CIRRELT

CIRRELT-2024-07

A Mathheuristic Approach for the Vehicle Routing Problem with Queuing Considerations

Ala-Eddine Yahiaoui
Mikael Rönnqvist
Jean-François Audy

March 2024

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

A Mathheuristic Approach for the Vehicle Routing Problem with Queuing Considerations

Ala-Eddine Yahiaoui[1,2,*], Mikael Rönnqvist[1,2], Jean-François Audy[1,3]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and FORAC Research Consortium, Université Laval
2. Department of Mechanical Engineering, Université Laval
3. Département de management, Université du Québec à Trois-Rivières

Abstract. Queuing in vehicle routing problems happens when a given node requires to be visited by several vehicles, whereas only a limited number of vehicles can perform the service simultaneously. Hence, some vehicles must wait until the node is available. We present in this paper a mathheuristic approach to solve the problem. This approach incorporates two phases. The first phase executes a rolling horizon heuristic multiple times to generate an initial set of solutions. Those generated solutions are used to initialize a pool of routes. In the second phase, a column generation based procedure is used to generate new routes. We implemented a set partitioning model that allocates pre-determined slots of time to service operations of vehicles. We proposed fast pricing heuristics to generate new routes with negative reduced costs. The new routes are generated based on existing ones, keeping the same physical description but the starting times of service operations are modified to better fit the queuing aspects. Performance evaluation has been conducted using instances derived from data provided by forest companies. Experiments proved the effectiveness of the proposed approach, by recording low route duration and achieving almost zero queuing times compared to the initial pool of solutions.

Keywords: Vehicle routing problem, queuing, mathheuristic, column generation, time slots, timber transportation

Acknowledgements. The authors would like to thank the FORAC research consortium (Université Laval) for funding this research.

---

* Corresponding author: ala-eddine.yahiaoui.1@ulaval.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2024

1 Introduction

Queuing in Vehicle Routing Problems (VRP) appears when a given node requires to be visited several times during the day, and where only a limited number of vehicles can perform their services at a given time and a given site, mainly due to limited resources on-site. Transportation of logs from harvest areas to mills in forestry is a relevant application of such VRP variant, where a set of loads must be transported between each pair of harvest areas and mills using a fleet of trucks. To load and unload logs, specific equipment called loaders are necessary. When the number of trucks that arrives during the same time and at the same location exceeds the available number of loaders, queuing times are incurred for some trucks. A good coordination between trucks arrivals and availability of loaders is necessary to avoid bottlenecks and to minimize queuing times. We call this problem as the Timber Transport Vehicle Routing Problem with Queuing considerations (TTVRPQ). In addition to pick-up and delivery and queuing attributes, the TTVRPQ is also characterized by additional attributes such as multiple visits and multi-depots.

Several papers have been proposed to solve the variant of VRP similar to TTVRP generalization in the forestry context. Malladi and Sowlati (2017) [11] provided a detailed literature on log-truck scheduling problems and their applications in timber and biomass transportation, whereas Audy et al. (2022) [1] presented a detailed survey on the characteristics and attributes of TTVRP and their solution methods, along with their deployment in decision support systems. Weintraub et al. (1996) [15] proposed a fast heuristic embedded in a computerized system called ASICAM. This heuristic uses several rules and a rolling horizon-based simulation to construct truck routes for a whole day-horizon, while reducing queuing times and congestion levels at harvest areas and mills. ASICAM is widely adopted by forest companies in several countries, mainly due to its efficiency. However, since ASICAM is a rolling horizon heuristic, early in the planning horizon the routes may be very efficient with low queuing, but towards the end of the planning horizon the quality can decrease considerably with increased queuing. Moreover, it lacks a back-track mechanism, which would allow the heuristic to further improve the obtained solutions.

Several approaches have been proposed to solve the TTVRP based on Integer Programming (IP). Bordòn et al. (2020) [3] proposed another approach to minimize duration and queuing times at harvest areas and mills. This approach consists in splitting the planning horizon into equal time intervals, called time slots, corresponding to loading/unloading times. Slot-based MIP formulation facilitates the modeling of the queuing operations by simply solving the problem of time slots allocation at harvest areas and mills to trucks. Additional constraints are then added to link the time slots with the arrival of trucks at mills and harvest areas. The route of each truck is modeled as a sequence of trips, each trip represents either an empty-loaded travel from the mill or base garage to a harvest area, then a loaded travel from the harvest area to a mill. Having a limited number of trucks and a limited number of trips performed by each truck allows to easily represent the problem using an arc-based MIP formulation by adding truck and trip indices to each decision variable. However, a limitation on the number of trips that can be performed by each route is imposed *a priori* to prevent combinatorial explosion.

Another way to tackle the TTVRP is the consideration of a network flow-based model. Additional nodes are added to the initial network graph to represent each type of activity, such as loading, unloading, loaded trips, unloaded trips and queuing times. El Hachemi et al. (2015) [5] present a decomposition approach for a weekly tactical log truck scheduling problem. The problem is described as the routing of trucks that pick-up logs from forest areas and deliver them to mills during a planning horizon of several days (a week) while considering the inventory stock limit and the production planning at each mill. The authors solve the problem in two phases. The first phase of the solution method consists of a mathematical formulation used to solve the tactical transportation and allocation problem of truckloads from the harvest areas to mills. The second phase consists in solving the daily routing and scheduling of truck routes while minimizing the empty-loaded travel times and queuing times for service. A mathematical formulation is proposed based on an enriched space-time network model. Moreover, the planning horizon is discretized into equal time slots corresponding to the loading and unloading times, assumed to be similar, and the arcs corresponding to loading and unloading activities are duplicated as many as the number of slots in each day. Subsequently, the restriction of a single loading/unloading operation per time slot is automatically enforced by the flow conservation constraints and the unit capacity on the loading/unloading arcs.

Palmgren et al. (2004) [12] tackled a log-truck routing problem with transportation allocation and proposed a near-exact method based on column generation to solve a similar problem as in [4]. New columns are generated using a k-shortest path algorithm applied on an extended network, where most of the complicating constraints are astutely embedded, such as supply and demand levels and transported quantities. After solving the column generation at the root node to optimality, the integer version of the master problem is solved after including all the new generated columns. Rix et al. (2015) [14] also used a column generation approach to solve a tactical wood flow problem. The authors presented a MIP model and solved it using column generation. The pricing problem in the column generation phase is solved as a resource constrained shortest path problem (RCSPP). Rey et al. (2009) [13] proposed a similar approach and used a dynamic programming approach to solve the shortest path problem. Although the efficiency of the column generation approaches, solving RCSPP constitutes a substantial computational burden. Moreover, solving the pricing problem optimally improves the linear relaxation but not necessarily the optimal solution, which minimizes the outcome of such complex pricing approach.

Constraint programming (CP) is a practical approach, thanks to the ease of synchronization modeling and loaders and the optimization of the queuing times. Moreover, in contrast to IP, CP is able to find feasible solutions close to the optimal since early stages of the search. CP represents loading and unloading times as domain variables and by propagating the synchronization constraints only when needed, it allows an efficient implementation of those constraints. Audy et al. (2011) [2] propose a three-phase heuristic for an operational weekly log-transportation problem. A large set of feasible routes are generated by simple enumeration following a set of rules. Then, the construction of the set of circuits that satisfy all the demand and supply constraints is performed using a set covering formulation. The objective function in

the first two phases is the minimization of transportation costs. The final phase of the heuristic consists of a CP model that aims at the construction of a sequence of non-overlapping trips for each truck and the scheduling of loading and unloading activities at the harvest areas and mills by fixing their start and end times. The objective function in this phase is the minimization of queuing times of trucks which mainly depend on the availability of resources (loaders) that were relaxed in the previous models. El Hachemi et al. (2013) [7] present a decomposition approach for the same problem tackled in [5]. The authors solve the problem in two phases. The first phase consists in solving a MIP for the tactical inventory problem for the whole planning horizon. The second phase is a CP-based hybrid Iterated Local Search that tackles the routing of trucks and scheduling of loading/unloading activities where the CP part of the algorithm performs the synchronization of trucks and loaders as well as the optimization of queuing times. El Hachemi et al. (2011) [6] tackled a daily log-truck scheduling problem. In this problem, a fixed set of transportation requests is considered. The requests are supposed to be derived from decisions made at the weekly planning level. The objective function express a combination of empty driven routing costs, queuing times at harvest areas and mills, and also the queuing times of log-loaders in forest areas. The authors propose a two-phase hybrid method where the first phase is an IP that models the routes of trucks between harvests and mills as a network flow problem. This model optimizes the empty driven distance, whereas in the second phase the queuing times of trucks and log-loaders are optimized through a CP model.

We have clearly noticed the lack of methods that efficiently can solve large scale TTVRP with queuing considerations, and very often, the destination is done on a tactical level without considering queuing. We present a hybrid method to solve the TTVRPQ. The solution approach is based on the discretization of the planning horizon into small intervals called *time slots*, which allows easy management of loading/unloading capacities. The hybrid method incorporates two phases. The first phase is based on a rolling horizon heuristic used to generated an initial set of solutions for the problem. Those generated solutions are then used to initialize a pool of routes which are provided to the second phase, consisting of a column-generation based procedure used to generated new routes. We use in the master problem a set partitioning formulation with side constraints that allocates time slots required by loading and unloading operations of trucks, in addition to the assignment of routes to trucks. The key feature of this procedure is the use of pricing heuristics to construct new routes with negative costs in order to improve the best solution found so far. Moreover, The key idea of those heuristics is to astutely generate new columns based on the existing ones, by astutely modifying the time intervals of loading and unloading operations, without modifying the physical structure of the basis columns or re-arranging the sequence of visits. One motivation is that there are relatively small number of trips in a full route so it is likely that the optimal physical representation are already generated in the initial pool. Our approach is highly flexible with no limitations on the number of requests per route, obtained good solutions relatively faster, succeeded to solve instances with more that 250 requests and 120 trucks, and capable of dealing with fine-grained discretization of the planning horizon, allowing the minimization of idle times. Performance evaluation of the proposed approach has been conducted using benchmark instances derived from real-life data.

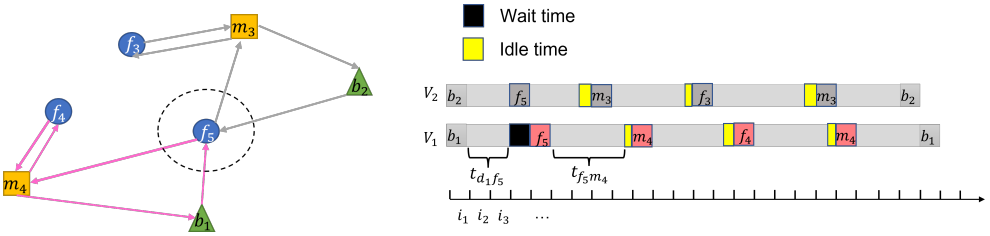
The remainder of this paper is as follows. Problem description and the mathematical formulation is provided in Section 2. Solution approach is described in Section 3. Experimentation tests and performance evaluation of the proposed approach are presented in Section 4. Finally, discussion of the results, conclusions and research perspectives are provided in Section 5.

2 Problem description and mathematical formulation

A major issue when solving the TTVRPQ are the constraints on loading and unloading capacities at, respectively, harvest areas and mills. To handle those constraints, we need first to record at every moment, and at every location the number of trucks being loaded or unloaded. Hence, we partition the planning horizon into a set of intervals of equal length called *time slots*. The width of time slots is generally chosen such that the durations of loading and unloading operations are easily expressed by a predetermined number of time slots. Before a truck can be loaded or unloaded, it should first find available time slots, during which the operation is going to be performed. The initial number of available time slots at a given site and at every moment is equal to its loading/unloading capacity.

We present in the following some necessary notation. A problem instance of the TTVRPQ can be modeled using a direct graph $G = (N, A)$ where N is the set of vertices associated with locations and A is the set of arcs. The set N incorporates a set of home depots B, a set of harvest areas H and a set of mills M. Each harvest area $h \in H$ has loading capacity l_h and each mill $m \in M$ has an unloading capacity l_m. The set of arcs is $A = \{(i,j)|(i,j) \in (B,H) \cup (M,H) \cup (H,M) \cup (M,B)\}$, where each arc $(i,j) \in A$ is associated with a travel time $t(i,j)$. We consider in the TTVRPQ a set of trucks V, each truck has a home depot $b_v \in B$, from which it starts and finishes its journey, an earliest departure time e_v and a latest end time f_v. Let us also consider the set of requests R, each request should be transported from its corresponding harvest area $h_r \in H$ and delivered to the associated mill $m_r \in M$. We also partition R into subsets $R_{hm}|(h,m) \in (H,M)$, where each subset R_{hm} includes the set of requests in R having as an origin $h \in H$ and a destination $m \in M$. The planning horizon is denoted by $\mathcal{Z}$, which is divided into small intervals of equal width called *time slots*. The set of time slots is denoted by I, where each time slot $i \in I$ has a starting time e_i and an ending time f_i. The width of the time slots is denoted by δ, whereas duration of loading/unloading operations is expressed by the number of time slots μ.

Figure 1 shows an example of a TTVRPQ solution composed of two routes. Figure 1a shows the topological distribution of the solution whereas Figure 1b shows the scheduling of loading and unloading operations, where queuing times are represented by black boxes. We assume a single loading/unloading capacity for all mills and harvest areas. The arrival of trucks V_1 and V_2 at the same time at harvest area f_2 forces truck V_1 to wait until the loading operation of truck V_2 is finished. The reader may notice the use of yellow boxes to represent the time between the truck arrival and the starting time of the next time slot. These residual time slots are called idle times, which are not included in queuing times. When the width of time slots is small enough, arrivals



(a) Topological representation of TTVRPQ solution

(b) Scheduling of loading and unloading operations

Fig. 1: Example of a TTVRPQ solution

within time slots do not incur longer times of inactivity. However, computational time when solving the problem may substantially grow. The best configuration would allow the best trade-offs between accuracy and computational times.

A solution S of the TTVRPQ is composed of a list of routes T_S, where each route $t \in T_S$ is associated with a truck $v_t \in V$. Each route $t \in T_S$ is composed of a sequence of requests. A request $r \in R$ can be seen as a sequence of two operations, a loading operation o_r^l and an unloading operation o_r^u. Moreover, for each truck $v \in V$, we associate two dummy operations o_v^e and o_v^f to represent the departure and the arrival of the truck at the associated home depot. Each operation o is associated with a node $n_o \in N$ and eventually a request $r_o \in R$ if $n_o \in H \cup M$, and has an arrival time ar_o, an arrival time slot a_o calculated as $a_o = \lceil \frac{ar_o}{\delta} \rceil$ if $n_o \in H \cup M$, and calculated as $a_o = \lfloor \frac{ar_o}{\delta} \rfloor$ if $n_o \in B$. The time slot where the operation o starts is denoted by s_o, with $s_o \geq a_o$.

Hence, a route $t \in T_S$ can be defined as a sequence of operations $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\sigma|}\}$. The truck associated with σ is denoted by $v_\sigma \in V$. A queuing time at an arbitrary operation σ_i is computed as $WT_i^\sigma = s_{\sigma_i} - a_{\sigma_i}$. The accumulated queuing times between positions i and j in σ is computed as : $TWT_{ij}^\sigma = \sum_{k=i+1}^{j} WT_k^\sigma$. The accumulated queuing times of σ is denoted by $TWT^\sigma = TWT_{1|\sigma|}^\sigma$.

The objective function also integrates queuing times so that they are minimized together with duration. The cost of a route is computed as :

$$c_\sigma = (s_{\sigma_{|\sigma|}} - s_{\sigma_1} + \rho \times TWT^\sigma) \tag{1}$$

Here ρ is a unit cost associated with queuing times.

We propose in the following a mathematical formulation of the TTVRPQ based on a set partitioning model. Let us first consider the set of all feasible routes Ω, each route is associated with a truck and starts from the corresponding home depot, performs a sequence of requests and returns back to the home depot. Time slots at the harvest areas and mills are allocated to loading and unloading operations performed by the truck routes. We define a three-index binary constant a_{tmi} to indicate whether a unloading operation is performed at mill $m \in M$ by truck route $t \in \Omega$ during time slot $i \in I$. Similarly, we define the three-index binary constant b_{thi} to indicate whether a loading operation is performed at harvest area $h \in H$ by the truck route

$t \in \Omega$ during time slot $i \in I$. We define a two-index boolean coefficient u_{tr} indicating whether request in $r \in R$ is fulfilled by route $t \in \Omega$. We note that the set of routes Ω is partitioned into $|V|$ subsets Ω_v, each corresponding to a specific truck $v \in V$. We need also to define a decision variable $y_t \in \{0, 1\}$ that indicates whether route $t \in \Omega$ is selected in the optimal solution.

Hence, the mathematical formulation for the set partitioning problem, denoted by $SPP1$, is as follows :

$$MIN \sum_{t \in \Omega} c_t y_t \tag{2}$$

$$\sum_{t \in \Omega_v} y_t = 1 \quad \forall v \in V \tag{3}$$

$$\sum_{t \in \Omega} b_{thi} y_t \leq l_h \quad h \in H, i \in I \tag{4}$$

$$\sum_{t \in \Omega} a_{tmi} y_t \leq l_s \quad m \in M, i \in I \tag{5}$$

$$\sum_{t \in \Omega} u_{tr} y_t = 1 \quad \forall r \in R \tag{6}$$

$$y_r \in \{0, 1\} \quad r \in \Omega \tag{7}$$

The objective function aims at the minimization of the sum of the route costs (2). The route costs include travel times, queuing times and loading/unloading times, and they are assumed to be known for each route. Constraints (3) guarantee that each truck can perform at most one route. Constraints (4) limit the number of trucks to be loaded during each time slot at each harvest area. Similarly, constraints (5) ensure the respect of unloading capacity at each mill during every time slot. Constraints (6) guarantee that each request $r \in R$ is fulfilled exactly once by a single route $t \in \Omega$. Constraints (7) are domains definition.

We suggest in the following an alternative formulation to $SPP1$. It can be seen as a compact formulation, where instead of considering each request $r \in R$ separately in constraints (6), subsets of requests $R_{hm} \subset R$ related to each pair of harvest area and mill $(h, m) \in (H, M)$ are grouped together into a single constraint. Let w_{thm} be an integer coefficient indicating the number of requests in R_{hm} fulfilled by route $t \in \Omega$. Hence, the extended formulation $SPP2$ for the set partitioning problem is a follows.

(2), (3), (4), (5), (7)

$$\sum_{t \in \Omega} w_{thm} y_t = |R_{hm}| \quad \forall (h, m) \in (H, M) \tag{8}$$

3 Column generation based approach

$SPP1$ is called the master problem. Solving the $SPP1$ on Ω using existing commercial solvers is not practically possible, since Ω grows exponentially with the size of the problem (number of requests $|R|$). Instead, in the column generation approach, the

basic idea is to solve the $SPP1$ while only considering a subset of routes and to generate new columns only if they can improve the linear relaxation of the $SPP1$. The objective is to find a set of routes Ω' that provides the same optimal solutions as the master problem when applied on Ω. The restricted version of the column generation form is called the *restricted* master problem (RMP). An initial subset Ω' of routes can be generated either heuristically or randomly.

Technically, the column generation process to solve VRP is carried out as follows. Once the linear relaxation of the RMP is solved, the optimal values of the dual variables are extracted and provided to the pricing problem. Those dual costs are used to compute to the reduced cost of the columns. The aim of the pricing problem then is to find the column with the minimum reduced cost, and if it has a negative value, it is added to Ω'. The augmented RMP is again solved and the new values of the dual variables are considered. This process is iterated until no columns with negative reduced costs are found [8].

In our case, let us first define dual variables α_v, β_{si}, λ_{hi} and γ_r to be respectively the optimal values of the dual variables for constraints (3), (4), (5) and (6) of the primal problem.

The dual problem of the linear relaxation of the set partitioning formulation is as follows:

$$MAX \sum_{v \in V} \alpha_v + \sum_{m \in M} \sum_{i \in I} l_m \beta_{mi} + \sum_{h \in H} \sum_{i \in I} l_h \lambda_{hi} + \sum_{r \in R} \gamma_r \tag{9}$$

$$\alpha_v + \sum_{m \in M} \sum_{i \in I} a_{tmi} \beta_{mi} + \sum_{h \in H} \sum_{i \in I} b_{thi} \lambda_{hi} + \sum_{r \in R} w_{tr} \gamma_r \leq c_t \quad \forall v \in V, \forall t \in \Omega_v \tag{10}$$

$$\alpha_v \leq 0 \quad v \in V \tag{11}$$

$$\beta_{mi} \leq 0 \quad m \in M, i \in I \tag{12}$$

$$\lambda_{hi} \leq 0 \quad h \in H, i \in I \tag{13}$$

$$\gamma_r \in \mathbb{R} \quad r \in R \tag{14}$$

It is noteworthy to mention that in our case, since we have a heterogeneous fleet, we consider solving $|V|$ separated pricing problems at each iteration.

For the calculation of the reduced cost of a route $t \in \Omega_v$, we proceed as follows :

$$C_t = c_t - \alpha_v - \sum_{m \in M} \sum_{i \in I} a_{tmi} \beta_{mi} - \sum_{h \in H} \sum_{i \in I} a_{thi} \lambda_{hi} - \sum_{r \in R} w_{tr} \gamma_r \tag{15}$$

3.1 Pricing problem

The basic idea of our approach is to generate new columns with negative reduced costs by simply modifying the existing columns in Ω'. Instead of generating new columns by solving a resource constrained shortest path problem (RCSPP), the columns selected

in the basis of the LP relaxation of $SPP1$ are used to generate new columns. The generation of new columns does not incur any alteration of the physical structure of the routes, that is, the subset of requests visited by the truck routes as well as the order in which the requests are carried out. Instead, we focus on the re-assignment of time slots to loading and unloading operations, while maintaining the feasibility of the routes.

For this purpose, we suggest the following useful definitions. Given a sequence of operations σ, Kindervater and Savelsbergh (2018) [10] proposed to compute the *Forward Time Slack* FTS_i at σ_i, indicating how much forward delay is possible at the i^{th} operation without exceeding the planning horizon by the subsequent operations including operation σ_i. FTS_i is computed as : $FTS_i^\sigma = \min_{i \leq k \leq |\sigma|} \{TWT_{ik}^\sigma + s_{\sigma_k} - WT_k^\sigma\}$. For convenience, we denote FTS_0^σ as FTS^σ.

We also define the *Allowed Backward Shift* of a given sequence σ_i as the maximum gain in duration at the final operation yielded by shifting in the backward direction the service starting of operation σ_i, assuming of course that the service can start in an earlier date without violating time constraints. The *Allowed Backward Shift* at σ_i is denoted by ABS_i^σ and it is computed as follows: $ABS_i^\sigma = min\{s_{\sigma_i} - a_{\sigma_i}, ABS_{i+1}^\sigma\}$. For convenience, we denote ABS_0 of σ by ABS^σ.

Adding new routes with negative reduced costs to the RMP has two benefits. The first benefit is the improvement of the objective value of the relaxation (since they have negative reduced costs). Moreover, delaying some operations may allow non-basic but promising columns to enter to the basis.

3.1.1 Forward shift operator

The heuristic looks for operations having a strict positive value of the dual variable associated with their time slots (β or λ) and perform as many shifts as possible. The forward shift is then propagated to the predecessors until the dummy operation associated with the arrival at the home depot. Every elementary movement, if feasible, can give rise to a new column. A forward shift of the operation σ_i by k time slots is feasible if : $k \leq FTS_i^\sigma$. If the modified column scores a negative reduced cost, a new route is then created and added to the pool.

3.1.2 Backward shift operator

The heuristic looks for operations having a strict positive value of the dual variable associated with their time slots (β or λ) and performs as many shifts as possible. Every elementary backward shift may give rise to a new column. Every backward shift is systematically propagated to the predecessors until the dummy operation associated with the departure. A backward shift of the operation σ_i by k time slots is feasible if : $k \leq ABS_i^\sigma$. If any of the modified columns scores a negative reduced cost, a new route is then created and added to the pool.

3.2 pricing procedure

We propose in the following the full procedure used to generate new columns with negative reduced costs. The procedure iteratively applies the above pricing heuristics

on every route in different sequences of movements and on different operations, covering thereby a large combination of elementary moves. Hence, a single column can yield to the creation of several new columns with different reduced costs while keeping the same basic structure of the initial route.

Algorithm 1 describes the pricing procedure. It starts from an initial set of columns Ω_{init} provided as an input. Then it starts an iterative process of generating new columns. At each iteration, columns are retrieved one by one from the current pool of columns Ω_{curr} (line 7), which is initialized in the beginning by Ω_{init} (line 3), then sequentially applies the pricing heuristics to generate new set of columns with negative reduced costs (lines 8,9). At the end of each iteration, the new generated columns in Ω_{tmp} are grouped in a set Ω_{new} and the current set of columns Ω_{curr} (line 11) is updated using the new set of columns Ω_{tmp} (line 12). A hash function is used to discard duplicates every time a generated column is added to Ω_{new}.

Algorithm 1: PRICING PROCEDURE

input : Initial set of columns Ω_{init}
output: Set of new columns Ω_{new}

```
1  Ω_new ← ∅
2  improve ← True
3  Ω_curr ← Ω_init
4  while (improve) do
5      improve ← False
6      Ω_tmp ← ∅
7      foreach C ∈ Ω_curr do
8          improve ← forwardShift(C, Ω_tmp) || improve
9          improve ← backwardShift(C, Ω_tmp) || improve
10         if (Improve) then
11             Ω_new ← Ω_new ∪ Ω_tmp
12             Ω_curr ← Ω_tmp
13         end
14     end
15 end
16 return Ω_new
```

3.3 Creation of initial pool of columns

We present in the following a fast heuristic to generate an initial pool of routes to populate Ω_{init}. The proposed approach is inspired by a rolling horizon heuristic embedded in a well-known decision support system called ASICAM [15]. The basic idea of the heuristic is to sequentially construct a set of truck routes while performing an allocation of loads from harvest areas to mills such that supply and demand constraints are respected. In our case, since the requests are already predetermined, the aim is

to minimize route duration, with particular interest to the minimization of queuing times.

Fig. 2 presents a flowchart for the rolling horizon heuristic.

The algorithm receives in its input a solution composed of a set of $|V|$ empty routes. Also, for each pair of harvest area and mill $(h, m) \in (H, M)$, the algorithm receives in its input the set of requests R_{hm}. The set of the unsatisfied requests is denoted by Δ, and it is initialized by the set of requests R.

The algorithm starts by the initialization of a roll-out T by the starting time. The list of trucks free to satisfy new requests at instant $T + \delta * \mu$, V_T, is computed, then for each subset of requests $R_{hm}|(h, m) \in (H, M)$, the truck yielding the best insertion according to a criterion cr_1, is selected. The goal of criterion cr_1 is to favor insertions that minimizes the total duration and queuing times. It is computed as follows. Let o be the last operation performed by truck $v \in V_T$, which can be an unloading operation or dummy departure operation, and let $s'_{o^l_r}$ and $s'_{o^u_r}$ be the earliest time slots going to be assigned, respectively, to the loading and unloading operations of the corresponding request r, let $ar'_{o^l_d}$ and $ar'_{o^u_d}$ be the arrival times at loading and unloading locations of r. $cr_1(v, r)$ is computed as follows:

$$cr_1(v, r) = \frac{WT_{vr} + \delta \times \mu}{\mathcal{Z}} \times \frac{t(n_o, h_r)}{T_{max}} \tag{16}$$

where $T_{max} = max\{t(i, j)|(i, j) \in (M, H)\}$ denotes the duration of the longest unloaded trip, whereas WT_{vr} is the queuing time incurred by the the current insertion and it is computed as:

$$WT_{vr} = (e_{s'_{o^u_r}} - ar'_{o^l_r}) + (e_{s'_{o^u_r}} - ar'_{o^u_r}) \tag{17}$$

If no feasible insertion is found for all subsets $R_{hm}|(h, m) \in (H, M)$, the algorithm checks whether all requests in Δ are inserted ($\Delta = \emptyset$). In this case the algorithm terminates. Otherwise, the roll-out T is incremented by $\delta + \mu$ and reiterates.

In case where feasible insertions are found, the best feasible insertion for each pair $(h, m) \in (H, M)$ is selected and inserted in a candidate list. The list is sorted in non-decreasing order using a second criterion $cr_2(v, r)$. Let us first denote by $R^H_h \subset R$ the set of requests originating from harvest area $h \in H$, and R^M_m the set of requests destined to mill $m \in M$. Also, we denote by f^H_h the actual number of fulfilled requests originating from harvest area $h \in H$, and f^M_m the number of those destined to mill $m \in M$ until the current value of T.

We also compute the route duration for each truck until the last request $c_v, v \in V$. The consumed time for each truck $v \in V$ is computed as $c_v = f_{s_o} - e_v$, where o is the last operation performed by truck $v \in V$ before returning to the depot. The total consumed time for all the routes is computed as $C = \sum_{v \in V} c_v$.

Hence, $cr_2(v, r)$ is computed as:

$$cr_2(r, v) = cr_1(r, v) \times cng_1(r, v) \times cng_2(r, v) \times fratio(r, v) \times iep(r, v) \tag{18}$$

where $cng_1(r, v)$ represents a congestion parameter at mill $m \in M$, and it is computed as :

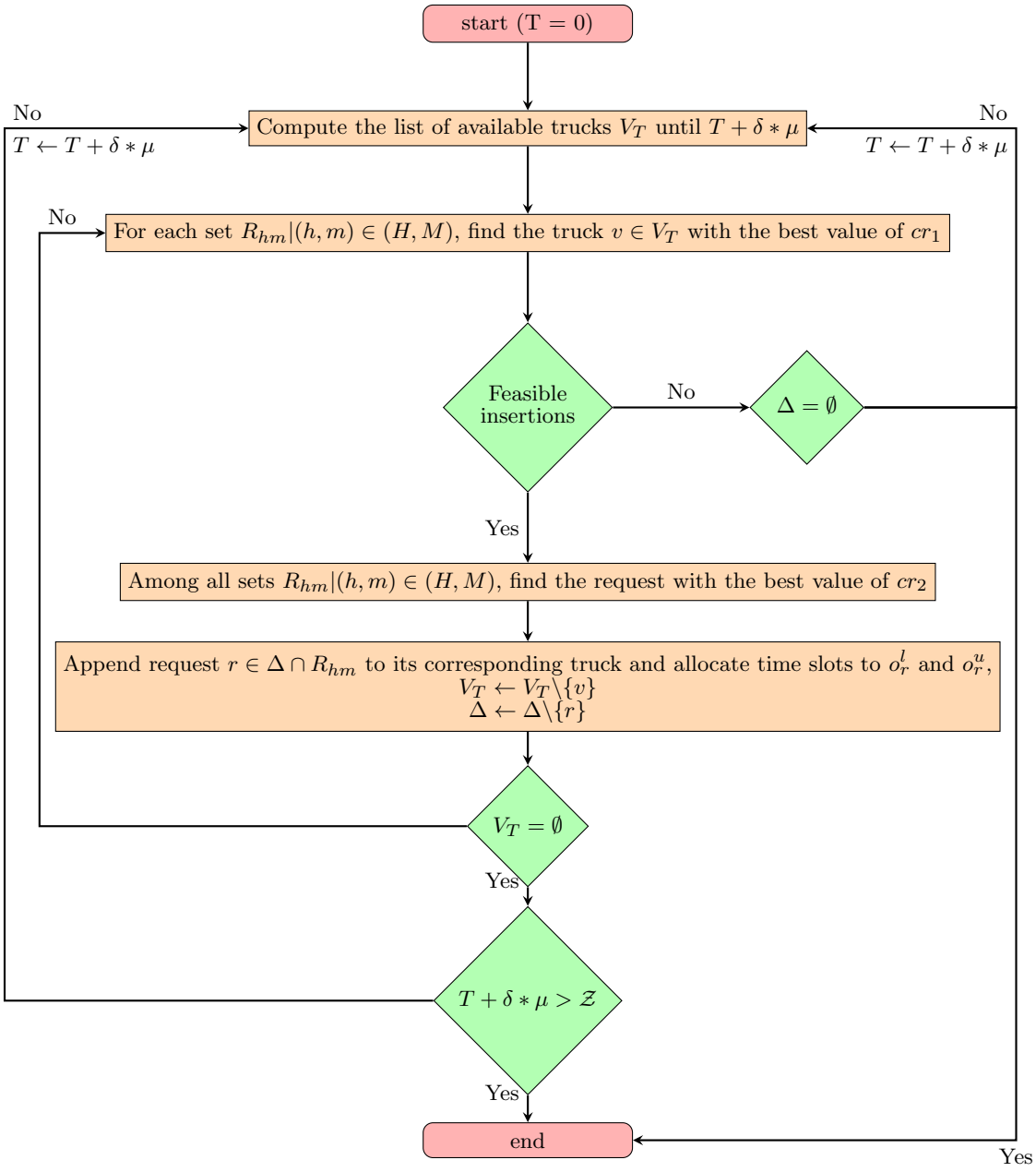


Fig. 2: Diagram flow of the rolling horizon heuristic

$$cng_1(r,v) = 1 - min(0, \frac{C}{\mathcal{Z} \times |V|} \times |R_{m_r}| - f^M_{m_r}) \tag{19}$$

$cng_2(r,v)$ is a congestion parameter at harvest area $h \in H$, and it is given by:

$$cng_2(r,v) = 1 - min(0, \frac{C \times |R_{h_r}|}{\mathcal{Z} \times |V|} - f^H_{h_r}) \tag{20}$$

$fratio(r,v)$ is the consumed time ratio of each truck $v \in V_T$:

$$fratio(r,v) = \frac{max(0, f_v - f_{s_o}) + \delta \times \mu}{\mathcal{Z}} \tag{21}$$

$iep(r,v)$ is used as a lower estimator for the maximum possible of requests between $(h_r, m_r) \in (H, M)$ to satisfy before the end of the planning horizon:

$$iep(r,v) = \frac{max(0, \frac{1}{2} \times \widehat{f}_{h_r m_r} - (|R_{h_r m_r}| - \hat{f}_{h_r m_r}))}{\widehat{f}_{h_r m_r}} \tag{22}$$

where $\hat{f}_{hm}$ denotes the number of requests in R_{hm} already satisfied in the solution, and $\widehat{f}_{hm}$ with wide hat notation is an upper bound on the number of requests between $h \in H$ and $m \in M$ that can be performed during the remaining time, and it is updated after each insertion.

While the first criterion cr_1 favors the insertion of trips incurring lesser queuing times and unloaded travel times, the second criterion cr_2 aims at ensuring regular arrivals at harvest areas and mills (cng_1 and cng_2). Moreover, this criterion also favors the early insertion of trips requiring much more time to be fulfilled (thanks to parameter $fratio$).

Once the best insertion according to cr_2 is applied and the corresponding truck is removed from V_T, the corresponding request $r \in \Delta$ is also removed. If V_T is not empty, the algorithm recomputes the next feasible insertion at the same rollout T, otherwise the rollout is incremented by a step of $\delta + \mu$. If the new rollout reaches the planning horizon $\mathcal{Z}$, the algorithm ends. Otherwise the algorithm starts a new iteration.

To further enhance the initialization procedure, we suggest to embed the rolling horizon heuristic inside an iterative procedure called the Adaptive Rolling Horizon Heuristic ARH^2. The goal of the ARH^2 is the introduction of randomness to allow the rolling horizon heuristic cover larger part of the search space. The basic idea is to modify the formula of cr_1 by adding two weights α and β, associated with the two terms related to, respectively, queuing times and unloaded travel times. Hence, cr_1 is rewritten as:

$$cr_1^{\alpha,\beta}(v,r) = (\frac{WT_{vr} + \delta \times \mu}{\mathcal{Z}})^{\alpha} * (\frac{t(n_o, h_r)}{T_{max}})^{\beta} \tag{23}$$

For every combination of values of (α, β), the execution of the rolling horizon heuristic can yield a different solution. Every modification of the weights changes the relative importance of criterion parameters and, hence, allowing to construct diversified solutions during the iterative process as in (Yahiaoui et al., 2023) [16].

Practically, a number of different combinations of (α,β) are generated during each iteration. Based on an initial values of (α,β), those combinations are generated as

follows. Four combinations are generated by either increasing or decreasing the current values of (α,β) by a step equal to 0.1, while ensuring that the values of (α, β) within their respective intervals ($[1, 2.5[, [0.5, 1.5[$). A fifth combination is generated by randomly choosing (α, β) within their respective intervals. The last combination maintains the same value of α while β is randomly chosen inside its interval $[0.5, 1.5[$. At the end of each iteration, the combination that led to the solution with the best objective value is used as a basis for the next iteration. In the very first iteration, both values of (α and β) are set to 1.

The total number of iterations of the ARH^2 is set to the size of the fleet $|V|$.

3.4 General flow

We describe in the following the general flow of the mathheuristic approach. Algorithm 2 provides a pseudo code of the general approach.

Algorithm 2: MATHEURISTIC APPROACH

input : Set of deliveries D, Available trucks V, set of harvest areas H, set of mills S, required number of deliveries between harvest area $h \in H$ and mill $s \in S$, number of iterations for adaptive heuristic $Iter_{max}$
output: Set of new routes Sol_{best}

1 $counter \leftarrow 0$
2 $\Omega \leftarrow \emptyset$
3 $\Omega_{init} \leftarrow ARH^2(V, R)$
4 $negative \leftarrow True$
5 $\Omega_{curr} \leftarrow \Omega_{init}$
6 $lpSolver \leftarrow initLP\_SPP1(V, R, \Omega)$
7 **while** $(negative)$ **do**
8 $\quad (\mathbf{T}^*, \mathbf{rc}^*) \leftarrow lpSolveSPP2(\Omega_{curr})$ (See Section 2)
9 $\quad \Omega_{new} \leftarrow PricingProcedure(\mathbf{T}^*, \mathbf{rc}^*)$ (See Section 3.2)
10 $\quad$ **if** $(\Omega_{new} \neq \emptyset)$ **then**
11 $\quad\quad \Omega_{curr} \leftarrow \Omega_{curr} \cup \Omega_{new}$
12 $\quad$ **end**
13 $\quad$ **else** $negative \leftarrow False$
14 **end**
15 $\Omega \leftarrow \Omega \cup \Omega_{curr}$
16 $mipSolver \leftarrow init\_SPP2(V, R, \Omega)$
17 $S_{best} \leftarrow mipSolver.solve()$
18 **return** S_{best}

The algorithm starts by the generation of a set of solutions using the ARH^2 (line 3). Then the column generation process takes place (lines 5-19). This iterative process starts by solving the RMP (line 8) using $SPP1$. The solver returns the optimal solution for the linear relaxation which is composed of the set of columns of the basis. The routes associated with these columns are provided to the pricing procedure (line 9) in

order to generate a new set of routes with negative reduced costs (See Section 3.2). If at least one new route is found, they are added to the current set of routes (lines 10,11) and the algorithm iterates. On the contrary, if no route is found, then the column generation procedure is terminated (line 13). Finally, the set of routes produced by the column generation process in addition to the initial set of routes are used to solve an integer set partitioning problem using formulation ($SPP2$) and the best solution found is returned (lines 16, 17). It is noteworthy to mention that during preliminary tests, the compact formulation $SPP2$ proved to be faster than $SPP1$ when solving the integer set partitioning.

4 Experimentation tests

We conduct experiments to assess the performance of our method. We compare our method against the ARH^2. The mathheuristic has been implemented using C++ standard library and tested using an HP Proliant DL 360 p G8, with 128 GB of RAM and two Intel Xeon E5-2670V2 2.50 GHz, with 20 cores able to execute up to 40 threads, equipped with a Ubuntu server 22.04 LTS. We perform ten runs of the mathheuristic based on random seeds for each instance, and we report the results of the initialization phase performed by ARH^2 and the final results after the column generation phase. For both methods, we report for each instance, the best objective value $Obj.$, the average objective value $\overline{Obj.}$, the best duration $Dur.$, the average duration $\overline{Dur.}$, the best travel distance $Dist.$, the average travel distance $\overline{Dist.}$, the best queuing times $QT.$, the average queuing times $\overline{QT}$. and average computational times CPU.

We propose a benchmark instances derived from two study cases provided by two companies in forestry. The case studies contain a list of requests performed during the year 2017. A major part of the requests are assigned to weeks. To construct instances for each working day, we performed a pre-processing on the data to assign requests to specific days. We used a modified version of the mathematical model presented by Gronalt and Hirsch (2013) [9]. As a result, we elaborated 75 instances, covering 15 weeks, with 5 instances for each week. The planning horizon is set to $\mathcal{Z} = 15h$ for all instances.

Travel distances and travel times are computed using Open Street Routing Machine (OSRM) API[1], which is a Modern C++ routing engine for shortest paths in road networks.

4.1 Parameters settings

We present in this section the tuning of some parameters of the mathheuristic. We denote by the GAP in the following sections the percentage optimality gap provided by CPLEX solver. It is computed as

$$GAP = \frac{UB - LB}{UB} * 100, \tag{24}$$

[1] http://project-osrm.org/

where UB is the objective value of the best feasible solution found so far and LB is the best lower bound.

4.1.1 Time limit for the integer SPP

Table 1 depicts the results obtained by the mathheuristic while varying the time budget allocated when solving the final MIP in Algorithm 2 (line 17) at the end of the solution process. We notice that the general performance parameters of the mathheuristic have not been impacted when increasing the time budget from 300 seconds to 500 seconds, except for computational times, which increased from 447 seconds for a time limit of 300 seconds up to 567.2 seconds when the time budget has a value of 500 seconds. It noteworthy to mention that the computational times comprise the initialization of the pool, the column generation phase, and finally the solution of the integer SPP.

The main reasons for such performance is that the vast majority of instances are solved to optimality within 300 seconds time budget, whereas some other instances require a substantial amount of time to find the optimal solution (greater than 500 seconds.

Table 1: Impact time limit on the performance of the mathheuristic

Time Limit (s)	300	400	500
$\overline{Dur.}(h)$	855.6	855.3	854.6
$\overline{Cost}(km)$	39928.8	39909.3	39870.9
$\overline{QT}(h)$	0.006	0.008	0.006
$GAP(\%)$	0.659	0.626	0.542
$CPU(s)$	447	511.2	567.2

4.1.2 Size of the initial pool

We investigate in this section the impact of the size of solution pool on the contribution to the overall performance of the mathheuristic. The results are presented in Table 2. During the execution of ARH^2, the constructed solutions are stored in a pool. It is the size of this pool that is going to be investigated. The pool size of solutions is varied from 200 to 500. At the end of ARH^2, individual routes are extracted from solutions in the pool, while avoiding duplicates (row called $Cols$ in Table 2). Additional performance indicators are defined. GAP denotes the overall percentage gap after solving the integer $SPP2$ while including the column generation phase, whereas Imp denotes the improvement percentage achieved thanks to the column generation compared to solving the integer $SPP2$ directly based on the initial pool of routes, $+Cols$ is the percentage increase in the number of the columns after the column generation phase.

According to Table 2, the overall objective value decreases when increasing the pool size. Moreover, we notice a slight increase in computational times when increasing the pool size going from 450.8 seconds for a pool size equal 300 and up to 470.6

Table 2: Impact time limit on the performance of the Math-heuristic

Pool size	200	300	400	500
$\overline{Obj.}$	3063.6	3046.5	3036.9	3034.1
$Cols$	11526	15060	18063	18991
$+Cols$	28.45	20.69	18.11	16.61
$Impr(\%)$	1.15	1.12	1.05	1.04
$GAP(\%)$	0.62	0.61	0.61	0.62
$\overline{Dur.}$	850.9	846.1	843.5	842.8
$\overline{QT.}$	0.02	0.03	0.01	0.01
CPU	450.8	465.6	464.8	470.6

seconds for 500-solution pool size. This observation suggests that the performance of the mathheuristic reaches a stability point with less than 500 solutions, especially in terms of the number columns, for which the increase is relatively small when the size increases from 400 to 500 solutions. We also notice that the number of columns generated by the column generation phase decreases in percentage when increasing the size of solutions pool, going from 28.45% for a pool size of 200 and reaching a percentage of 16.61% when the pool size is equal to 500. On row 5 of Table 2, the improvement achieved due to the additional columns generated by the pricing heuristics reaches 1.15% for a pool size equal to 200, and slightly decreases when increasing the pool size, reaching 1.04% for a pool size equal to 500. Regarding the optimality gap, we notice that it stays stable, which means that it only depends on the time budget (300 seconds). Regarding the performance of the general approach, we notice that the overall average objective function decreases when increasing the pool size decreasing from 3063 to 3034.1. The same behavior is observed on the overall average duration, whereas the queuing times stay relatively close to 0.

4.2 Sensitivity analysis

We investigate in this section the contribution of the column generation to the performance of the mathheuristic.

4.2.1 Sensitivity analysis of the column generation phase

We compare the results obtained by the integer SPP with and without a column generation phase.

Table 3 presents the overall performance parameters of both configurations. The column generation phase enriched the pool of columns by an overall of $12070 - 8953 = 3117$ additional column, that is 34.81% of the initial pool size, while consuming less computational times (447 seconds against 481.1 seconds) when solving the integer SPP. Regarding the gap to best, has also been improved from 0.86% down to 0.66%. In details, we notice that solving the integer SPP after the column generation phase has improved the overall duration, recording a decrease from 864.1 hours to 855.6 hours. Queuing times also has been slightly improved, from an overall of 0.177 hours

Table 3: Contribution of the CG compared to integer $SPP2$ (Time limit = $300s$)

$Method$	$no-CG$	CG
$\overline{Dur.}(h)$	864.1	855.6
$\overline{Dist.}(km)$	40440.8	39928.8
$\overline{QT}(h)$	0.177	0.006
$GAP(\%)$	0.861	0.659
$NbCols$	8953	12070
$CPU(s)$	482.1	447

to 0.006 hours. The overall travel distance has also been improved from 40440.8 km to 39928.8 km.

4.2.2 Extended formulation vs. compact formulation

We compare the contribution of the initial formulation $SPP1$ against the compact formulation $SPP2$ during the column generation phase when solving the relaxation. It is noteworthy to mention that in both cases, the compact formulation $SPP2$ is used at the end to solve the integer set partitioning. Table 4 compares the performance measures of the matheuristic when using each of the two formulations.

Table 4: Contribution of the CG when using the extended formulation

$Formulation$	$SPP1$	$SPP2$
$\overline{Obj.}$	3064	3087
$ImprGAP.$	1.15	0.47
$\overline{Dur.}$	851	857
$\overline{QT.}$	0.02	0.19
$Cols$	11526	11899
$+Cols$	35.31	39.61
GAP	0.62	0.86
CPU	451	486

Generating of new columns based on $SPP1$ yielded the best overall objective value, 3064 against 3087 when using $SPP2$, that is an improvement of 1.15% thanks to $SPP1$ against only 0.47% achieved by $SPP2$. This improvement is observed mainly on the overall average duration, 851 hours by the extended formulation against 857 hours achieved by the $SPP2$. Regarding the queuing times, the value is close to 0 for both cases, with a slight advantage to the $SPP1$. The increase in the size of the pool was relatively higher in the case of the compact formulation, that is 39.61% for $SPP2$

against 35.31% of $SPP1$. However, the $SPP1$ achieved better optimality gap compared to the compact one, 0.62% against 0.86%, while requiring shorter computational times, 451 seconds on overall against 486 seconds for the compact formulation. As a result, the use of an extended formulation ($SPP1$) in the column generation proves to be more efficient than using the compact one ($SPP2$). This can be justified by the fact that the dual problem of $SPP1$ provides a better and detailed information about dual costs, which allows to achieve an efficient column generation process.

4.3 Computational tests

We provide in this section a detailed comparison between the best solution obtained by the ARH^2 during the initialization phase and the final solution obtained when solving the integer SPP at the end of Algorithm 2. Table 5 presents the obtained results of both methods reported for each benchmark instance. Table 5 depicts the list of instances along with their number of trucks and number of requests to be performed in columns (1-3), respectively.

Table 5: Summary of results of adaptive rolling horizon heuristic and mathheuristic

Instance			ARH^2							mathheuristic						
$Name$	$\|V\|$	$\|R\|$	Obj.	$\overline{Obj.}$	Dur.	$\overline{Dur.}$	QT	$\overline{QT}$	CPU	Obj.	$\overline{Obj.}$	Dur.	$\overline{Dur.}$	QT	$\overline{QT}$	CPU
$W2-D1$	75	202	3900	10943	1069.3	1075.8	0.33	1.93	185.3	3312	3325	920	43447.8	0	0	321.1
$W2-D2$	75	203	3805	3874	1053.7	1065.1	0	1.10	267.3	3191	3202	886.3	41438.1	0	0	539.8
$W2-D3$	75	203	3822	11913	1057.3	1068.6	0.33	1.83	260.6	3238	3242	899.3	42392.4	0	0	436.3
$W2-D4$	75	203	3890	5928	1061.7	1069	0.67	2.20	247.1	3254	3271	904	41992.4	0	0	473.6
$W2-D5$	80	203	4004	4031	1105.7	1111.9	0.33	0.77	330	3264	3281	906.7	42763.9	0	0	621
			3884	7338	1069.5	1078.1	0.33	1.57	258.1	3252	3264	903.3	42406.9	0	0	478.3
$W3-D1$	60	183	3151	3204	855.3	862.1	2	2.80	125.1	2700	2735	750	34175.5	0	0	422.5
$W3-D2$	60	182	2977	2992	821.3	829.2	0	0.20	123.1	2572	2587	714.3	32181	0	0	421.3
$W3-D3$	60	182	3030	3046	836	842.6	0	0.37	124.6	2610	2630	725	32741.8	0	0	402.8
$W3-D4$	55	182	2944	3983	794.3	800.9	1	2.77	103.3	2527	2557	702	31530.2	0	0.03	409.2
$W3-D5$	55	182	2899	10904	787.3	792.2	0.33	1.43	112.1	2556	2578	710	31996.2	0	0	409.6
			3000	4826	818.9	825.4	0.67	1.51	117.7	2593	2617	720.3	32524.9	0	0.01	413.1
$W4-D1$	75	198	13888	17999	1040	1051.5	2	5.93	238.7	3214	3224	892.7	41641.7	0	0	303.4
$W4-D2$	75	197	3718	3762	1029	1039.2	0	0.57	259	3086	3097	857.3	39208	0	0	548.2
$W4-D3$	75	198	3728	3757	1032.7	1036.9	0	0.67	245.7	3160	3169	877.7	40263	0	0	547.2
$W4-D4$	75	198	3716	3735	1024.7	1031.4	0	0.60	253.8	3088	3098	857.7	39457.9	0	0	523.6
$W4-D5$	75	197	3652	3680	1012.3	1019.1	0	0.30	251.9	2994	3010	831.7	38146.4	0	0	533.5
			5740	6586	1027.7	1035.6	0.40	1.61	249.8	3108	3119	863.4	39743.4	0	0	491.2
$W5-D1$	70	195	3532	3557	971	979.8	0.33	0.83	194.8	2869	2884	797	35869.7	0	0	488.1
$W5-D2$	70	194	3506	3517	965	970.7	0	0.63	205.7	2815	2836	782	34835.1	0	0	503.3
$W5-D3$	70	195	3535	3576	978.7	987.2	0	0.60	185.8	2845	2874	790.3	35439.8	0	0	487
$W5-D4$	70	194	3528	3558	980	984.1	0	0.43	193.1	2882	2899	800.7	35604.9	0	0	496.2
$W5-D5$	75	195	3694	3736	1020	1032.7	0	0.50	225.7	3007	3018	835.3	38024.8	0	0	494
			3559	3589	982.9	990.9	0.07	0.60	201	2884	2902	801.1	35954.9	0	0	493.7
$W6-D1$	75	203	3731	3749	1028.7	1031	0.33	1.03	236.4	3116	3127	865.7	40356.5	0	0	477.4
$W6-D2$	75	203	3851	3969	1053	1062.9	1.67	3.97	226.1	3233	3241	898	42203	0	0	375.9
$W6-D3$	70	203	3534	3563	966.3	972.2	1	1.77	215.6	2980	3022	827.7	37810	0	0	519.7
$W6-D4$	70	203	3605	3667	976	987.9	1.33	3.07	211.3	3037	3060	843.7	38572.5	0	0	515.4
$W6-D5$	70	203	3644	3709	994	1001	1.33	2.93	217.4	3089	3111	858	39842.7	0	0	508.7
			3673	3731	1003.6	1011	1.13	2.55	221.4	3091	3113	858.6	39756.9	0	0	479.4
$W7-D1$	75	215	3850	3916	1053.3	1062.7	1	2.50	209.2	3324	3342	923.3	43751.4	0	0	579.2
$W7-D2$	80	216	4278	11268	1127.7	1137.6	1.67	4.80	294.5	3566	3592	990.7	46855.2	0	0	589
$W7-D3$	85	216	4387	4480	1171.7	1183.2	3.33	6.13	320.4	3716	3736	1032.3	50079.1	0	0	551.1
$W7-D4$	80	215	3928	3971	1084.3	1093.4	0.33	0.97	283.6	3433	3447	953.7	45080.6	0	0	440.6
$W7-D5$	75	215	3794	3814	1040.7	1051.5	0.33	0.80	275.4	3244	3273	901	43062.7	0	0	579.4
			4047	5490	1095.5	1105.7	1.33	3.04	276.6	3457	3478	960.2	45765.8	0	0	547.9
$W8-D1$	105	239	16266	23585	1407	1426.9	21.33	40.23	594.3	4334	4372	1204	59877.2	0	0.03	948.3
$W8-D2$	120	238	9446	15349	1543.7	1564	78.33	103.30	840.8	4627	4704	1285.3	65847.2	0	0.33	1172
$W8-D3$	90	238	4639	4678	1267.3	1274.4	1.67	2.50	487.9	3734	3766	1037.3	50897.5	0	0	779.2
$W8-D4$	90	238	4804	12760	1283	1288.2	0.67	3.40	429.9	3836	3855	1065.7	51065.2	0	0	748.5
$W8-D5$	90	238	4789	6797	1281	1289.7	0.67	4.27	522.8	3878	3929	1077.3	54810.5	0	0.03	812
			7989	12634	1356.4	1368.6	20.53	30.74	575.1	4082	4125	1133.9	56499.5	0	0.08	892
$W10-D1$	80	198	3918	3957	1088	1096	0	0.33	263.9	3427	3433	952	46441.2	0	0	285.6
$W10-D2$	80	198	4379	11289	1118	1124.7	4.33	6.67	239.9	3521	3529	978	47356.3	0	0	318.9
$W10-D3$	70	198	3545	3608	978	983.9	0.67	1.83	212.7	3079	3089	855.3	40440.2	0	0	506
$W10-D4$	75	198	3809	3893	1044.7	1057.3	1.33	2.40	241.9	3342	3352	928.3	45443.6	0	0	487.7
$W10-D5$	80	199	5654	15687	1132.3	1139	38.67	44.07	235.9	3548	3561	985.7	47171.4	0	0	397.5
			4261	7687	1072.2	1080.2	9	11.06	238.9	3384	3393	939.9	45370.5	0	0	399.1
$W11-D1$	70	174	3664	3708	935.3	942	8	8.80	156.4	2892	2901	803.3	37521	0	0	157.5
$W11-D2$	60	173	3144	3177	842.3	850.4	1.67	3.20	127	2491	2498	692	31558.8	0	0	204.1
$W11-D3$	55	172	2800	2825	770.3	776.8	0	0.80	106.3	2378	2405	660.7	30500.5	0	0	411.6
$W11-D4$	55	172	2935	6094	787	795.8	2	6.37	98	2437	2455	677	30408.7	0	0	405.7
$W11-D5$	60	172	2984	2998	822.7	827.2	0.33	0.57	132.8	2422	2433	672.7	30519.9	0	0	328.2
			3105	3760	831.5	838.4	2.40	3.95	124.1	2524	2538	701.1	32101.8	0	0	301.4

continued on next page

Table 5 – continued from previous page

Instance			ARH^2							mathheuristic						
Name	$\|V\|$	$\|R\|$	Obj.	$\overline{Obj.}$	Dur.	$\overline{Dur.}$	QT	$\overline{QT}$	CPU	Obj.	$\overline{Obj.}$	Dur.	$\overline{Dur.}$	QT	$\overline{QT}$	CPU
$W22-D1$	75	193	3784	3806	1033.7	1042.3	0.67	1.50	217.6	3302	3309	917.3	45436	0	0	316.8
$W22-D2$	75	192	3715	3757	1028.7	1038.9	0	0.47	245.8	3277	3287	910.3	45018.7	0	0	369.1
$W22-D3$	75	192	3737	3782	1034.7	1045.9	0	0.47	240.9	3282	3294	911.7	44407	0	0	290.1
$W22-D4$	75	192	3803	3827	1044	1050.7	0.33	1.23	193.1	3298	3304	916	45227.9	0	0	444.5
$W22-D5$	75	192	3725	3745	1028	1034.9	0	0.53	234	3242	3252	900.7	44599.3	0	0	428.7
			3753	3783	1033.8	1042.5	0.20	0.84	226.3	3280	3289	911.2	44937.8	0	0	369.8
$W23-D1$	65	178	3316	3346	921	925.7	0	0.37	137.9	2773	2778	770.3	35932.9	0	0	212.4
$W23-D2$	70	178	3444	3473	950	959.3	0	0.53	166.7	2879	2893	799.7	37978.5	0	0	178.4
$W23-D3$	65	179	3330	3357	906.3	913.1	1	1.93	139.1	2740	2760	761	34726.8	0	0	153.2
$W23-D4$	65	179	3464	6532	924	931.1	3	5	148.5	2855	2862	793	36536.2	0	0	161.9
$W23-D5$	65	178	3264	3280	903.7	909.4	0	0.17	152.5	2693	2711	748	34547.3	0	0	413.4
			3364	3997	921	927.7	0.80	1.60	148.9	2788	2801	774.4	35944.3	0	0	223.8
$W24-D1$	65	205	3400	3421	928.3	934.7	0.67	1.57	143.9	2887	2907	802	35452.8	0	0	393.9
$W24-D2$	75	206	3979	4059	1028	1041.1	6	8.63	222.4	3152	3178	875.7	40079.4	0	0	317.9
$W24-D3$	70	205	3546	3577	971.7	981.2	0.33	1.23	209.3	3000	3010	833.3	37706.8	0	0	260.3
$W24-D4$	65	205	3343	3390	926	932.8	0	0.90	149.2	2897	2916	804.7	36889.8	0	0	432
$W24-D5$	65	205	3312	3319	913.7	919.3	0	0.27	175.4	2947	3038	818.7	40500.6	0	0.03	452.6
			3516	3553	953.5	961.8	1.40	2.52	180	2977	3010	826.9	38125.9	0	0.01	371.3
$W28-D1$	60	169	3149	10128	849.7	858.9	0	1	118.8	2681	2692	744.7	34947.7	0	0	368
$W28-D2$	60	170	3017	3034	835.7	839.9	0	0.30	114.1	2656	2664	737.7	34535	0	0	272.9
$W28-D3$	60	169	2935	2942	813.7	816.9	0	0.03	105.5	2599	2610	722	33752.6	0	0	249.4
$W28-D4$	55	169	2784	2814	773.3	780.1	0	0.17	86.7	2456	2467	682.3	31209.2	0	0	374.3
$W28-D5$	55	169	2792	2807	775.7	778.8	0	0.10	85.2	2425	2445	673.7	30502.9	0	0	391.1
			2935	4345	809.6	814.9	0	0.32	102.1	2563	2576	712.1	32989.5	0	0	331.1
$W38-D1$	70	201	3773	3816	991.3	1004.8	4	5.53	205.1	3036	3058	843.3	39365	0	0	443.7
$W38-D2$	70	201	3697	3782	994.7	1001.1	2.67	4.93	184	2951	2967	819.7	37957.8	0	0	491.5
$W38-D3$	70	201	3704	3793	993	1001.3	3.33	5.23	180	2951	2964	819.7	37151.8	0	0	310.7
$W38-D4$	70	201	3770	3870	999.7	1007.2	3.67	6.77	187	3004	3025	834.3	38951.4	0	0.03	492.3
$W38-D5$	70	201	3572	3601	987	995	0	0.53	200.5	2963	3005	823	38543.6	0	0	507.1
			3703	3772	993.1	1001.9	2.73	4.60	191.3	2981	3004	828	38393.9	0	0.01	449.1
$W39-D1$	80	194	3862	3887	1060	1069.8	0.33	1	296.4	3334	3346	926	44831.7	0	0	301.7
$W39-D2$	65	193	3425	11452	924.3	932.5	0.67	2.63	159.3	2930	2939	814	37839.4	0	0	479.3
$W39-D3$	70	193	3420	3439	949.3	952.4	0	0.30	192.4	2968	2978	824.3	38865.5	0	0	451.9
$W39-D4$	60	193	3088	3104	847.7	856.3	0	0.60	157.4	2690	2716	747.3	33437.5	0	0	461
$W39-D5$	65	193	3304	3314	910	914.3	0	0.63	175.6	2874	2898	798.3	37737.4	0	0	485.7
			3420	5039	938.3	945.1	0.20	1.03	196.2	2959	2975	822	38542.3	0	0	435.9
Mean			3997	5342	993.8	1001.9	2.75	4.50	220.5	3062	3080	850.4	39937.2	0	0.01	445.2

Based on Table 5, the mathheuristic succeeded to improve the objective value and the overage objective value of all instances compared to the best solution obtained by ARH^2 during initialization phase, with an improvement of 23.39% of the overall best objective value, and 42.34% improvement of the overall average objective value. The same observation is valid when comparing the total duration, for which the mathheuristic recorded an improvement of 14.43% compared to ARH^2 (850.4 hours against 993.8 hours for ARH^2). This is justified by the fact that ARH^2 succeeded to cover a large set of diversified routes, which allowed the $SPP2$ afterwards to substantially improve the best solution, with further improvement achieved after column generation (See Section 4.2.1). Interestingly, the mathheuristic succeeded to almost eliminate queuing times on all instances, achieving 0 hour overall best queuing times, and 0.01 hours on average overall. Even for some instances such as $W8-D1$, $W8-D2$, $W10-D5$, $W11-D1$, where the queuing times achieved by the ARH^2 are substantial (21.33 hours, 78.33 hours, 38.67 hours, and 8 hours respectively), the matheuristic achieved 0 queuing times. Regarding computational times, the overall budget time consumed by the column generation and the integer SPP is $445.2 - 220.5 = 224.7$ seconds, while imposing a time limit when solving the the integer SPP to 300 seconds.

4.4 Impact of time slot width on overall results

We investigate in this section the impact of varying the width of the time slots on the overall performance of the mathheuristic. As explained in the description section 2, the duration of loading and unloading operations is expressed as an integer multiplier μ of the width of time slots δ. We conducted experiments where we vary the δ, and consequently μ, while ensuring that $\mu \times \delta$ is always equal to the average duration of loading/unloading operations, that is, 1200 seconds.

Table 6: Impact of time slot variations on the overall results

	60	300	600	1200
$\overline{Dur.}$	855.3	863.9	860.1	866.7
$\overline{QT.}$	2.06	1.29	0.27	0
$\overline{Idle}$	2.97	14.79	30.57	59.2
GAP	2.72	2.31	1.10	0.08
CPU	2982	2448	1987	745

Table 6 summarizes the obtained results. We vary the δ from 60 seconds to 1200 seconds. We allocated 2000 seconds to solve the integer SPP. We observe that the total duration increases when increasing the width of time slots going from 855.3 hours up to 866.7 hours. However, queuing times decrease when the width of time slots decreases, from 2.06 when $\delta = 60$ seconds hours to almost 0 hours when $\delta = 1200$ seconds. In the same time, idle times evolve from an average of 2.72 hours and reaches up to 59.2 hours. The reduction of idle times is a direct consequence of reducing the width of time slots, and also part of this idle time is captured by queuing times, which

explains the increase in queuing times when reducing the width of time slots. We also notice that the overall gap to optimality increases when reducing the width of time slots, reaching up to 2.72% when $\delta = 60$ seconds, whereas it has only a value of 0.08% when $\delta = 1200$ seconds. Computational times also substantially increase when reducing the width of time slots, going from 745 seconds when $\delta = 1200$ seconds, and reach up to 2982 seconds when reducing the width of time slots to $\delta = 60$ seconds.

5 Conclusion and perspective

We presented in this paper a mathheuristic approach to solve the TTVRPQ. The results showed that combining a rolling horizon heuristic and a column generation based post-optimization phase can yield very good results. Moreover, the use of fast heuristics instead a ESPPRC during the pricing problem in column generation proved to efficient, first by finding new columns with negative reduced costs, and second, by providing a multitude of new columns (not only the columns with the best reduced costs), which appears to be more suitable to our approach, since we only solve the column generation in the root node, as part of a sub-optimal branch-and-price approach. In addition to its ease of implementation, the proposed approach can be generalized to a large number of vehicle routing problems with queuing considerations. Experimental tests showed that the proposed approach achieved a major goal when tackling the TTVRPQ, that is zero queuing times, and succeeded to minimize total duration of routes. We investigated in the computational tests several aspects of the solution approach, mainly the contribution of the column generation phase to the overall results, and we compared the use of an extended formulation against a compact one. Then, we compared the results obtained by the matheuristic approach and those obtained by a rolling horizon heuristic inspired from the literature.

Aknowledgement

The authors would like to thank the FORAC research consortium (Université Laval) for funding this research.

References

[1] Jean-François Audy, Mikael Rönnqvist, Sophie D'Amours, and Ala-Eddine Yahiaoui. Planning methods and decision support systems in vehicle routing problems for timber transportation: a review. *International Journal of Forest Engineering*, pages 1–25, 2022.

[2] Jean-François Audy, Nizar El Hachemi, Laurent Michel, and Louis-Martin Rousseau. Solving a combined routing and scheduling problem in forestry. In *International Conference on Industrial Engineering and Systems Management, May*, pages 25–27, 2011.

[3] Maximiliano Ramon Bordon, Jorge Marcelo Montagna, and Gabriela Corsano. Mixed integer linear programming approaches for solving the raw material allocation, routing and scheduling problems in the forest industry. 2020.

[4] Dick Carlsson and Mikael Rönnqvist. Backhauling in forest transportation: models, methods, and practical usage. *Canadian Journal of Forest Research*, 37(12):2612–2623, 2007.

[5] Nizar El Hachemi, Issmail El Hallaoui, Michel Gendreau, and Louis-Martin Rousseau. Flow-based integer linear programs to solve the weekly log-truck scheduling problem. *Annals of Operations Research*, 232(1):87–97, 2015.

[6] Nizar El Hachemi, Michel Gendreau, and Louis-Martin Rousseau. A hybrid constraint programming approach to the log-truck scheduling problem. *Annals of Operations Research*, 184(1):163–178, 2011.

[7] Nizar El Hachemi, Michel Gendreau, and Louis-Martin Rousseau. A heuristic to solve the synchronized log-truck scheduling problem. *Computers & Operations Research*, 40(3):666–673, 2013.

[8] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[9] Patrick Hirsch and Manfred Gronalt. The timber transport order smoothing problem as part of the three-stage planning approach for round timber transport. *Journal of Applied Operational Research*, 5(2):70–81, 2013.

[10] Gerard AP Kindervater and Martin WP Savelsbergh. 10. vehicle routing: handling edge exchanges. In *Local search in combinatorial optimization*, pages 337–360. Princeton University Press, 2018.

[11] Krishna Teja Malladi and Taraneh Sowlati. Optimization of operational level transportation planning in forestry: a review. *International Journal of Forest Engineering*, 28(3):198–210, 2017.

[12] Myrna Palmgren, Mikael Rönnqvist, and Peter Värbrand. A near-exact method for solving the log-truck scheduling problem. *International Transactions in Operational Research*, 11(4):447–464, 2004.

[13] Pablo A Rey, Juan Andrés Muñoz, and Andrés Weintraub. A column generation model for truck routing in the chilean forest industry. *INFOR: Information Systems and Operational Research*, 47(3):215–221, 2009.

[14] Gregory Rix, Louis-Martin Rousseau, and Gilles Pesant. A column generation algorithm for tactical timber transportation planning. *Journal of the Operational Research Society*, 66(2):278–287, 2015.

[15] Andres Weintraub, Rafael Epstein, Ramiro Morales, Jorge Seron, and Pier Traverso. A truck scheduling system improves efficiency in the forest industries. *Interfaces*, 26(4):1–12, 1996.

[16] Ala-Eddine Yahiaoui, Aziz Moukrim, and Mehdi Serairi. Grasp-ils and set cover hybrid heuristic for the synchronized team orienteering problem with time windows. *International Transactions in Operational Research*, 30(2):946–969, 2023.