Online Appendix: The robust time-dependent vehicle routing problem with time windows and budget uncertainty

Igor Malheiros,[a] Michael Poss,[a] Anand Subramanian,[b] Vitor Nesello[c]
[a]LIRMM, Université de Montpellier, CNRS, Montpellier, France; [b]Departamento de Computação Científica, Universidade Federal da Paraíba, João Pessoa, Brazil; [c]Atoptima, Bordeaux, France

EC.1. Incorporating service times

We assume that $\bar{B}_{ij}$ and $\hat{B}_{ij}$ are computed without considering the service time. Let τ_i be the service time in vertex $i \in \mathcal{V}$. We modify the breakpoints of $\bar{B}_{ij}$ and $\hat{B}_{ij}$ to incorporate the service time as follows:

$$\begin{aligned}
\bar{\mathrm{s}}_{ij}^{p} &\leftarrow \bar{\mathrm{s}}_{ij}^{p} - \tau_i && (i,j) \in \mathcal{A}, p \in [P] \\
\bar{\mathrm{t}}_{ij}^{p} &\leftarrow \bar{\mathrm{t}}_{ij}^{p} + \tau_i && (i,j) \in \mathcal{A}, p \in [P] \\
\hat{\mathrm{s}}_{ij}^{p} &\leftarrow \hat{\mathrm{s}}_{ij}^{p} - \tau_i && (i,j) \in \mathcal{A}, p \in [P].
\end{aligned}$$

Note that the values of $\hat{\mathrm{t}}_{ij}^{p}$ are not modified, since the deviations account only for the travel time.

EC.2. Formulating the problem without FIFO assumption

We discuss next why formulation (1a)–(1i) is not valid when Assumption 1 is not satisfied. In this case, it may be beneficial to wait extra time at a given node i to benefit from a steep decrease in travel time, resulting in an earlier arrival time at node j. Such an extra waiting time can be inferred from variables $\boldsymbol{y}$, in case

$$y_j(\boldsymbol{\delta}) > \max\left\{e_j, y_i(\boldsymbol{\delta}) + t_{ij}(y_i(\boldsymbol{\delta}), \boldsymbol{\delta})\right\}, \tag{EC.1}$$

for some arc (i,j) for which $x_{ij} = 1$. Hence, two models arise in this setting: one forbidding this extra waiting time (as in Section 5), and one allowing it.

We see that formulation (1a)–(1i) is not valid for the no-waiting model, since the formulation may lead to an optimal solution satisfying (EC.1). Nevertheless, it is not hard to see that this situation can be prevented by adding the constraints

$$y_j(\boldsymbol{\delta}) \leq \max\left\{e_j, y_i(\boldsymbol{\delta}) + t_{ij}(y_i(\boldsymbol{\delta}), \boldsymbol{\delta})\right\}, \tag{EC.2}$$

which can be linearized using additional binary variables and big-M coefficients. The resulting formulation can be leveraged with techniques similar to those presented in Section 6 to obtain compact and non-compact formulations for the problem.

Formulation (1a)–(1i) is not valid for the model with extra waiting either because each variable $y_i(\boldsymbol{\delta})$ depends on the full vector $\boldsymbol{\delta}$, meaning it has perfect information of the scenario when deciding if extra waiting is beneficial (see also **?** for a discussion on robust optimization with perfect information). In reality, however, such a variable y_i should depend only on the components of $\boldsymbol{\delta}$ that appear prior to node i in the path. This issue appears to be particularly challenging to model, since it involves a multi-stage problem in which the order that the uncertain parameters are revealed depends on the decision variables $\boldsymbol{x}$. Alternatively, one can come up with a formulation based on no-good cuts as in Section 6.3. Nevertheless, one would still need to verify the feasibility of a given path π. We already proved in Section 5 that this problem is $\mathcal{NP}$-complete in the case of no-waiting. The feasibility problem does not seem easier when extra waiting is allowed, in which case (2) is replaced with

$$a_i(\boldsymbol{\delta}) = \begin{cases} e_1 & \text{if } i = 1, \\ \max\left\{e_i, \min\limits_{a_{i-1}(\boldsymbol{\delta}) \le s_i \le l_i} s_i + t_{i-1}(s_i, \boldsymbol{\delta})\right\} & \text{otherwise.} \end{cases} \tag{EC.3}$$

We observe that the possible values for this extra waiting at some node i, represented by s_i in (EC.3), typically depends on the knickpoints of the travel time functions after i in the path. If $\boldsymbol{\delta}$ were known, these values might possibly be computed using back-propagation from the end of the path (e.g. **?**). However, these knickpoints depend on the value of $\boldsymbol{\delta}$ after i, possibly leading to an exponential number of different back-propagated values.

EC.3. Illustrative numerical example

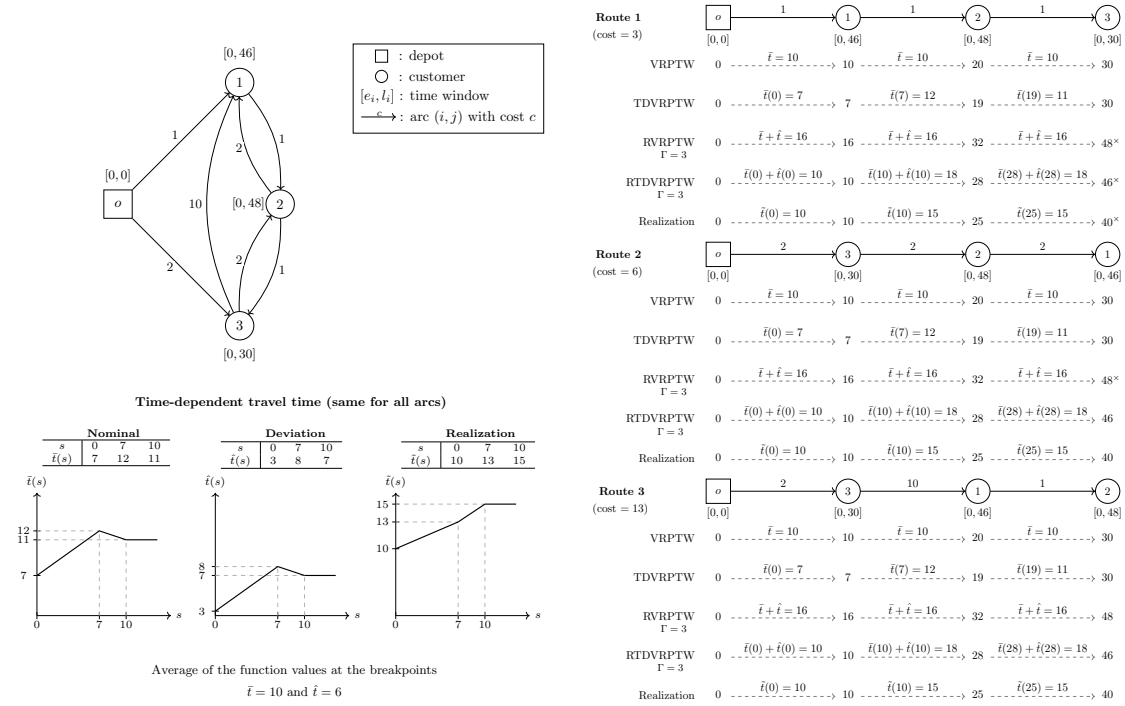
Figure EC.1 presents an illustrative comparison of four models: VRPTW, TDVRPTW, RVRPTW, and RTDVRPTW, where the two robust variants assume $\Gamma = 3$. The corresponding instance is shown in Figure EC.1a. The depot o is depicted as a square, while

the destination depot is omitted for simplicity, and customers 1 to 3 are represented as circles, with their respective time windows $[e_i, l_i]$ indicated next to each node. Arc costs are displayed alongside each arc, and travel times are assumed to be identical across all arcs.

The nominal travel time $\bar{t}(s)$ is modeled as a time-dependent function with breakpoints $(0, 7)$, $(7, 12)$, and $(10, 11)$, while the deviation function $\hat{t}(s)$ is defined by $(0, 3)$, $(7, 8)$, and $(10, 7)$. The realized travel time $\tilde{t}(s)$ follows $(0, 10)$, $(7, 13)$, and $(10, 15)$, satisfying $\bar{t}(s) \leq \tilde{t}(s) \leq \bar{t}(s) + \hat{t}(s)$. For models without time dependence, these functions are approximated by averaging their values at the breakpoints, yielding $\bar{t} = 10$ and $\hat{t} = 6$.

Figure EC.1b presents the three possible routes, together with the corresponding arrival times at each vertex for each model and for the realized travel times. Arrival times that violate the corresponding time window are marked with the superscript "$\times$". The travel-time transition costs are indicated above the dashed arrows in each case.

For the first (least-cost) route, both VRPTW and TDVRPTW find it feasible and therefore select it as optimal. However, models that account for uncertainty, as well as the realized scenario, indicate that this route is infeasible due to a time-window violation at customer 3. In the second route, VRPTW and TDVRPTW again identify it as a feasible (but suboptimal) route, while RTDVRPTW and the realized scenario also consider it feasible and optimal. In contrast, RVRPTW classifies this route as infeasible due to a violation at customer 1. The third (most expensive) route is considered feasible by all models and is therefore optimal for the RVRPTW. Overall, this example shows that models ignoring uncertainty may accept infeasible solutions, whereas RVRPTW can be overly conservative by rejecting feasible ones. In contrast, RTDVRPTW provides a more accurate representation, correctly identifying the optimal solution consistently with the realized travel times. This is due to the fact that RVRPTW considers worst-case travel times across the arc's entire planning horizon, as opposed to RTDVRPTW, which considers worst-case travel times separately for each time period.



(a) Instance (b) Routing solutions

Figure EC.1 Illustrative numerical example (the destination depot is omitted for simplicity).

EC.4. Reachability Inequalities

The separation problem for the reachability inequalities relies on computing various shortest paths among the customers. Transposing this idea to the RTDVRPTW, leads to solving the robust time-dependent shortest path problem

$$\min_{\pi \in \Pi^{SP}} \max_{\boldsymbol{\delta} \in \Delta_\Gamma} \sum_{(i,j) \in \mathcal{A}} \left[\bar{t}_{ij}(a_i(\pi, \boldsymbol{\delta})) + \delta_{ij} \hat{t}_{ij}(a_i(\pi, \boldsymbol{\delta})) \right] x_{ij}, \tag{EC.4}$$

where Π^{SP} denotes the feasibility set of the shortest path problem defined on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, and $a_i(\pi, \boldsymbol{\delta})$ denotes the arrival time at vertex i for scenario $\boldsymbol{\delta} \in \Delta_\Gamma$ and path $\pi \in \Pi^{SP}$, extending (2) by making the dependency on the path explicit. Unfortunately, problem (EC.4) appears to be much harder than its nominal counterpart, making the reachability inequalities difficult to use in the robust setting.

THEOREM EC.1. *Problem* (EC.4) *is $\mathcal{NP}$-hard in the strong sense. Furthermore, there is no polynomial-time algorithm that approximates* (EC.4) *within a factor n^κ, for some constant $\kappa > 0$, unless $\mathcal{P} = \mathcal{NP}$.*

Proof Let us consider an instance of the robust path with deadlines which, given the weighted graph $G = (V, A, \hat{r})$, and deadlines $\bar{b}_i$ for each $i \in V$, asks whether there exists a path $\pi = (o = v_1, \ldots, v_k = d)$ between o and d in G such that, for each $h = 2, \ldots, k$:

$$\max_{\delta \in \Delta_\Gamma} \sum_{\ell=1}^{h-1} \delta_{v_\ell v_{\ell+1}} \hat{r}_{v_\ell v_{\ell+1}} \leq \bar{b}_{v_h}. \tag{EC.5}$$

The robust path with deadlines is shown to be strongly $\mathcal{NP}$-hard in **?**.

The reduction considers $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ defined by $\mathcal{V} = V$ and $\mathcal{A} = A$, and $K = n^\kappa \cdot R$, with $R = \sum_{(i,j)\in A} \hat{r}_{ij}$. Then, for each $(i, j) \in \mathcal{A}$, we define $\bar{t}_{ij}(s) = 0$ for each $s \in [0, \bar{b}_j]$, and $\bar{t}_{ij}(s) = K$ for each $s > \bar{b}_j$, while $\hat{t}_{ij}(s) = \hat{r}_{ij}$ for each $s \geq 0$. We see that a path π in $\mathcal{G} = G$ avoids the high cost value K if and only if π satisfies all deadline constraints (EC.5). Hence, if π is a yes certificate to the robust path with deadlines, then π is also a yes certificate to (EC.4) with a cost less than R. Otherwise, if any path π in G violates at least one constraint (EC.5), any such path π must take at least one arc with cost $K = n^\kappa \cdot R$, so the optimal solution cost of (EC.4) is not less than $n^\kappa \cdot R$. □

EC.5. ILS parameter tuning

In these experiments, we aim to calibrate the parameters I_{max} and ρ of the primal heuristic described in Subsection 7.3. We follow a strategy similar to that of Malheiros et al. (2021), performing multiple runs across the candidate parameter configurations and identifying the non-dominated ones using a Pareto chart. We varied $I_{max} \in \{5, 10, 15, 20\}$ and $\rho \in \{1, 2, 3, 4\}$ on 30% of the RTDVRPTW Solomon instances (Section 8.1). For each instance and each of the 16 parameter configurations (I_{max}, ρ), we performed five runs of the ILS. For every configuration, we computed the gap to the best solution found across all runs in these experiments and the CPU time (in seconds), both averaged over the five runs.

The results are presented in Figure EC.2. Black circles denote Pareto-dominated configurations, and white circles denote non-dominated ones. One can observe that several

configurations are non-dominated. We adopt $I_{max} = 10$ and $\rho = 3$ (configuration 7), as it yields an average gap of around 0.3% while requiring only about half a second of runtime, offering a promising balance between the two metrics.

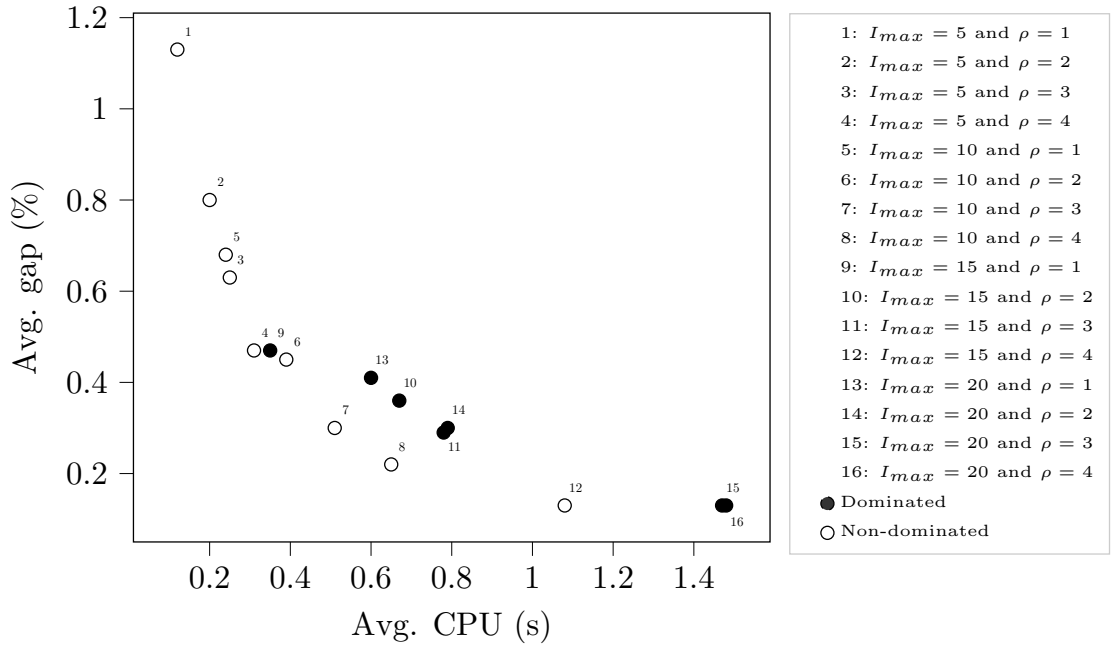


Figure EC.2 ILS parameter tuning: average gap versus average CPU time

EC.6. Piecewise linear representation

Our results require a breakpoint and closed-form description of the time-dependent travel time functions, and therefore it is necessary to have an alternative to the algorithm introduced by Ichoua, Gendreau, and Potvin (2003). A similar case was studied by Vidal et al. (2021), in which the authors proposed an algorithm capable of computing the segments of the piecewise linear functions of the arrival times. Their algorithm uses the functions $\Phi_{ij}(s)$ and $\Phi_{ij}^{-1}(s)$, where the former represents the arrival time at j departing from i at time s and the latter represents the departure time from i to arrive at j at time s. In our case,

we aim for the descriptions of the travel time function. Thus, we next present an adaptation of the algorithm proposed by Vidal et al. (2021) based on the following observation.

OBSERVATION EC.1. *Let* $\tilde{B}_{ij} = \{(\tilde{\mathfrak{s}}_{ij}^1, \tilde{\mathfrak{a}}_{ij}^1), \ldots, (\tilde{\mathfrak{s}}_{ij}^P, \tilde{\mathfrak{a}}_{ij}^P)\}$ *be the breakpoints to produce the closed-form description of the time-dependent piecewise linear arrival time function. Moreover, let* $\bar{B}_{ij} = \{(\bar{\mathfrak{s}}_{ij}^1, \bar{\mathfrak{t}}_{ij}^1), \ldots, (\bar{\mathfrak{s}}_{ij}^{P'}, \bar{\mathfrak{t}}_{ij}^{P'})\}$ *be the breakpoints to produce the closed-form description of the nominal time-dependent piecewise linear travel time function. Then* $P = P'$ *and* $\tilde{\mathfrak{s}}_{ij}^p = \bar{\mathfrak{s}}_{ij}^p$ *for* $p \in [P]$.

The key idea is to maintain the same values for $\bar{\mathfrak{s}}_{ij}^p$ and use $\Phi_{ij}(s) - s$ or $s - \Phi_{ij}^{-1}(s)$ to find the $\bar{\mathfrak{t}}_{ij}^p$ values. Let $[S_h, S_{h+1}]$, for $h \in [Z]$ be the planning horizon partitioned in $Z-1$ intervals, where $S_1 = 0$ and $S_Z = H$. The complete procedure to compute $\bar{B}_{ij}$ for an arc $(i, j) \in \mathcal{A}$ is presented in Algorithm 1.

Algorithm 1 Algorithm computes $\bar{B}_{ij}$ given an arc (i, j)

function TRAVEL_TIME_BREAKPOINTS((i, j))
Input: Arc (i, j)
Output: Set of breakpoints $\bar{B}_{ij}$
$\bar{B}_{ij} \leftarrow \emptyset$ ▷ Initialize breakpoint set
$r \leftarrow \max\{h \mid S_h \le \Phi_{ij}^{-1}(H)\}$ ▷ Last index within inverse mapping domain
$l \leftarrow \min\{h \mid S_h \ge \Phi_{ij}(0)\}$ ▷ First index within forward mapping domain
$\bar{B}_{ij} \leftarrow \bar{B}_{ij} \cup (0, \Phi_{ij}(0))$ ▷ Initial breakpoint: travel time at earliest departure
for $h \in \{1, \ldots, r\}$ **do**
  $\bar{B}_{ij} \leftarrow \bar{B}_{ij} \cup (S_h, \Phi_{ij}(S_h) - S_h)$ ▷ Compute travel time via arrival function: $\Phi_{ij}(s) - s$
for $h \in \{l, \ldots, Z\}$ **do**
  $\bar{B}_{ij} \leftarrow \bar{B}_{ij} \cup (\Phi_{ij}^{-1}(S_h), S_h - \Phi_{ij}^{-1}(S_h))$ ▷ Compute travel time via inverse function: $s - \Phi_{ij}^{-1}(s)$
$\bar{B}_{ij} \leftarrow \bar{B}_{ij} \cup (\Phi_{ij}^{-1}(H), H - \Phi_{ij}^{-1}(H))$ ▷ Final breakpoint: travel time at latest arrival
return $\bar{B}_{ij}$ ▷ Return complete set of breakpoints

EC.7. Heuristic runtime analysis for the simulations

This section presents the heuristic runtime analysis for the simulations involving the RVRPTW and RTDVRPTW models under different uncertainty sets using the Paris delivery instances introduced in Section 8.1. Figure EC.3 illustrates the results obtained from the optimization phase in the simulation experiments. Each run requires between 5 and

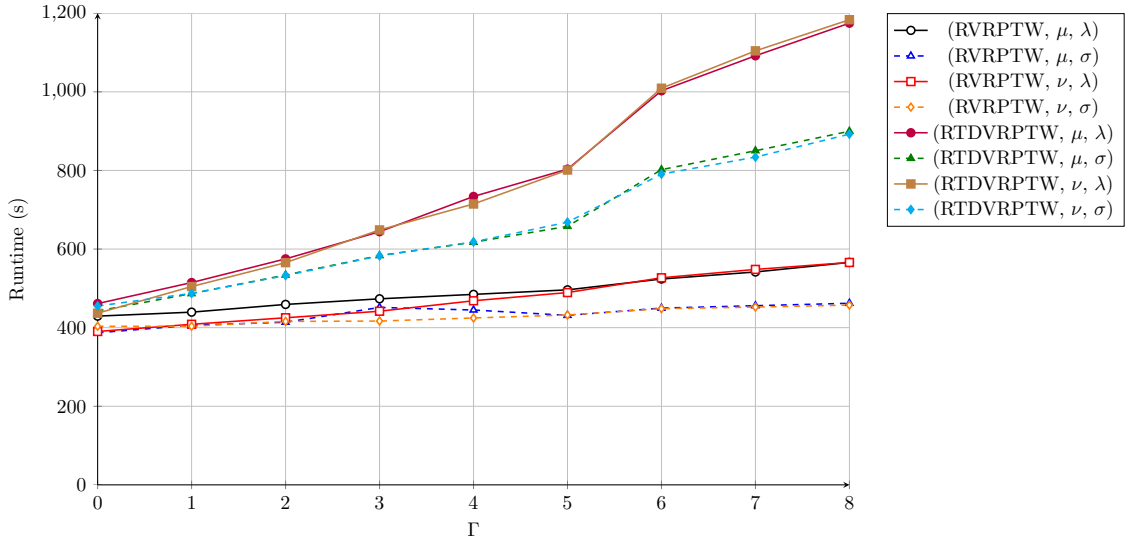


Figure EC.3 Runtime analysis for each model and uncertainty set

20 minutes to complete, which is acceptable since the planning is conducted offline, prior to execution. As expected from theoretical complexity results, runtime increases approximately linearly with the parameter Γ. Furthermore, the RVRPTW consistently outperforms the RTDVRPTW in terms of runtime for the same Γ values. In particular, for $\Gamma > 2$, the RVRPTW requires less CPU time under any uncertainty set than the RTDVRPTW does under any of them, suggesting that RVRPTW may be more suitable in applications that require faster response.

Another notable pattern is that the λ uncertainty sets tend to be more computationally expensive than the σ uncertainty sets, especially for higher values of Γ. This may be due

to the fact that λ represents a more conservative, extreme deviation, which increases the solution space. In contrast, using σ leads to less extreme scenario construction, reaching fewer infeasible solutions.