Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 12–14

THE ROBUST VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

AGOSTINHO AGRA, MARIELLE CHRISTIANSEN, ROSA FIGUEIREDO, LARS MAGNUS HVATTUM, MICHAEL POSS AND CRISTINA REQUEJO

KEYWORDS: robust linear programming; uncertainty polytope; dynamic programming; vehicle routing problem; time windows.

ABSTRACT: This paper addresses the robust vehicle routing problem with time windows. We are motivated by a problem that arises in maritime transportation where delays are frequent and should be taken into account. Our model only allows routes that are feasible for all values of the travel times in a predetermined uncertainty polytope, which yields a robust optimization problem. We propose two new formulations for the robust problem, each based on a different robust approach. The first formulation extends the well-known resource inequalities formulation by employing robust programming with recourse. We propose two techniques, which, using the structure of the problem, allow to reduce significantly the number of extreme points of the uncertainty polytope. The second formulation generalizes a path inequalities formulation to the uncertain context. The uncertainty appears implicitly in this formulation, so that we develop a new cutting plane technique for robust combinatorial optimization problems with complicated constraints. In particular, efficient separation procedures are discussed. We compare the two formulations on a test bed composed of maritime transportation instances. These results show that the solution times are similar for both formulations while being significantly faster than the solutions times of a layered formulation recently proposed for the problem.

1. Introduction

This paper demonstrates how to efficiently solve the *vehicle routing problem with time windows* (VRPTW) when travel times are uncertain. The aim is to find robust solutions, where routes are feasible for all travel times defined by a predetermined uncertainty polytope. Although the formulations developed

Received April 25, 2012
This research was carried out with financial support from the DOMinant II project, partly funded by the Research Council of Norway. Agostinho Agra, Rosa Figueiredo, and Cristina Requejo are supported by *FEDER* founds through *COMPETE*–Operational Programme Factors of Competitiveness and by Portuguese founds through the CIDMA (University of Aveiro) and FCT, within project PEst-C/MAT/UI4106/2011 with COMPETE number FCOMP-01-0124-FEDER-022690. Michael Poss is supported by CMUC and FCT (Portugal), through European program COMPETE/FEDER within project PEst-C/MAT/UI0324/2011 and under the postdoctoral scholarship SFRH/BPD/76331/2011.

in this paper are general enough to describe many types of applications, the motivation for the work comes from maritime transportation, where routing problems are known to include many types of uncertainty [11] and where travel times and service times can vary due to unforeseen events such as bad weather, mechanical breakdowns and port congestions.

Much research has been performed on vehicle routing problems, not the least due to its importance for applications in transportation, distribution and logistics [15]. Two well known classes of vehicle routing problems are the *capacitated vehicle routing problem* (CVRP) and the VRPTW. The $VRPTW$ and $CVRP$ share many common features, and path-flow formulations where integer variables represent paths in the network are very similar for both problems [23]. However, arc-flow formulations, where integer variables represent single arcs in the network, have notable differences: While it is straightforward to express the capacity constraint in the space of arc variables, time windows require either additional variables or an exponential number of inequalities [16].

We study integer programming formulations for a variant of the $VRPTW$. More specifically, we study the problem where travel times belong to an uncertainty polytope. Hence, our approach falls into the framework of robust programming, where a solution is said to be feasible only if it is feasible for all realizations of the data in a predetermined uncertainty set $\mathcal{T}$. Robust programming stems from the original work of [24] and has witnessed a continuous attention in the last decade. We refer the interested reader to the survey from [6].

Prior to our recent note [1], [27] was the only work that mentioned a *robust vehicle routing problem with time windows and uncertain travel times* ($\mathcal{T}$-$VRPTW$). However, their modeling assumption led to all travel times taking their maximum values, yielding an over-conservative model. In fact, [27] mainly focused on the robust $CVRP$ and study conditions under which robust versions of the $CVRP$ can be solved through methods similar to the ones used for the deterministic version of the $CVRP$, see also [20] for a survey on the robust $CVRP$. The literature on stochastic versions of the $VRPTW$ is scant when it comes to stochastic travel times, the only example coming from [9] who propose pre-processing techniques based on stochastic inequalities. In contrast to this, the stochastic versions of the $CVRP$ have witnessed continued attention for many years, see [10] and the references therein.

[1] present the first general approach to the robust vehicle routing problem with time windows and uncertain travel times. Travel times belong to a demand uncertainty polytope, which makes the problem harder to solve than its deterministic counterpart. The benefit of the addition in complexity is that the model from [1] is more flexible than the one from [27] and leads to less conservative robust solutions. The work presented in [1] focuses on applying the classical dualization technique for robust programming which yields a very large formulation that are hardly solved for instances with more than 20 nodes. The limited results obtained in [1] motivate us to tackle the $\mathcal{T}$-$VRPTW$ with the more sophisticated robust approaches used in this paper. Our first approach uses robust programming with recourse, while the second one substitutes robust constraints with canonical cuts.

The first of our formulations extends the classical resource inequalities formulation to the robust context. This yields a two-stage robust program, based on the framework of robust programming with recourse from [4]. Robust integer programs with arbitrary recourse are extremely hard to solve exactly so that most authors have devised approximation schemes that are computationally tractable, see for instance [4] and [14]. However, some authors have raised the possibility of obtaining exact solutions to robust programs with arbitrary recourse, see [8, 22]. When it is possible to compute all extreme points of the uncertainty set and the number of these points is limited, [22] suggest to consider all of them and to solve the resulting formulation. Very often, the numbers of extreme points are too large to be handled explicitly so that [8] propose decomposition algorithms that require solving non-convex subproblems.

For our two-stage robust program, we follow the approach of [8] and generate dynamically the extreme points of the uncertainty polytope. First, we propose two techniques that allow a significant reduction of the number of extreme points that are needed to formulate the problem. The first technique shows that the number of extreme points which we must consider is not bigger than the number of extreme points of the projection of the uncertainty polytope into the subspace corresponding to the coefficients defining any of its constraints. The second technique introduces the notion of domination among extreme points. Domination has been described already in the context of robust network design by [21]. In this paper, we extend this property to $\mathcal{T}$-$VRPTW$. Finally, we apply a column-and-row generation algorithm to generate only a subset of the non-dominated extreme points. In contrast to

the approach of [8] that solves $\mathcal{NP}$-hard subproblems, our subproblem can be solved in polynomial time by a dynamic programming algorithm.

The second of our formulations extends the path inequalities formulation from [16]. The uncertain parameters do not appear explicitly in the constraints of this formulation, so that we decompose the problem into a master problem and a subproblem. The master problem contains the deterministic constraints of the original problem plus a set of canonical cuts ensuring that the robust constraints are also satisfied, while the subproblem generates additional canonical cuts when the master problem's solution violates some of the robust constraints. We apply this approach to $VRPTW$ and $\mathcal{T}$-$VRPTW$ and our numerical results show that the resulting optimization problems for $VRPTW$ and $\mathcal{T}$-$VRPTW$ are of the same difficulty for our instances. This approach can easily be extended to other robust combinatorial optimization problems where the number of robust constraints is limited and their satisfaction is "easy" to check.

This paper is structured as follows. The next section introduces two different formulations of the $VRPTW$. Section 3 presents some key aspects of robust programming that are needed to provide finite linear programming formulations for linear problems under polyhedral uncertainty. In particular, Section 3.1 presents our new technique based on implicit representation of robust constraints. The tools from Section 3 are used in Section 4 to provide two formulations for the $\mathcal{T}$-$VRPTW$. Section 4.1 also presents a detailed study on how to reduce the number of scenarios that must be considered. Section 5 presents a numerical assessment of our formulations on a maritime transportation problem that is described in Section 5.1, and we conclude the paper in Section 6.

2. The vehicle routing problem with time windows

We first present a definition of the $VRPTW$. Considering the application to maritime transportation that we will present in Section 5, the following definition is more general than the standard $VRPTW$. By allowing travel costs and travel times to be different for each vehicle, the definition includes other related problems such as the $VRPTW$ with multiple depots. Also due to our maritime transportation application, we omit capacity constraints in the formulation. However, including capacity is easy in all of the formulations given in this paper. We are given a directed graph $G = (N, A)$, a set of vehicles K, a cost function $c : A \times K \to \mathbb{R}_+$, and a time function $t : A \times K \to$

$\mathbb{R}_+$ for traveling along the arcs of G. The graph contains special depot nodes o (origin) and d (destination) connected to all other nodes of G, and we denote by N^* the set of nodes that are not depots, $N^* := N \backslash \{o, d\}$. We are given time windows $[a_i, b_i]$ with $a_i, b_i \in \mathbb{R}_+$, for each $i \in N^*$. Because different vehicles may have access to different routes, we also introduce the subset A^k of A for each $k \in K$.

The $VRPTW$ consists of defining routes for the vehicles in K such that the union of all routes passes exactly once by each $i \in N^*$. When $|K| = 1$, the problem contains a unique vehicle and reduces to the *asymmetric traveling salesman problem with time windows* [2].

In this section, we recall two well-known formulations for the $VRPTW$, based on resource inequalities and path inequalities, respectively, and introduce a new layered formulation for the problem. These formulations suppose that all parameters are known with certainty.

2.1. Resource inequalities. We first recall a classical formulation for the $VRPTW$ based on resource inequalities. The formulation uses a set of binary flow variables x_{ij}^k which indicates weather vehicle k travels from node $i \in N$ to node $j \in N$, and a set of continuous variables y_i^k indicating the arrival time of vehicle k at node $i \in N$. In fact, since only one vehicle can serve node i, we may drop the index k and let y_i be the arrival time at node i of the vehicle that serves i. Time windows $[a_i, b_i]$ are imposed at each node $i \in N^*$, and we assume that a vehicle arriving earlier than a_i can wait until a_i at no cost. The resource inequalities model (RI) for $VRPTW$ follows.

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k x_{ij}^k \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{j \in N:(i,j) \in A^k} x_{ij}^k = 1, \qquad i \in N^*, \tag{2}$$

$$\sum_{j \in N:(j,i) \in A^k} x_{ji}^k - \sum_{j \in N:(i,j) \in A^k} x_{ij}^k = \begin{cases} -1 & i = o \\ 1 & i = d \\ 0 & otherwise \end{cases}, \quad i \in N,\ k \in K, \tag{3}$$

$$x_{ij}^k (y_i + t_{ij}^k - y_j) \leq 0, \qquad (i,j) \in A^k,\ k \in K, \tag{4}$$

$$a_i \leq y_i \leq b_i, \qquad i \in N^*, \tag{5}$$

$$x_{ij}^k \in \{0, 1\}, \qquad (i,j) \in A^k,\ k \in K.$$

The objective function (1) minimizes the cost of operating the set of vehicles. Constraints (2) ensure that all $i \in N^*$ are serviced exactly once, and constraints (3) are the flow conservation constraints for each vehicle. Constraints (4) link routes and schedules, that is, y_j must be greater than or

equal to $y_i + t_{ij}^k$ whenever vehicle k travels from i to j, while constraints (5) ensure that the time windows are respected. Constraint (4) can be linearized and replaced with constraints

$$y_i - y_j + (b_i + t_{ij}^k - a_j)x_{ij}^k \leq b_i - a_j, \qquad (i,j) \in A^k,\ k \in K. \tag{6}$$

It is shown in the next proposition that model (RI) can be improved by strengthening constraints (6). Proof of Proposition 1 is straightforward and thus, we omit it.

Proposition 1. *Constraints*

$$y_i - y_j + \sum_{k \in K:(i,j) \in A^k} \max\{b_i + t_{ij}^k - a_j, 0\} x_{ij}^k \leq b_i - a_j. \tag{7}$$

are valid for (RI)*. Moreover, any* $(x, y) \in \{0,1\}^{|A||K|} \times \mathbb{R}^{|N^*|}$ *that satisfies constraints* (7) *also satisfies constraints* (6)*.*

An important characteristic of (RI) is the presence of variables y that depend explicitly on the travel time values t_{ij}^k. Given routes described by variables x, variables y enable us to know how much time the vehicles have to wait before being able to serve each node along their route. This level of information is useful in some applications that consider costs related to waiting times such as described in [13]. However, in the application considered in this paper, only travel costs are relevant. Hence, the formulations presented in the next two sections only contain variables related to the vehicle routes. This shall have a crucial impact when applying robust models and methods to the $VRPTW$, as it will be discussed in Sections 3 and 4.

2.2. Path inequalities. A recent formulation that is based only on arc variables x has been proposed by [16] for the $VRPTW$. The formulation does not consider explicitly the satisfaction of the time windows. Instead, it forbids routes in G for which it is not possible to construct a feasible schedule. Let $\mathcal{P}^k$ be the set of infeasible paths from o to d in G, that is, the set of paths in G for which it is not possible to define arrival times y_i that satisfy constraints (4) and (5). For $p \in \mathcal{P}^k$, we denote by $|p|$ the number of arcs contained in p. The main idea of this formulation is simply to forbid

such paths.

$$
\begin{aligned}
& & \min \quad & \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k x_{ij}^k & & \\
(PI) & & \text{s.t.} \quad & (2),(3) & & \\
& & & \text{cycle-breaking inequalities,} & & (8)\\
& & & \sum_{(i,j)\in p} x_{ij}^k \leq |p|-1, & p\in\mathcal{P}^k,\ k\in K, & (9)\\
& & & x_{ij}^k \in \{0,1\}, & (i,j)\in A^k,\ k\in K. &
\end{aligned}
$$

Formulation (PI) contains one set of variables which, as before, indicates which arcs are used by each vehicle. Constraints (8) can be any set of constraints (possibly with additional variables) that forbid cycles. In our computational results we use the MTZ inequalities [18] which, essentially, uses an auxiliary set of variables similar to y in (RI) to impose an order on the nodes visited by the vehicles. Then, constraints (9) forbid infeasible paths. Formulation (PI) contains a very large number of inequalities (9), possibly exponential in the size of the problem, so that branch-and-cut algorithms must be devised to solve (PI) exactly.

3. Robust linear programming

In this work, we consider uncertain travel times that belong to a polytope $\mathcal{T}$. This makes the problem a robust program, a class of optimization problems that has witnessed a tremendous attention in the recent years. Conducting an exhaustive literature review of robust programming is beyond the scope of this paper and we redirect the interested reader to [6], among others.

The classical approach for robust programming relies on static models where the variables of the problem are not allowed to vary to account for the different values taken by the uncertain parameters. This is different from robust programming with recourse that will be introduced later in this section. Consider the following linear program in $\{0,1\}-$variables

$$
\begin{aligned}
& \min \quad c^T x & & \\
(P) \qquad & \text{s.t.} \quad Bx \leq b, & & (10)\\
& \qquad\ \ Tx \leq d, & & (11)\\
& \qquad\ \ x\in\{0,1\}^n, &
\end{aligned}
$$

with $c \in \mathbb{R}^n, b \in \mathbb{R}^r, d \in \mathbb{R}^s, T \in \mathbb{R}^{sn}$, and $B \in \mathbb{R}^{rn}$. Suppose that the problem is subject to uncertainty in the sense that matrix T belongs to a polytope $\mathcal{T} \subset \mathbb{R}^{sn}$. The robust counterpart of (P) is

$$\begin{array}{lll} & \min & c^T x \\ (\mathcal{T}\text{-}P) & \text{s.t.} & Bx \leq b, \\ & & Tx \leq d \qquad\qquad T \in \mathcal{T}, \\ & & x \in \{0,1\}^n, \end{array} \tag{12}$$

where the s linear constraints in (11) must now be satisfied for each value of $T \in \mathcal{T}$. Hence, the finite set of constraints (11) has been replaced by the infinite set of constraints (12).

The classical approach in linear robust programming under polyhedral uncertainty to handle the infinite set of constraints (12), see [5], relies on dualizing constraints (12). This results in the addition of a polynomial set of constraints and variables that depend on the definition of the uncertainty polytope $\mathcal{T}$. To be applied to the $\mathcal{T}$-$VRPTW$, this approach requires to use an extended formulation, which contains a set of constraints $Tx \leq d$ that describe the time windows in a static manner, that is, using only variables related to the routes (and not to the actual schedule). Such a formulation has been proposed in [1]. The formulation from [1] yields very poor numerical results already in the deterministic case. For this reason, we introduce alternative formulations in this paper, respectively based on the implicit representation of (12) and on robust programming with recourse.

3.1. Implicit reformulation. The method described in this section is based on the implicit representation of (12) via canonical cuts. Implicit reformulation has been used already in this paper to obtain formulation (PI) where the satisfaction of time windows is not included explicitly. Instead, (9) play this role by forbidding individual paths that do not satisfy the time windows. The idea of replacing complicating constraints by canonical cuts has been used in other contexts as well, see [12, 25] among others. According to our knowledge, this paper is the first work that applies this implicit reformulation to a robust program. In the following, we recall first the general idea for the deterministic problem (P). Then, we extend it to the robust problem $(\mathcal{T}\text{-}P)$.

Let $\overline{\mathcal{X}} \subset \{0,1\}^n$ be the set of all binary vectors that violate at least one of the constraints (11). For $x^* \in \{0,1\}^n$, we denote by $x^*(1) \subseteq \{0,\ldots,n\}$

(resp. $x^*(0) \subseteq \{0, \ldots, n\}$) the set of indices where x^* is equal to 1 (resp. 0). Hence, a vector $x^* \in \overline{\mathcal{X}}$ can be cut-off by the following canonical cut

$$\sum_{i \in x^*(1)} (1 - x_i) + \sum_{i \in x^*(0)} x_i \geq 1,$$

first mentioned by [3]. Thus, constraints (11) for x binary are equivalent to the following set of canonical cuts

$$\sum_{i \in x^*(1)} (1 - x_i) + \sum_{i \in x^*(0)} x_i \geq 1, \qquad x^* \in \overline{\mathcal{X}}. \tag{13}$$

We see that constraints (9) are an example of (13). Similarly, let $\overline{\mathcal{X}}(\mathcal{T})$ be the set of binary vectors that violate at least one of the constraints in (12). This set of constraints for x binary are equivalent to

$$\sum_{i \in x^*(1)} (1 - x_i) + \sum_{i \in x^*(0)} x_i \geq 1, \qquad x^* \in \overline{\mathcal{X}}(\mathcal{T}). \tag{14}$$

Therefore, ($\mathcal{T}$-P) can equivalently be written as

$$(\mathcal{T}\text{-}Png) \qquad \begin{array}{ll} \min & c^T x \\ \text{s.t.} & (10), (14) \\ & x \in \{0,1\}^n, \end{array}$$

which is a finite linear program in $\{0,1\}-$variables.

Let us make a couple of remarks about ($\mathcal{T}$-Png). First, canonical cuts employed in (14) are extremely weak because each of these constraints cuts off a unique binary vector. Hence, one should try to reinforce these constraints with problem-dependent valid inequalities. For instance, each path inequality in (9) cuts-off a unique path o to d. They can be improved as follows. Instead of considering a whole path from o to d, we can forbid its smallest subpath that is not feasible for the time windows. By doing so, we cut off all paths from o to d that contain the forbidden subpath. Even further, these inequalities can be lifted to obtain the tournament inequalities [2].

Second, ($\mathcal{T}$-Png) is likely to contain a very large number of constraints in (14). Therefore, a solution method that intends to solve ($\mathcal{T}$-Png) efficiently should employ a cutting plane algorithm that alternates between feasibility checks – does the current x^* belong to $\overline{\mathcal{X}}(\mathcal{T})$ – and the addition of canonical cuts to a master problem, see for instance [12, 2, 16]. An important issue in these iterative techniques is related to how to perform the feasibility check.

A binary vector x^* belongs to $\overline{\mathcal{X}}(\mathcal{T})$ if and only if there exists a $T \in \mathcal{T}$ such that $Tx > d$, which is equivalent to

$$\exists i \in \{1, \ldots, s\} \text{ s.t. } \max_{T_i \in \mathcal{T}_i} T_i x > d_i. \tag{15}$$

Hence, checking whether x^* belongs to $\overline{\mathcal{X}}(\mathcal{T})$ amounts to solve s linear programs. Solving s linear programs can be time consuming in general. Hence, it is useful to devise more efficient algorithms that make use of the particular structure of the problem under the consideration and the uncertainty polytope $\mathcal{T}$.

We show later in this paper that this check can be improved when using the budget uncertainty from [7] and even further for $\mathcal{T}$-$VRPTW$ by taking into account that vector x describes a set of paths from o to d.

3.2. Robust programming with recourse. Model ($\mathcal{T}$-P) suffers from a certain rigidity in the sense that a vector x must satisfy constraints (12) for all $T \in \mathcal{T}$ to be feasible for ($\mathcal{T}$-P). In particular, the problem variables are not allowed to adjust themselves to the values taken by the uncertain parameters. This is an important modeling restriction that may not suit many problem formulations, including the formulation (RI) from last section. Namely, adapting (RI) in the way suggested by model ($\mathcal{T}$-P) would yield an optimization problem where the arrival times would be fixed once for all travel times in the uncertainty set. Such an optimization problem is likely to be infeasible whenever $\mathcal{T}$ is not a singleton, see Example 1 from [1].

[4] have introduced a more flexible class of robust programs, where a subset of variables is allowed to adapt itself as the uncertain parameters vary in the uncertainty set $\mathcal{T}$. Applied to (P), their model essentially allows a subset of variables, which we denote by x^2, to become functions defined on $\mathcal{T}$. To keep notations simple, we suppose that these functions take only real values, $x^2 : \mathcal{T} \to \mathbb{R}^{n_2}$. Hence, $x = (x^1, x^2), c = (c^1, c^2), B = (B^1, B^2), T = (T^1, T^2)$, and x^2 is allowed to vary as T^1 takes different values in $\mathcal{T}$. For the sake of

simplicity, we also suppose that $c^2 = B^2 = 0$. The problem becomes:

$$
\begin{array}{lll}
\min & (c^1)^T x^1 & \\
\text{s.t.} & B^1 x^1 \leq b, & \\
& T^1 x^1 + T^2 x^2(T^1) \leq d, & T^1 \in \mathcal{T}, \\
& x^1 \in \{0,1\}^{n_1}, & \\
& x^2(T^1) \in \mathbb{R}^{n_2}, & T^1 \in \mathcal{T}.
\end{array}
\qquad (\mathcal{T}R\text{-}P) \qquad (16)
$$

Problem ($\mathcal{T}R$-P) is often called a robust program with recourse, which features two levels of decisions: first-stage variables x^1 must be fixed before the uncertainty is revealed, while recourse variables x^2 can react to account for the uncertainty. Notice that ($\mathcal{T}R$-P) can be extended to the case of uncertain cost c^1 by replacing the objective function with $\min z$ and adding the restrictions $z \geq (c^1)^T x^1$ to the set of uncertain constraints. Similarly, one can suppose that $B^2 \neq 0$ or that $c^2 \neq 0$. In the latter, one must add term $\max_{T \in \mathcal{T}^1} (c^2)^T x^2(T^1)$ to the objective function. However, the situation where c^2 or T^2 is uncertain is more complicated and we do not address it in the following.

Model ($\mathcal{T}R$-P) has an infinite number of variables $x^2(T^1)$ and constraints (16). However, given that all constraints present in the problem are linear, it is easy to show that we can restrict ourselves to the extreme points of $\mathcal{T}$, $\text{ext}(\mathcal{T})$, which exist in finite number since $\mathcal{T}$ is a polytope. This simple result is recalled below.

Lemma 1. *Let $\mathcal{T} \subset \mathbb{R}^{sn_1}$ be a polytope and $\text{ext}(\mathcal{T})$ be the set of its extreme points. Consider vectors $x^1 \in [0,1]^{n_1}$ and $d \in \mathbb{R}^s$. There exists $x^2 : \mathcal{T} \to \mathbb{R}^{n_2}$ such that $T^1 x^1 + T^2 x^2(T^1) \leq d, \forall T^1 \in \mathcal{T}$ if and only if there exists $x^2 : \text{ext}(\mathcal{T}) \to \mathbb{R}^{n_2}$ such that $T^1 x^1 + T^2 x^2(T^1) \leq d, \forall T^1 \in \text{ext}(\mathcal{T})$.*

Lemma 1 allows us to solve ($\mathcal{T}R$-P) through ($\text{ext}(\mathcal{T})R$-P). Though finite, ($\text{ext}(\mathcal{T})R$-P) tends to be very large because the number of extreme points of the uncertainty polytope tends to grows rapidly with the problem size. For this reason, we present in Section 4.1.1 techniques to reduce the number of extreme points.

4. The robust $VRPTW$

From this section on, we suppose that travel times t_{ij}^k are not known with precision and belong to an uncertainty set. This is because in our application

in maritime transportation, it often happens that delays occur during some of sailings due to unstable weather. We must however ensure that the routes proposed for the ships are feasible in most situations. Hence, we model the travel times with the help of an uncertainty polytope $\mathcal{T} \subset \mathbb{R}^{|A||K|}$, making the optimization problem a robust program. In the following subsections, we apply the methods described in Section 3 to the formulations for $VRPTW$ from Section 2. For each formulation, we first present the robust equivalent for a general uncertainty polytope $\mathcal{T}$. Then, we particularize the formulations to take into account the structure of the polytope $\mathcal{T}_\Gamma$ used in our numerical experiments. We suppose that each component t_{ij}^k of t lies between its mean value $\bar{t}_{ij}^k$ and its peak value $\bar{t}_{ij}^k + \hat{t}_{ij}^k$ and that, for each $k \in K$, at most Γ of them can reach their peak values simultaneously. Formally, this is defined by $\mathcal{T}_\Gamma = \times_{k\in K}\mathcal{T}_\Gamma^k$ where each $\mathcal{T}_\Gamma^k$ is such that each t_{ij}^k lies in $[\bar{t}_{ij}^k, \bar{t}_{ij}^k + \delta_{ij}^k \hat{t}_{ij}^k]$ with $0 \leq \delta_{ij}^k \leq 1$, $\sum_{ij} \delta_{ij}^k \leq \Gamma$ for some $\Gamma \in \mathbb{Z}$ with $\Gamma < |A|$. This is the budget uncertainty polytope studied by [7].

Before presenting the robust formulations, we introduce a set of constraints that has been proposed by [9] to check that time windows are satisfied without the need of additional variables. Consider a binary vector $\overline{x} \in \{0,1\}^{|A||K|}$ that describes a path p from i_0 to i_n, that is, $p = i_0, \ldots, i_n$ and $\overline{x}_{ij}^k = 1$ for each $(i,j) \in p$, such that $\overline{x}_{ij}^k = 0$ otherwise. Constraints (4) and (5) for k along p are equivalent to

$$a_{i_{l_1}} + \sum_{l=l_1,\ldots,l_2-1} t_{i_l i_{l+1}}^k \leq b_{i_{l_2}}, \qquad 0 \leq l_1 < l_2 \leq n. \tag{17}$$

Constraints (17) will be used in the next two subsections to check that a path satisfies the time windows.

4.1. Resource inequalities. Model (RI) can be naturally extended to handle uncertain polytope $\mathcal{T}$: x becomes the set of first-stage variables, while y becomes $y(t)$, a function of $t \in \mathcal{T}$. Thanks to Lemma 1, we only need to consider travel times vectors t that belong to $\text{ext}(\mathcal{T})$. Hence, the robust problem contains equations (5) and (7) written for every scenario $t \in \text{ext}(\mathcal{T})$, that is

$$a_i \leq y_i(t) \leq b_i, \qquad i \in N, t \in \text{ext}(\mathcal{T}), \tag{18}$$

and

$$y_i(t)-y_j(t)+\sum_{k\in K:(i,j)\in A^k} \max\{(b_i+t_{ij}^k-a_j),0\}x_{ij}^k \leq b_i-a_j,\ (i,j)\in A, t\in \text{ext}(\mathcal{T}). \tag{19}$$

The robust version of (RI) becomes

$$\begin{aligned} & \min && \sum_{k\in K}\sum_{(i,j)\in A^k} c_{ij}x_{ij}^k \\ (\text{ext}(\mathcal{T})\text{-}RI) \quad & \text{s.t.} && (2),(3),(18),(19) \\ & && x_{ij}^k \in \{0,1\}, \qquad (i,j)\in A^k,\ k\in K. \end{aligned}$$

To simplify notations, we assume in the rest of this section that $A^k = A$ for each $k \in K$. However, the results presented next are easy to generalize to the case where A^k can be different from A^h for any pair of distinct vehicles k and h. In what follows, we denote the elements of $\text{ext}(\mathcal{T})$ either by extreme points or by scenarios. Considering every scenario in $\text{ext}(\mathcal{T})$ is certainly prohibitive when solving reasonable size instances. In the next subsection, we focus on approaches to reduce the number of scenarios to be considered by proposing formulations that are *equivalent* to $(\text{ext}(\mathcal{T})\text{-}RI)$. Given a finite set $\mathcal{S} \subset \mathbb{R}^{|A||K|}$, we define $(\mathcal{S}\text{-}RI)$ as $(\text{ext}(\mathcal{T})\text{-}RI)$ by replacing $\text{ext}(\mathcal{T})$ with $\mathcal{S}$ in constraints (18) and (19). Hence, we want to characterize finite sets $\mathcal{S}$ with the lowest possible cardinality that satisfies the following property: a first stage solution x is feasible for $(\mathcal{S}\text{-}RI)$ if and only if it is feasible for $(\text{ext}(\mathcal{T})\text{-}RI)$. We mention that $\mathcal{S}$ may not be a subset of $\text{ext}(\mathcal{T})$.

4.1.1. *Reducing the number of scenarios.* The simplest approach tries to withdraw individual scenarios from $\text{ext}(\mathcal{T})$ by comparing them to other scenarios that are more representative in the sense explained next. Namely, we say that an extreme point $t \in \text{ext}(\mathcal{T})$ is dominated by another extreme point $\tau \in \text{ext}(\mathcal{T})$ if any solution for first stage variables x feasible for scenario τ is also feasible for t. Hence, only those extreme points that are not dominated need to be considered. In particular, there exists an easy sufficiency condition to check whether t is dominated by τ.

Proposition 2. *Let t, τ be two vectors in $\text{ext}(\mathcal{T})$. If $\tau_{ij}^k \geq t_{ij}^k$ for each $k \in K$ and $(i,j) \in A$, then t is dominated by τ.*

Proof: The result follows directly by considering the rewriting of the time windows performed in (17). ∎

In what follows we make an abuse of language and say that a vector t is dominated by τ when they satisfy the conditions of Proposition 2. The concept of domination has already been used with success in the context of robust network design, see [21, 22].

FIGURE 1. The set of scenarios $\{\bar{t}, \hat{t}\}$ *dominates* $\tilde{t}$.

(a) node i'

(b) projection of $\mathcal{T}$ in the space corresponding to arcs (j', i') and (j'', i')

In what follows, we refine the concept of domination by using the special structure of the $\mathcal{T}$-$VRPTW$, and more specifically the fact that a route can follow at most one arc that enters any node of the graph. Example 1 gives an intuitive description of our idea on a simple instance of the problem.

Example 1. *Consider an instance of the problem with* $|K| = 1$ *and let* i' *be a node with two incoming arcs, see Figure 1(a). We assume that* $\mathcal{T}$ *is a polytope whose projection in the space corresponding to arcs* (j', i') *and* (j'', i') *is the triangle depicted in Figure 1(b). Hence, none of the scenarios in* $\{\bar{t}, \hat{t}, \tilde{t}\}$ *is dominated by another scenario in* $\{\bar{t}, \hat{t}, \tilde{t}\}$*. Also, we suppose that* $\bar{t}_{ij} > \tilde{t}_{ij}$ *and* $\hat{t}_{ij} > \tilde{t}_{ij}$ *for all* $(i, j) \in A \backslash \{(j', i'), (j'', i')\}$*.*

We are going to show that a path p *feasible for* both $\bar{t}$ *and* $\hat{t}$ *is always feasible for* $\tilde{t}$*. The result follows easily from the next observation:* p *must contain at most one of the arcs* (j', i') *and* (j'', i')*, see Figure 1(a). Then, recall that* $p = i_0, \ldots, i_n$ *is feasible for the time windows if and only if constraints* (17) *are satisfied. If* $(j', i') \in p$ *(resp.* $(j'', i') \in p$*), then constraint* (17) *written for* $\bar{t}$ *(resp.* $\hat{t}$*) implies constraint* (17) *written for* $\tilde{t}$*. If* $i' \notin p$*, constraint* (17) *written for* $\tilde{t}$ *is implied by either of the two other constraints.*

Let us extend Example 1 to the general case. Consider a node $i \in N^*$. Let $\delta^+(i)$ (resp. $\delta^-(i)$) be the set of arcs leaving (resp. entering) node i. We say that an extreme point $t \in \text{ext}(\mathcal{T})$ is dominated by a subset $\mathcal{S} \subseteq \text{ext}(\mathcal{T}) \backslash \{t\}$ if there exists an arc set

$$A' = \delta^+(i) \text{ or } A' = \delta^-(i) \tag{20}$$

and a vehicle $k' \in K$ such that the following is satisfied:

$$t_{ij}^k \leq \tau_{ij}^k \qquad (i,j) \in A \backslash A',\ k \in K,\ \tau \in \mathcal{S}, \tag{21}$$

$$t_{ij}^k \leq \tau_{ij}^k \qquad (i,j) \in A',\ k \in K \backslash \{k'\},\ \tau \in \mathcal{S}, \tag{22}$$

$$t_{ij}^{k'} \leq \max_{\tau \in \mathcal{S}} \tau_{ij}^{k'} \qquad (i,j) \in A'. \tag{23}$$

Proposition 3. *Let t be a scenario in $\operatorname{ext}(\mathcal{T})$ dominated by $\mathcal{S} \subseteq \operatorname{ext}(\mathcal{T}) \backslash \{t\}$. Any solution for first stage variables x feasible for each scenario $\tau \in S$ is also feasible for t.*

Proof: Let x be any first stage solution feasible for each scenario $\tau \in \mathcal{S}$ and A' be an arc set that satisfies (20)–(23). The result follows directly by noticing that x is equal to one on at most one arc from A' and considering the rewriting of the time windows performed in (17). ∎

Any scenario that is dominated either by one scenario or by a group of scenarios may be withdrawn from the set of scenarios that must be considered, which will be used in our numerical experiments.

In what follows, we present a different approach for reducing the scenario set that combines the components of t for different vehicles. Let $\mathcal{T}^k \subset \mathbb{R}^{|A|}$ be the projection of $\mathcal{T}$ into the components corresponding to vehicle k, see Figure 2. For any finite set $\mathcal{S} \subset \mathbb{R}^{|A||K|}$, let $\mathcal{S}^k$ be the projection of $\mathcal{S}$ into the components corresponding to vehicle k. We are going to prove that we may replace $\operatorname{ext}(\mathcal{T})$ by any finite set $\mathcal{S} \subset \mathbb{R}^{|A||K|}$ such that $\operatorname{ext}(\mathcal{T}^k) = \mathcal{S}^k$ for each $k \in K$.

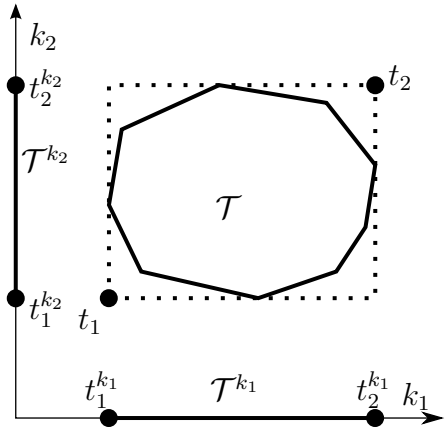
Proposition 4. *Consider $x \in \{0,1\}^{|A||K|}$ and a finite set $\mathcal{S} \subset \mathbb{R}^{|A||K|}$ such that $\operatorname{ext}(\mathcal{T}^k) = \mathcal{S}^k$ for each $k \in K$. There exists $y \in \mathbb{R}^{|N|\,|\operatorname{ext}(\mathcal{T})|}$ such that (x,y) is feasible for ($\operatorname{ext}(\mathcal{T})$-RI) if and only if there exists $\overline{y} \in \mathbb{R}^{|N||\mathcal{S}|}$ such that $(x,\overline{y})$ is feasible for ($\mathcal{S}$-RI).*

Proof: Please see Appendix A. ∎

The interest of Proposition 4 lies in the fact that $\mathcal{S}$ can be chosen in such a way that $|\mathcal{S}|$ is much smaller than $|\operatorname{ext}(\mathcal{T})|$. More precisely, it is easy to see that any $\mathcal{S}$ that satisfies the assumption of Proposition 4 must have at least $\max_{k \in K} |\operatorname{ext}(\mathcal{T}^k)|$ elements, and we show below how to construct such a set that contains exactly $\max_{k \in K} |\operatorname{ext}(\mathcal{T}^k)|$ elements.

Consider the collection of discrete sets $\{\operatorname{ext}(\mathcal{T}^1), \ldots, \operatorname{ext}(\mathcal{T}^{|K|})\}$. We examine first the case where $|\operatorname{ext}(\mathcal{T}^1)| = \ldots = |\operatorname{ext}(\mathcal{T}^{|K|})|$ and let m be the

FIGURE 2. Polytope $\mathcal{T}$ and its projections $\mathcal{T}^{k_1}$ and $\mathcal{T}^{k_2}$.



cardinality of each of these sets. Hence, $\text{ext}(\mathcal{T}^k) = \{t_1^k, \ldots, t_m^k\}$ for each $k \in K$. We construct the diagonal subset of $\times_{k\in K} \text{ext}(\mathcal{T}^k)$:

$$\text{diag}(\mathcal{T}) = \{(t_i^1, \ldots, t_i^{|K|}),\ i = 1, \ldots, m\}.$$

It is easy to see that $\text{diag}(\mathcal{T})^k = \text{ext}(\mathcal{T}^k)$ for each $k \in K$. This construction is illustrated in Example 2.

Example 2. *Consider the polytope from Figure 2. Ordering the elements in* $\mathcal{T}^{k_1} = \{t_1^{k_1}, t_2^{k_1}\}$ *and* $\mathcal{T}^{k_2} = \{t_1^{k_2}, t_2^{k_2}\}$*, we obtain that* $\text{diag}(\mathcal{T}) = \{t_1, t_2\} = \{(t_1^{k_1}, t_1^{k_2}), (t_2^{k_1}, t_2^{k_2})\}$*. In particular, the elements of* $\text{diag}(\mathcal{T})$ *are not extreme points of* $\mathcal{T}$*, that is,* $\text{diag}(\mathcal{T}) \nsubseteq \text{ext}(\mathcal{T})$*. Then, applying the domination from Proposition 2 to* $\text{diag}(\mathcal{T})$*, we obtain an even smaller uncertainty set that contains only* $\{t_2\}$*.*

Consider now that the cardinalities of $\{\text{ext}(\mathcal{T}^1), \ldots, \text{ext}(\mathcal{T}^{|K|})\}$ are different, and suppose without loss of generality that $|\,\text{ext}(\mathcal{T}^1)| \leq \ldots \leq |\,\text{ext}(\mathcal{T}^{|K|})|$. Then, we extend these sets by adding copies of their last elements so that each of the extended sets has a cardinality equal to $|\,\text{ext}(\mathcal{T}^{|K|})|$, and we define the diagonal for the extended sets.

In the next subsection, we characterize explicitly $\text{ext}(\mathcal{T})$ and $\text{diag}(\mathcal{T})$ for the budget uncertainty set used in our computational experiments.

4.1.2. *Extreme points of the budget uncertainty polytope.* Recall that $\mathcal{T}_\Gamma = \times_{k\in K}\mathcal{T}_\Gamma^k$ where each $\mathcal{T}_\Gamma^k$ is such that each t_{ij}^k lies in $[\bar{t}_{ij}^k, \bar{t}_{ij}^k + \delta_{ij}^k \hat{t}_{ij}^k]$ with $0 \leq \delta_{ij}^k \leq 1$, $\sum_{ij} \delta_{ij}^k \leq \Gamma$ for some $\Gamma \in \mathbb{Z}$ with $\Gamma < |A|$. Below we characterize

the extreme points of $\mathcal{T}_\Gamma$ by providing the following two results. Their proofs are straightforward.

Proposition 5. *t is an extreme point of $\mathcal{T}_\Gamma$ if and only if $t = \times_{k\in K} t^k$ and t^k is an extreme point of $\mathcal{T}_\Gamma^k$.*

Proposition 6. *Let $t = \times_{k\in K} t^k$. For each $k \in K$, t^k is an extreme point of $\mathcal{T}_\Gamma^k$ if and only if $t_{ij}^k = \bar{t}_{ij}^k + \delta_{ij}^k \hat{t}_{ij}^k$ and $\delta_{ij}^k \in \{0,1\}$, $\forall (i,j) \in A$.*

Proposition 6 provides a characterization of the extreme points of $\mathcal{T}_\Gamma^k$ from the values of δ_{ij}^k. Restricting ourselves to scenarios in $\text{ext}(\mathcal{T}_\Gamma)$, the parameters δ_{ij}^k can be assumed to be binary. In that case, δ_{ij}^k indicates whether there is a delay of vehicle k in arc $(i,j) \in A$ or not. Thus, parameters δ_{ij}^k permit to define the combination of the extreme points as the set of arcs where the delays occurs for each vehicle.

In the following, we apply the methods described in Section 4.1.1 to reduce the number of scenarios to consider. First, recall that we must only consider non-dominated scenarios. Applying Proposition 2 to $\mathcal{T}_\Gamma$, we obtain immediately the next result.

Proposition 7. *Let $t \in \text{ext}(\mathcal{T}_\Gamma)$. If $t = \times_{k\in K} t^k$ and $t_{ij}^{k'} = \bar{t}_{ij}^{k'} + \delta_{ij}^{k'} \hat{t}_{ij}^{k'}$ with $\sum_{(i,j)\in A} \delta_{ij}^{k'} < \Gamma$ for some $k' \in K$, then t is dominated by $\tau \in \mathcal{T}_\Gamma \backslash \{t\}$.*

Proposition 7 establishes that only scenarios where $\sum_{(i,j)\in A} \delta_{ij}^k = \Gamma$ need to be considered in (18) and (19). Those scenarios correspond to the most adverse situations in our application. In view of Proposition 7, we shall define a smaller uncertainty set $\overline{\mathcal{T}}_\Gamma = \times_{k\in K} \overline{\mathcal{T}}_\Gamma^k$ where each $\overline{\mathcal{T}}_\Gamma^k$ is such that each t_{ij}^k lies in $[\bar{t}_{ij}^k, \bar{t}_{ij}^k + \delta_{ij}^k \hat{t}_{ij}^k]$ with $0 \le \delta_{ij}^k \le 1$, $\sum_{ij} \delta_{ij}^k = \Gamma$ for some $\Gamma \in \mathbb{Z}$. In doing so, we reduce the number of extreme points from

$$|\,\text{ext}(\mathcal{T}_\Gamma)| = |\,\text{ext}(\mathcal{T}_\Gamma^1)|^{|K|} = \left[\sum_{i=0}^{\Gamma} \binom{|A|}{i}\right]^{|K|}$$

to

$$|\,\text{ext}(\overline{\mathcal{T}}_\Gamma)| = |\,\text{ext}(\overline{\mathcal{T}}_\Gamma^1)|^{|K|} = \binom{|A|}{\Gamma}^{|K|}. \tag{24}$$

Then, it is easy to apply Proposition 4 to $\overline{\mathcal{T}}_\Gamma = \times_{k\in K} \overline{\mathcal{T}}_\Gamma^k$ because all sets $\text{ext}(\overline{\mathcal{T}}_\Gamma^k)$ contain the same number of elements, so that the construction of

$\text{diag}(\overline{\mathcal{T}}_\Gamma)$ does not require to use extended sets. Namely, each element of $\text{diag}(\overline{\mathcal{T}}_\Gamma)$ can be related to a set of exactly Γ arcs that take their maximum travel time:

$$\begin{aligned}\text{diag}(\overline{\mathcal{T}}_\Gamma) \;=\; & \{(\bar{t}_{ij}^1 + \delta_{ij}\hat{t}_{ij}^1, \ldots, \bar{t}_{ij}^{|K|} + \delta_{ij}\hat{t}_{ij}^{|K|}), (i,j) \in A, \text{ s.t. } \delta_{ij} \in \{0,1\} \text{ and} \\ & \sum_{(i,j)\in A} \delta_{ij} = \Gamma\}.\end{aligned}$$

Applying Proposition 4 reduces the number of extreme points from (24) to

$$|\,\text{diag}(\overline{\mathcal{T}}_\Gamma)| = |\,\text{ext}(\overline{\mathcal{T}}_\Gamma^1)| = \binom{|A|}{\Gamma}.$$

Finally, using the more general domination concept from Proposition 3, we can reduce the number of elements of each $\text{ext}(\overline{\mathcal{T}}_\Gamma^k)$ that must be considered to construct $\text{diag}(\overline{\mathcal{T}}_\Gamma)$. Namely, we can withdraw from $\text{ext}(\overline{\mathcal{T}}_\Gamma^k)$, and therefore from $\text{diag}(\overline{\mathcal{T}}_\Gamma)$, all vectors where the delay occurs on two arcs that enter or leave the same node. Hence the number of scenarios that we must consider is a number comprised between $\binom{|V|}{\Gamma}$ and $\binom{|A|}{\Gamma}$ that depends on the topology of G. In the rest of this paper, we use the following abuse of language. We denote by ($\mathcal{T}$-RI) the formulation ($\text{diag}(\overline{\mathcal{T}}_\Gamma)$-$RI$) from which dominated scenarios have been withdrawn by using Proposition 3.

4.1.3. *Column-and-row generation.* For $\Gamma = 1$, the reduction techniques from the previous section enable us to solve our maritime transportation instances in limited time. However, when $\Gamma > 1$, the reduced numbers of scenarios are still very large. Hence, we implement a column-and-row generation algorithm to solve ($\mathcal{T}$-RI) by generating the required scenarios on the flow. First, we choose arbitrarily a non-dominated scenario t^0 from $\text{diag}(\overline{\mathcal{T}}_\Gamma)$ and solve the resulting problem ($\mathcal{T}^0$-RI) where $\mathcal{T}^0 := \{t^0\}$. Then, we check whether the optimal solution to ($\mathcal{T}^0$-RI) satisfies the time windows for each non-dominated $t \in \text{diag}(\overline{\mathcal{T}}_\Gamma)$, which can be performed in polynomial time (see Section 4.2.2). If the solution violates the time windows for some $t^1 \in \text{diag}(\overline{\mathcal{T}}_\Gamma)$, we define $\mathcal{T}^1 := \{t^0, t^1\}$ and repeat the procedure with ($\mathcal{T}^1$-RI). This approach ends whenever the solution satisfies the time windows for each $t \in \text{diag}(\overline{\mathcal{T}}_\Gamma)$.

4.2. Path inequalities. In this section, we first explain how to modify (PI) to handle uncertain travel times in a general uncertainty polytope $\mathcal{T}$. Then,

we show that the efficiency of the separation procedure can be improved significantly whenever we consider the budget uncertainty polytope $\overline{\mathcal{T}}_\Gamma$.

4.2.1. *General uncertainty polytope.* Let $\mathcal{P}^k_{\mathcal{T}}$ be the set of non-feasible paths in G from o to d for the uncertainty polytope $\mathcal{T}$, that is, the set of paths in G for which it is not possible to define arrival times y_i that satisfy (18) and (19). The robust version of (PI) is as follows:

$$
\begin{aligned}
& \min && \sum_{k\in K}\sum_{(i,j)\in A^k} c^k_{ij}x^k_{ij} \\
(\mathcal{T}\text{-}PI) \quad & \text{s.t.} && (2),(3),(8) \\
& && \sum_{(i,j)\in p} x^k_{ij} \le |p|-1, \quad p\in\mathcal{P}^k_{\mathcal{T}},\ k\in K, && (25)\\
& && x^k_{ij}\in\{0,1\}, \qquad\qquad (i,j)\in A^k,\ k\in K,
\end{aligned}
$$

where the uncertainty polytope appears implicitly in the description of $\mathcal{P}^k_{\mathcal{T}}$.

In what follows, we study a cutting plane algorithm where constraints (25) are generated iteratively by solving an associated separation problem [19]. Of course, an efficient separation method is essential to the success of the cutting plane algorithm. Hence, we address below how to separate constraints (25). Consider a path $p=(o=i_0,\ldots,i_{n+1}=d)$ described by a binary vector $\overline{x}^k$ for vehicle $k\in K$. We show below that we can find whether $p\in\mathcal{P}^k_{\mathcal{T}}$ in pseudo-polynomial time. Recall that the time windows along p are satisfied if and only if constraints (17) are satisfied. Therefore, $p\in\mathcal{P}^k_{\mathcal{T}}$ if there exist $0\le l_1<l_2\le n$ such that

$$
a_{i_{l_1}} + \max_{t\in\mathcal{T}^k} \sum_{l=l_1,\ldots,l_2-1} t_{i_l i_{l+1}} > b_{i_{l_2}},
$$

that is, one of the inequalities in (17) is violated for some $t\in\mathcal{T}^k$. Hence, the question whether $p\in\mathcal{P}^k_{\mathcal{T}}$ amounts to solve at most $n(n-1)/2$ linear programs.

Proposition 8. *Let p be a path in G from o to d for vehicle k. The question whether $p\in\mathcal{P}^k_{\mathcal{T}}$ can be answered in pseudo-polynomial time.*

4.2.2. *Budget uncertainty polytope.* Whenever each $\mathcal{T}^k$ has a particular structure, it may be used to devise more efficient algorithms than solving $O(n^2)$ linear programs. Consider again the non-dominated budget uncertainty set

$\overline{\mathcal{T}}_\Gamma = \times_{k\in K}\overline{\mathcal{T}}^k_\Gamma$ defined in the first paragraph of Section 4 and consider the general robust constraints

$$T^k x \leq d^k, \qquad k \in K, \mathcal{T}^k \in \overline{\mathcal{T}}^k_\Gamma \tag{26}$$

where Γ is integer.

Proposition 9. *Consider a robust program under the uncertainty set $\overline{\mathcal{T}}_\Gamma$ and a vector $x \in \mathbb{R}^n$. Then, checking whether x satisfies the robust inequalities* (26) *can be done in polynomial time, more specifically, by applying a sorting algorithm $|K|$ times.*

Proof: Please see Appendix B.1. ■

In the proposition below, we refine Proposition 9 for the $\mathcal{T}$-$VRPTW$ by using the fact that we separate path inequalities, whose structure is defined on paths from o to d.

Proposition 10. *Let $p = (o = i_0, \ldots, i_{n+1} = d)$ be a path in G from o to d for vehicle $k \in K$. The question whether $p \in \mathcal{P}^k_{\overline{\mathcal{T}}^k_\Gamma}$ can be answered in $(n - \Gamma' + 1)\Gamma'$ steps where $\Gamma' = \min(\Gamma, n)$.*

Proof: Please see Appendix B.2. ■

One observes that if $\Gamma' = \Gamma$, the separation problem for constraints (25) is solved in $O(n\Gamma)$. On the other hand, if $\Gamma' = n$, the problem is solved in $O(n)$.

Separating path inequalities (25) when x is fractional is more complicated because the arcs on which x^k takes positive values do not define a single path from o to d. However, as explained by [16], we can use the fact that there is only a polynomial number of paths for which the associated path inequality (25) is violated. Moreover, since paths inequalities are weak, works addressing $VRPTW$ or the asymmetric traveling salesman problem with time windows rather use a lifted version of the path inequalities called tournament inequalities [2].

In this paper, we also separate tournament inequalities rather then paths inequalities whether x is fractional or not. The only difference between our separation heuristic and the one from [16] is that we need to apply the dynamic programming procedure from Proposition 10 to check whether the time windows are satisfied along a candidate path. Also, in the case where x^k is binary, that is, x^k defines a unique path p from o to d, the tournament

inequality defined for p is generated as soon as x^k violates the path inequality (25) for p.

5. Computational experiments

In this section, we present a numerical assessment of the two formulations introduced in this paper as well as the extended formulation from [1] on a maritime transportation problem. Section 5.1 motivates and explain the real-world application. The instances composing our test bed are then presented in Section 5.2 while the numerical results are discussed in Section 5.3.

5.1. Application to the ship routing and scheduling problem. Maritime transportation is an area that gives rise to a wide variety of routing problems, and [11] give a thorough introduction to many of the important issues. In the following we will consider industrial shipping, where a company is using its own fleet to transport its own cargoes. In this setting the goal will be to minimize the total transportation costs, while making sure that all cargoes are transported.

For some shipping segments, it is a natural restriction that a ship can only carry at most one cargo at any time. This is the case for some type of bulk transportation where the ship is always loaded to its capacity or where different cargoes cannot be mixed. Then, one does not need to explicitly model both the pickup and the delivery. Instead, one can use each node to represent both the pickup and the subsequent delivery. Such a problem is already expressed through the models presented in this paper: an arc $(i, j) \in A^k$ represents that a ship k starts in the pickup port of cargo i, moves to the delivery port of cargo i and then sails to the pickup port of cargo j. The cost and time to perform these two legs can vary by ship, and are denoted c_{ij}^k and t_{ij}^k respectively. Since a ship always sails directly from a pickup port to the corresponding delivery port, it makes sense to include time windows for the pickup port only, and y_i will correspond to the time when a ship starts service of cargo i at the pickup port of that cargo. In maritime transportation there is no central depot and ships operate continuously. Hence, the ships may start at different positions (usually in the port where their previous delivery was made), and they may end at any position when completing their route. Thus, o represents the actual origin of a ship, and d is an artificial node representing that a ship has completed its schedule.

Travel times are highly stochastic in maritime transportation. A study by [17] reported on probability distributions for sailing times between selected ports in Europe, and showed that sailing times may vary significantly. Since the satisfaction of the end consumer usually requires on time deliveries throughout the supply chain, avoiding unnecessary delays in transportation has an economic consequence. In the case of port congestion, shipping companies will usually receive demurrages from the ports, but if the congestion leads to delays that propagate through the route of the ship, the shipping company may in turn end up paying penalties to many customers due to late deliveries. It is therefore essential to make robust schedules that are able to absorb some delays, which is exactly the purpose of the models presented in this paper.

5.2. Details of the instances. This section describes how the instances have been created for the maritime ship routing and scheduling problem described above. A random instance generator is used, but where the instances are made as realistic as possible. The instance generator takes as input the number of ships, the number of cargoes to generate and a distance matrix. The distance matrix used here contains 56 ports from around the world, with actual sailing distances between each pair of ports.

Two non-overlapping subsets of ports are selected as pickup ports and delivery ports respectively, to represent the structure of a company operating within deep sea industrial shipping. Cargo requests are generated between two ports based on a simple inventory model for the delivery port. Time windows are associated with each cargo based on when the request would be generated and an acceptable time before the delivery should be made.

Vessel attributes are generated so that the fleet is typically heterogeneous. That is, ships have different capacities, speeds and cost structures. The capacity is relevant in that some cargoes may be too big to be handled by smaller ships (if so, the smaller ship cannot service the cargo). In addition, some ports may be inaccessible by larger ships due to draft limits and port capacity (if so, the larger ship cannot service the cargo). In the instances generated, ship speeds vary between 13 and 20 knots, giving sailing times of more than one month between distant ports when using the slowest ships.

The instance generator also specifies the possible delay in sailing time for each arc in the network. This delay is calculated based on the time normally required to perform the transportation represented by the arc. The delay

also depends on the specific pickup port and delivery port involved, where some ports are associated with more delay than others. Such a structure is reasonable since bad weather affects the schedule more severely in some areas. Since the planning horizon is long, there is a significant risk of a ship being delayed at some point during its route, but the probability of experiencing longer travel times for all legs would be small. Hence it makes sense to make routes that can handle some delays, with Γ equal to some small number.

The computational testing contains instances with 20, 30, 40, and 50 different cargoes. For each number of cargoes, we consider four values of Γ: 0 (deterministic case), 1 (low uncertainty), $3 + (|N| - 20)/10$ (middle uncertainty), and $5 + (|N| - 20)/5$ (high uncertainty). For each number of cargoes, we also consider three number of ships: 1, $3 + (|N| - 20)/10$, and $5 + (|N| - 20)/5$. Finally, we generate five instances for each combination of values for the number of cargoes and number of ships.

5.3. Numerical results. All models and algorithms have been coded using the modeling language Xpress Mosel 3.2.3 and solved by Xpress Optimizer 22.01.09 [26]. A time limit of 1800 seconds has been set for each instance. They were run on a computer equipped with a processor Intel Core i5 at 2.53 GHz and 4 GB of RAM memory. The objectives of this section are (i) assessing the computational cost of solving the robust models as compared to their deterministic counterparts, and (ii) comparing formulations ($\mathcal{T}$-RI) and ($\mathcal{T}$-PI) as well as the layered formulation ($\mathcal{T}$-LF) described in [1].

Next we illustrate the reduction techniques described in Section 4.2 on an instance with 20 nodes and 3 ships. With no reduction at all, the numbers of extreme points of $\mathcal{T}_\Gamma$ for this instance are equal to $1.70\ 10^7$ and $2.85\ 10^{14}$ for Γ equal to 1 and 2, respectively. Using the diagonal space from Proposition 4, these numbers are reduced to $2.57\ 10^2$ and $6.62\ 10^4$, respectively. Then, using the dominations from Proposition 2 and Proposition 3, the number of extreme points for $\Gamma = 2$ is further reduced to $2.96\ 10^4$.

In view of these very large numbers of extreme points, we cannot expect to solve the complete formulation of ($\mathcal{T}$-RI) in once when $\Gamma > 1$. Hence, we solve ($\mathcal{T}$-RI) by the column-and-row generation algorithm presented in Section 4.1.3 and report the results of that algorithm in the following.In Table 1, we present the average number of extreme points generated to solve ($\mathcal{T}$-RI) for each number of cargoes and uncertainty level. We see that these numbers are very small compared to the total number of reduced extreme points.

Then, Table **??** reports the average numbers of cuts generated by ($\mathcal{T}$-PI) for each number of cargoes and uncertainty level. We see that ($\mathcal{T}$-PI) generates significantly more cuts than ($\mathcal{T}$-RI) generates extreme points. This can be explained by the fact that the cuts generated by ($\mathcal{T}$-PI) are tournament inequalities, which, as well as ensure that the time windows are respected, also increases the linear programming relaxation. Hence, their violation is checked in every node in the branch-and-cut tree solving ($\mathcal{T}$-PI). In opposition to this, the extreme points generated by ($\mathcal{T}$-RI) only enforce the satisfaction of the time windows. In addition, their necessity is checked only after an optimal integer solution has been found for the previous set of extreme points.

TABLE 1. Average numbers of extreme points generated by ($\mathcal{T}$-RI).

	Uncertainty level (Γ)		
$\|N\|$	low	mid	high
20	2.93	7.2	7.67
30	3.1	9.33	9.13
40	6.67	20.7	21.7
50	7.93	22.8	23.2

TABLE 2. Average numbers of cuts generated by ($\mathcal{T}$-PI).

	Uncertainty level (Γ)			
$\|N\|$	det	low	mid	high
20	120	251	346	762
30	1210	313	867	795
40	25097	9501	17997	17870
50	17919	10364	24547	25072

Average solution times are presented in Tables 3 and 4 for each group of 5 instances. Rows entitled "av" compute the average of the three rows above them. Table 3 provides average solution times for instances with 20 nodes for the three formulations. Notice that solutions times for ($\mathcal{T}$-LF) assume that the instances have already been pre-processed by computing longest paths [1]. We see that ($\mathcal{T}$-RI) and ($\mathcal{T}$-PI) are about two orders of magnitude faster than ($\mathcal{T}$-LF). Then, while ($\mathcal{T}$-RI) is faster than ($\mathcal{T}$-PI) for the deterministic instances ($\Gamma = 0$), it is slower than ($\mathcal{T}$-PI) for the instances

where $\Gamma > 0$. Table 4 presents average solution times for the larger instances for formulations ($\mathcal{T}$-RI) and ($\mathcal{T}$-PI). The numbers of unsolved instances within the 1800 seconds are given in parentheses and their values have been set to 1800 seconds when computing the averages. We see from Table 4 that the performance of ($\mathcal{T}$-RI) and ($\mathcal{T}$-PI) are comparable, although ($\mathcal{T}$-RI) seems to be more efficient for the larger instances. The results for both approaches present, however, important differences. The solution times for ($\mathcal{T}$-RI) are highly impacted by the value of Γ. Deterministic instances are always solved faster than robust instances. Moreover, the number of extreme points of the uncertainty sets also influences the solution times since instances with uncertainty sets defined by few extreme points (low) are solved faster than instances with uncertainty sets defined by larger number of extreme points (mid and high). In opposition to this, the presence of uncertainty does not seem to influence the solution times of ($\mathcal{T}$-PI).

TABLE 3. Average solution times in seconds for the three formulations for instances with 20 nodes.

	($\mathcal{T}$-LF)				($\mathcal{T}$-RI)				($\mathcal{T}$-PI)			
Γ / $\lvert K\rvert$	det	low	mid	high	det	low	mid	high	det	low	mid	high
1	21	244	255	162	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
3	39.9	289	330	242	0.1	2.5	15.2	15.5	1.2	1.7	2.2	3.8
5	13.6	53	406	177	0.1	0.2	1.6	1.3	0.8	0.4	0.5	0.6
av	24.8	195	330	194	0.1	0.9	5.6	5.6	0.7	0.7	0.9	1.4

6. Conclusion

This research addresses the vehicle routing problem with time windows and travel times that belong to an uncertainty polytope $\mathcal{T}$. We present two new formulations for the problem that are based on resource inequalities (($\mathcal{T}$-RI)) and path inequalities (($\mathcal{T}$-PI)), respectively, and extend well-known formulations for the deterministic version of the problem.

Each formulation uses different robust optimization tools to handle the uncertainty. We propose for ($\mathcal{T}$-PI) a new method to handle the uncertainty implicitly with the help of canonical cuts, which does not increase the complexity of the formulation itself. Instead, this approach sends the additional complexity to the separation routine. Then, formulation ($\mathcal{T}$-RI)

TABLE 4. Average solution times in seconds for ($\mathcal{T}$-RI) and ($\mathcal{T}$-PI) for larger instances.

		($\mathcal{T}$-RI)				($\mathcal{T}$-PI)			
	Γ	det	low	mid	high	det	low	mid	high
$\|N\|$	$\|K\|$								
	1	0.1	0.1	0.1	0.1	1	1.2	4.8	2.9
30	4	0.6	1.1	5.3	4.1	2.9	1.6	1.7	2.1
	7	3.3	8.2	58.1	66.1	27.3	7.6	16.6	17
	av	1.6	3.3	21.2	23.4	10.4	3.5	7.7	7.3
	1	0.1	0.1	0.3	1.4	1.5	2.2	5.1	24
40	5	11.3	160	640 (2)	617 (2)	391 (1)	30	249	227
	9	364 (1)	368 (1)	477 (1)	444 (1)	452 (1)	391 (1)	429 (1)	427 (1)
	av	125	176	372	354	281	141	228	226
	1	0.1	0.3	1.2	2.8	6.7	7.3	28.2	55.3
50	6	13.8	31.1	537 (1)	485 (1)	534 (1)	216	805 (2)	779 (2)
	11	109	748 (2)	1070 (3)	962 (3)	1020 (3)	773 (2)	1160 (3)	1150 (3)
	av	41	260	536	483	520	332	664	661

relies on robust programming with recourse and we present domination rules that significantly reduce the number of extreme points needed to define the uncertainty polytope. We propose efficient solution algorithms for both formulations: ($\mathcal{T}$-PI) is solved by a branch-and-cut algorithm while ($\mathcal{T}$-RI) is solved by a column-and-row generation algorithm in line of the approach proposed by [8].

We present computational results performed on a set of instances that model a maritime transportation problem using the budget uncertainty polytope studied in [7]. The performances of ($\mathcal{T}$-PI) and ($\mathcal{T}$-RI) are comparable for our instances. In addition, the results show that ($\mathcal{T}$-PI) is almost as easy to solve as its deterministic counterpart. We think that this is a very interesting result since it can be generalized to other robust combinatorial optimization problems. Hence, a side contribution of this work is the introduction of an alternative approach to the dualization technique habitually used for static robust programming.

Appendix A. Proof of Proposition 4

Proposition 4 1 (Proposition 4.). *Consider $x \in \{0,1\}^{|A||K|}$ and a finite set $\mathcal{S} \subset \mathbb{R}^{|A||K|}$ such that $\text{ext}(\mathcal{T}^k) = \mathcal{S}^k$ for each $k \in K$. There exists $y \in \mathbb{R}^{|N||\text{ext}(\mathcal{T})|}$ such that (x, y) is feasible for ($\text{ext}(\mathcal{T})$-RI) if and only if there exists $\overline{y} \in \mathbb{R}^{|N||\mathcal{S}|}$ such that $(x, \overline{y})$ is feasible for ($\mathcal{S}$-RI).*

Proof: Consider first the following simple property whose proof is straightforward.

Lemma 2. *Let $\mathcal{T}$ be a polytope in $\mathbb{R}^{|K|}$. The following holds:*

(1) $\text{ext}(\mathcal{T}^k) \subseteq \text{ext}(\mathcal{T})^k$,
(2) $\text{ext}(\mathcal{T})^k \subseteq \text{conv}(\text{ext}(\mathcal{T}^k))$.

Sufficiency: Let $(x, y) \in \{0,1\}^{|A||K|} \times \mathbb{R}^{|A||\text{ext}(\mathcal{T})|}$ be feasible for ($\text{ext}(\mathcal{T})$-RI). We construct next $\overline{y} \in \mathbb{R}^{|A||\mathcal{S}|}$ such that $(x, \overline{y})$ is feasible for ($\mathcal{S}$-RI). First, we introduce the following notation. Given $\tau \in \mathcal{S}$ and $k \in K$, we define

$$t(\tau, k) = \{t \in \text{ext}(\mathcal{T}) \text{ s.t. } t^k = \tau^k\}. \tag{27}$$

Lemma 2.1 implies that $t(\tau, k)$ is non-empty for all $\tau \in \mathcal{S}$ and $k \in K$. Then, notice that constraints (2) and (3) force x to describe a set of $|K|$ routes that partition the nodes of the graph, $N = N^1 \cup \ldots \cup N^{|K|}$. This enables us to define $\overline{y} \in \mathbb{R}^{|A||\mathcal{S}|}$ as follows. For each $k \in K$ and $\tau \in \mathcal{S}$, we choose arbitrarily $t \in t(\tau, k)$ and set $\overline{y}_i(\tau) = y_i(t)$ for each $i \in \mathcal{N}$. One can easily verify that $(x, \overline{y})$ is feasible for ($\mathcal{S}$-RI), that is, $(x, \overline{y})$ satisfies (18) and (19) where $\text{ext}(\mathcal{T})$ is replaced by $\mathcal{S}$.
Necessity: We cannot extend directly (27) to this situation because $\text{ext}(\mathcal{T})^k \backslash \mathcal{S}^k$ can be non-empty. For each $t \in \text{ext}(\mathcal{T})$, we define $K(t) = \{k \in K \text{ s.t. } t^k \in \mathcal{S}\}$. The pendant of (27) is defined as $\tau(t, k) = \{\tau \in \mathcal{S} \text{ s.t. } \tau^k = t^k\}$, for each $t \in \text{ext}(\mathcal{T})$ and $k \in K(t)$. Then, we extend $\tau(t, k)$ to the other couples (t, k) by using Lemma 2.2. Namely, we define

$$\lambda(t, k) = \{\lambda \in [0,1]^{|\mathcal{S}^k|} \text{ s.t. } \sum_{j=1}^{|\mathcal{S}^k|} \lambda_j = 1 \text{ and } \sum_{j=1}^{|\mathcal{S}^k|} \lambda_j \tau_j^k = t^k\},$$

for each $t \in \text{ext}(\mathcal{T})$ and $k \in K \backslash K(t)$.

We can now set up $y \in \mathbb{R}^{|A||\text{ext}(\mathcal{T})|}$ as follows. For each $t \in \text{ext}(\mathcal{T})$ and $k \in K(t)$, we choose arbitrarily $\tau \in \tau(t, k)$ and set $y_i(t) = \overline{y}_i(\tau)$ for each $i \in \mathcal{N}$. Then, for each $t \in \text{ext}(\mathcal{T})$ and $k \in K(t)$, we choose arbitrarily $\lambda \in \lambda(t, k)$ and set $y_i(t) = \sum_{j=1}^{|\mathcal{S}^k|} \lambda_j \overline{y}_i(\tau_j)$ for each $i \in \mathcal{N}$. One can easily check that (x, y) satisfies (18) and (19). ∎

Appendix B. Dynamic programming approaches

B.1. Proof of Proposition 9.

Proposition 4 2 (Proposition 9.)**.** *Consider a robust program under the uncertainty set $\overline{\mathcal{T}}_\Gamma$ and a vector $x \in \mathbb{R}^n$. Then, checking whether x satisfies the robust inequalities* (26) *can be done in polynomial time, more specifically, by applying a sorting algorithm $|K|$ times.*

Proof: The left-hand side of each equation k in (26) can be rewritten for $\overline{\mathcal{T}}^k_\Gamma$ as

$$\max_{\delta^k \in \{0,1\}^n, \sum \delta^k = \Gamma} \sum_{i=1}^{n} (\overline{T}^k_i + \delta^k_i \hat{T}^k_i) x_i = \sum_{i=1}^{n} \overline{T}^k_i x_i + \max_{\delta^k \in \{0,1\}^n, \sum \delta^k = \Gamma} \sum_{i=1}^{n} \delta^k_i \hat{T}^k_i x_i. \quad (28)$$

The maximum in the right-hand side of (28) can be obtained by using a sorting algorithm that returns the Γ highest values among the elements $\hat{T}^k_i x_i$, $i = 1, \ldots, n$. ■

B.2. Proof of Proposition 10.

Proposition 4 3 (Proposition 10.)**.** *Let $p = (o = i_0, \ldots, i_{n+1} = d)$ be a path in G from o to d for vehicle $k \in K$. The question whether $p \in \mathcal{P}^k_{\overline{\mathcal{T}}^k_\Gamma}$ can be answered in $(n - \Gamma' + 1)\Gamma'$ steps where $\Gamma' = \min(\Gamma, n)$.*

Proof: Let $\alpha(i_j)$ be the earliest arrival time at node $i_j \in p$ when the travel times are deterministic, which is formally defined by

$$\alpha(i_j) = \max(a_{i_j}, \alpha(i_{j-1}) + t_{i_{j-1} i_j}).$$

In that case, the question whether $p \in \mathcal{P}^k$ would be answered by checking that

$$\alpha(i_j) \leq b_{i_j} \qquad 1 \leq j \leq n,$$

which can be done in $O(n)$.

Let $\alpha(i_j, \gamma)$ be the earliest arrival time at i_j when at most γ arcs are using their maximum time in subpath $i_0, \ldots, i_j$. The robust version of the earliest arrival at i_j becomes the following recursive function

$$\alpha(i_j, \gamma) = \begin{cases} \alpha(i_0, \gamma) = a_{i_0} & 0 \leq \gamma \leq \Gamma' \\ \alpha(i_j, 0) = \max(a_{i_j}, \alpha(i_{j-1}, 0) + \overline{t}_{i_{j-1} i_j}) & 1 \leq j \leq n \\ \alpha(i_j, \gamma) = \max(a_{i_j}, \alpha(i_{j-1}, \gamma - 1) + \overline{t}_{i_{j-1} i_j} & \\ \qquad + \hat{t}_{i_{j-1} i_j}, \alpha(i_{j-1}, \gamma) + \overline{t}_{i_{j-1} i_j}) & 1 \leq \gamma \leq j \\ \alpha(i_j, \gamma) = -\infty & j < \gamma \end{cases}$$

The question whether $p \in \mathcal{P}^k_{\overline{\mathcal{T}}^k_\Gamma}$ is answered by checking if

$$\alpha(i_{n+1}, \Gamma') \leq b_{i_{n+1}},$$

which can be done in $(n - \Gamma' + 1)\Gamma'$ steps. ■

References

1. A. Agra, M. Christiansen, R. Figueiredo, L. Magnus Hvattum, M. Poss, and C. Requejo, *Layered formulation for the robust vehicle routing problem with time windows*, Accepted for the LNCS proceedings of the 2nd International Symposium on Combinatorial Optimization, 2012.
2. N. Ascheuer, M. Fischetti, and M. Grötschel, *A polyhedral study of the asymmetric traveling salesman problem with time windows*, Networks **36** (2000), no. 2, 69–79.
3. E. Balas and R Jeroslow, *Canonical cuts on the unit hypercube*, SIAM Journal on Applied Mathematics **23** (1972), 61–79.
4. A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, *Adjustable robust solutions of uncertain linear programs*, Mathematical Programming **99** (2004), no. 2, 351–376.
5. A. Ben-Tal and A Nemirovski, *Robust solutions of uncertain linear programs*, Operations Research Letters **25** (1999), 1–13.
6. D. Bertsimas, D.B. Brown, and C. Caramanis, *Theory and applications of robust optimization*, SIAM Review **53** (2011), 464–501.
7. D. Bertsimas and M. Sim, *The price of robustness*, Operations Research **52** (2004), 35–53.
8. D. Bienstock and N. Özbay, *Computing robust basestock levels*, Discrete Optimization **5** (2008), no. 2, 389–414.
9. M. Chardy and O. Klopfenstein, *Handling uncertainties in vehicle routing problems through data preprocessing*, Transportation Research Part E: Logistics and Transportation Review **48** (2012), no. 3, 667 – 683.
10. C. H. Christiansen and J. Lysgaard, *A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands*, Operations Research Letters **35** (2007), no. 6, 773 – 781.
11. M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen, *Handbooks in operations research and management science*, vol. 14, ch. Maritime Transportation, pp. 189–284, 2007.
12. G. Codato and M. Fischetti, *Combinatorial Benders' cuts for mixed-integer linear programming*, Operations Ressearch **54** (2006), no. 4, 756–766.
13. G. Desaulniers and D. Villeneuve, *The shortest path problem with time windows and linear waiting costs*, Transportation Science **34** (2000), no. 3, 312–319.
14. J. Goh and M. Sim, *Distributionally robust optimization and its tractable approximations*, Operations Ressearch **58** (2010), 902–917.
15. B. Golden, S. Raghavan, and E. A. Wasil, *The vehicle routing problem : latest advances and new challenges*, Operations research/Computer science interfaces series, 43, Springer, 2008.
16. B. Kallehauge, N. Boland, and O. B. G. Madsen, *Path inequalities for the vehicle routing problem with time windows*, Networks **49** (2007), no. 4, 273–293.
17. W. Kauczynski, *Study of the reliability of the ship transportation*, In Proceeding of the International Conference on Ship and Marine Research, 1994.
18. C. E. Miller, A. W. Tucker, and R. A. Zemlin, *Integer programming formulation of traveling salesman problems*, Journal of the ACM **7** (1960), 326–329.

19. G.L. Nemhauser and L.A Wolsey, *Integer programming and combinatorial optimization*, Wiley-Interscience, 1988.
20. F. Ordónez, *Robust vehicle routing*, TUTORIALS in Operations Research (2010), 153–178.
21. G. Oriolo, *Domination Between Traffic Matrices*, Mathematics of Operations Research **33** (2008), no. 1, 91–96.
22. M. Poss and C. Raack, *Affine recourse for the robust network design problem: between static and dynamic routing*, ZIB Report 11-03, Zuse Institute Berlin, Germany, February 2011.
23. G. Righini and M. Salani, *New dynamic programming algorithms for the resource constrained elementary shortest path problem*, Networks **51** (2008), no. 3, 155 – 170.
24. A. L. Soyster, *Convex programming with set-inclusive constraints and applications to inexact linear programming*, Operations Research **21** (1973), 1154–1157.
25. A. Subramanian, E. Uchoa, A. A. Pessoa, and L. Satoru Ochi, *Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery*, Operations Research Letters **39** (2011), no. 5, 338 – 341.
26. FICO Xpress Optimization Suite, *Xpress-Optimizer reference manual*, Tech. Report Release 22.01, 2011.
27. I. Sungur, F. Ordónez, and M. Dessouky, *A robust optimization approach for the capacitated vehicle routing*, IIE Transactions **40** (2008), no. 5, 509–523.

Agostinho Agra
CIDMA, Department of Mathematics, University of Aveiro, 3810-193 Aveiro, Portugal
E-mail address: aagra@ua.pt

Marielle Christiansen
Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
E-mail address: marielle.christiansen@iot.ntnu.no

Rosa Figueiredo
CIDMA, Department of Mathematics, University of Aveiro, 3810-193 Aveiro, Portugal
E-mail address: rosa.figueiredo@ua.pt

Lars Magnus Hvattum
Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
E-mail address: lars.m.hvattum@iot.ntnu.no

Michael Poss
CMUC, Department of Mathematics, University of Coimbra, 3001-454 Coimbra, Portugal,, GOM, Department of Computer Science, Faculté des Sciences, Université Libre de Bruxelles, Brussels, Belgium
E-mail address: mjposs@gmail.com

Cristina Requejo
CIDMA, Department of Mathematics, University of Aveiro, 3810-193 Aveiro, Portugal
E-mail address: crequejo@ua.pt