	4.8	35.3%	231.3	5.1	22.4%	78.3	4.7	26.1%	202.0	5.3
		1.5	37.2%	220.4	5.0	47.2%	207.7	5.1	38.8%	71.8	5.0	48.3%	143.7	5.8
		2	37.8%	249.8	4.7	48.5%	263.8	5.3	41.5%	72.6	5.0	60.0%	158.2	6.1
Average			15.4%	152.8	2.8	19.4%	217.3	3.7	12.4%	72.2	3.0	16.0%	142.6	3.8

Table A10: Results obtained through the heuristic on instances with 20 and 50 vertices.

				$K = 1$						$K = 2$					
				$D = 1$			$D = 2$			$D = 1$			$D = 2$		
$\|V\|$	s_n	α	$\mathcal{E}_r$	Obj.	t	Ops	Obj.	t	Ops	Obj.	t	Ops	Obj.	t	Ops
20	1	1	0.2	398.7	68.3	1.0	397.6	79.7	1.5	278.0	63.4	0.9	277.7	63.6	1.1
			0.4	373.0	242.5	3.5	371.0	191.7	5.1	271.6	71.0	2.7	268.7	119.6	4.6
			0.6	368.8	164.5	4.0	350.5	187.3	6.2	266.4	96.0	3.3	259.4	299.9	5.9
			1	362.6	156.3	3.7	332.9	148.6	6.4	250.5	100.2	3.8	239.1	521.1	6.9
		2	0.2	393.3	87.4	2.3	392.4	103.2	2.8	276.2	64.1	2.0	274.9	64.6	2.1
			0.4	367.3	145.2	5.3	353.3	161.1	6.8	262.7	112.6	5.4	256.2	172.8	6.6
			0.6	340.7	143.4	6.5	307.0	196.9	8.4	242.6	183.9	6.8	233.5	414.1	8.8
			1	330.1	159.9	6.4	280.7	200.6	8.9	221.6	340.3	7.6	202.7	511.6	10.8
		3	0.2	392.9	87.4	2.2	392.0	105.1	2.8	276.2	64.1	2.0	274.9	64.5	2.1
			0.4	365.5	153.8	5.5	352.0	173.1	6.9	261.5	116.3	5.6	252.5	177.9	6.4
			0.6	336.6	157.6	6.8	296.0	211.2	8.4	236.9	189.3	7.1	229.3	451.6	9.1
			1	322.0	156.4	6.7	272.9	184.4	8.8	215.4	330.2	7.9	193.1	531.9	11.1
	Average			362.6	143.6	4.5	341.5	161.9	6.1	255.0	144.3	4.6	246.8	282.8	6.3
	2	1	0.2	394.9	156.4	2.3	397.1	166.3	2.8	277.2	66.8	1.8	276.5	71.5	2.4
			0.4	402.9	147.7	3.9	392.2	135.0	5.7	269.5	114.4	3.7	265.9	282.5	5.7
			0.6	397.2	120.7	4.4	365.1	152.9	6.9	263.9	224.9	4.4	259.5	292.3	6.9
			1	381.5	135.3	4.0	350.3	139.0	6.6	254.2	214.0	4.2	248.5	244.4	7.1
		2	0.2	394.1	145.9	4.8	397.8	161.7	5.4	272.2	79.7	3.6	271.3	87.7	4.3
			0.4	378.3	139.8	6.2	357.2	140.0	7.5	256.7	208.7	6.9	251.9	303.8	8.4
			0.6	357.5	130.8	6.1	325.3	137.0	8.3	240.1	326.5	7.8	233.7	361.9	9.8
			1	346.8	142.1	6.1	307.1	143.4	8.3	223.9	250.6	7.9	209.0	286.9	10.5
		3	0.2	393.0	151.9	4.7	396.7	161.7	5.5	271.8	77.8	3.7	270.9	89.7	4.5
			0.4	373.6	140.6	6.3	346.7	157.9	7.6	252.0	215.5	6.9	246.9	313.8	8.5
			0.6	342.3	142.5	6.2	309.2	147.4	8.2	233.7	292.5	8.2	226.4	379.8	10.2
			1	336.6	131.5	6.4	291.7	159.9	8.5	213.5	303.5	8.6	197.8	292.0	10.9
	Average			374.9	140.4	5.1	353.0	150.2	6.8	252.4	197.9	5.6	246.5	250.5	7.4
50	1	1	0.2	642.8	155.8	7.0	638.0	140.5	11.5	376.9	149.3	6.5	379.0	126.4	9.8
			0.4	597.8	163.1	9.2	578.1	127.4	17.4	366.2	129.6	10.1	360.9	122.9	17.4
			0.6	585.7	163.5	8.3	561.0	122.6	15.6	359.1	139.5	9.4	349.2	145.9	17.1
			1	583.6	148.7	7.9	552.9	135.5	14.3	353.1	143.9	8.7	343.2	136.0	15.1
		2	0.2	609.9	118.7	13.0	600.2	128.5	15.1	374.6	115.2	12.5	366.8	135.4	14.3
			0.4	536.9	120.3	16.2	516.3	103.6	22.1	340.5	127.2	17.8	321.6	193.4	23.2
			0.6	546.6	128.8	15.1	509.0	111.4	20.6	332.3	143.1	17.4	314.5	179.4	23.5
			1	556.8	146.3	13.0	514.4	129.8	18.8	338.4	138.1	15.6	314.4	173.2	21.9
		3	0.2	608.7	118.8	13.1	599.1	126.8	15.1	372.0	111.4	12.6	367.4	129.8	14.4
			0.4	533.6	116.0	16.7	515.3	96.0	21.9	338.9	124.4	18.2	319.6	211.2	23.4
			0.6	544.3	126.8	15.3	505.7	113.7	20.9	330.3	165.6	18.1	312.3	185.8	24.5
			1	554.1	141.0	13.7	511.3	136.9	19.0	338.2	137.7	15.9	314.2	169.0	21.7
	Average			575.1	137.3	12.4	550.1	122.7	17.7	351.7	135.4	13.6	338.6	159.0	18.9
	2	1	0.2	646.9	265.5	9.6	651.7	165.5	15.2	386.2	163.0	8.7	389.0	158.6	12.9
			0.4	603.3	276.1	10.2	605.2	169.7	17.9	370.4	192.6	10.6	372.9	132.4	18.4
			0.6	589.2	339.4	9.2	573.6	194.0	16.1	361.5	186.7	10.0	358.6	168.4	17.3
			1	586.1	386.0	7.9	557.3	230.4	14.4	355.2	215.0	8.8	348.1	171.5	15.6
		2	0.2	597.5	138.1	16.3	591.9	121.3	18.4	370.6	131.6	16.2	364.6	117.3	18.5
			0.4	554.2	174.9	15.5	532.6	132.6	20.0	346.2	142.2	17.5	336.2	156.7	22.7
			0.6	557.0	227.0	13.7	525.0	163.5	19.5	341.6	166.8	16.0	331.8	123.3	21.8
			1	564.5	281.5	12.0	526.1	185.2	17.6	341.6	195.1	13.8	326.0	153.4	19.8
		3	0.2	586.0	132.7	16.6	579.1	123.1	18.6	362.6	132.1	16.5	363.8	120.4	18.8
			0.4	542.0	172.4	15.7	518.1	129.7	20.3	338.9	138.1	18.1	331.4	138.7	22.4
			0.6	549.2	215.7	14.2	514.0	158.0	19.7	338.8	168.4	16.2	323.8	138.1	21.9
			1	559.3	272.6	12.6	520.7	192.6	17.5	339.1	197.9	14.2	322.1	156.5	20.5
	Average			577.9	240.2	12.8	557.9	163.8	17.9	354.4	169.1	13.9	347.4	144.6	19.2