Lower and upper bounds for the joint batching, routing and sequencing problem

Olivier Briant, Hadrien Cambazard, Diego Cattaruzza, Nicolas Catusse, Anne-Laure Ladier, Maxime Ogier

Lower and upper bounds for the joint batching, routing and sequencing problem

Olivier Briant[1], Hadrien Cambazard[1], Diego Cattaruzza[2], Nicolas Catusse[1], Anne-Laure Ladier[3], and Maxime Ogier[2]

[1]CNRS, Grenoble INP, G-SCOP, University of Grenoble Alpes, Grenoble 38000, France
[2]Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2]Univ Lyon, INSA Lyon, DISP, EA 4570, Villeurbanne F-69100, France
{olivier.briant, hadrien.cambazard, nicolas.catusse}@grenoble-inp.fr
{diego.cattaruzza, maxime.ogier}@centralelille.fr
anne-laure.ladier@insa-lyon.fr

June 29, 2023

Abstract

Warehouses are the scene of complex logistic problems integrating different decision layers. This paper addresses the Joint Order Batching, Picker Routing and Sequencing Problem with Deadlines (JOBPRSP-D) in rectangular warehouses. To tackle the problem an exponential linear programming formulation is proposed. It is solved with a column generation heuristic able to provide valid lower and upper bounds on the optimal value. We start by showing that the JOBPRSP-D is related to the bin packing problem rather than the scheduling problem. We take advantage of this aspect to derive a number of valid inequalities that enhance the resolution of the master problem. The proposed algorithm is evaluated on publicly available data-sets. It is able to optimally solve instances with up to 18 orders in few minutes. It is also able to prove optimality or to provide high-quality lower bounds on larger instances with 100 orders. To the best of our knowledge this is the first paper that provides optimality guarantee on large size instances for the JOBPRSP-D, the results can therefore be used to assert the quality of heuristics proposed for the same problem.

Keywords: picker sequencing; order batching; picker routing; bin packing; column generation.

1 Introduction

The scene takes place in a tall and dim building, lights flickering with a buzzing noise. Pickers move across the warehouse, collecting items, fulfilling customer's orders, with a digital voice for companionship.

To increase the efficiency of the operations, several orders might be batched, i.e., put together to be collected by a picker in a single route. Batching is possible as long as orders grouped

together respect capacity constraints, namely they fit in a trolley. Since customer satisfaction is at the core of competitiveness in e-commerce, each order is required to be prepared and shipped before a strict deadline that usually corresponds to the departure of a truck from the warehouse to deliver orders. Pickers must therefore collect their batches in an appropriate sequence in such a way that all orders meet their deadlines. Planning the entire process is referred to as the Joint Order Batching, Picker Routing and Sequencing Problem with Deadlines (JOBPRSP-D). In the present work, we consider optimal routing policies in a rectangular warehouse with several blocks (or cross-aisles). Such optimal routes can be efficiently computed with dynamic programming following the work of Ratliff and Rosenthal (1983); Pansart et al. (2018).

Order picking in warehouses is part of a complex logistic process involving a wide range of problems such as storage assignment, order batching, picker routing and picker scheduling (de Koster et al. (2007)). Relatively efficient methods have been designed for single problems such as the picker routing problem. However integrated problems are receiving attention since significant gains can be achieved by taking into account the interrelations of the planning problems (van Gils et al. (2018)). The resulting combinatorial problems embed many NP-Hard subproblems with distinct decision levels which make them very challenging to solve in practice. Moreover, practical instances tend to be of large size which complicates even more the resolution. Consequently, heuristic approaches have been favored so far and, as mentioned in the review of van Gils et al. (2018) (page 6), *the use of mathematical programming as a research method to integrate different order picking planning problems is limited*.

Briant et al. (2020) propose a column generation heuristic able to produce lower and upper bounds for the Joint Order Batching and Picker Routing Problem (JOBPRP). In this work, we show that the methodology proposed by Briant et al. (2020), can be extended to tackle the sequencing question as well *i.e.,* the additional assignment and sequencing of batches to each picker in order to meet the deadlines of the orders. This aspect is currently seen as a scheduling problem in the literature (van Gils et al. (2019); Haouassi et al. (2022)). We take a different view-point and propose a *bin packing* formulation. As a result, we benefit from families of valid inequalities identified for packing problems and derived from Dual-Feasible Functions (see Clautiaux et al. (2010)). A high-quality lower bound for the integrated problem can therefore be derived by adding such cuts in the formulation proposed in Briant et al. (2020). The bin packing view-point also provides new insights to design efficient heuristics. Finally, a key cutting plane that was used for solving the pricing of Briant et al. (2020) can be considerably strengthened by using timing considerations related to the sequencing aspect of the problem.

The contribution of the present paper are as follows:

- an extended mathematical formulation based on bin packing considerations is proposed for the JOBPRSP-D ;
- solving this formulation yields proven optimal solutions on all the small benchmark instances for the JOBPRSP-D;
- families of valid inequalities for the JOBPRSP-D are derived from Dual-Feasible Functions;
- tour constraints proposed in Briant et al. (2020) are strengthened based on the particular structure of the benchmark of instances for the JOBPRSP-D;

- the column generation based heuristic proposed is able to provide high-quality upper and lower bounds for subsets of the benchmark instances.

The paper is organized as follows. The problem specification and notations are given in Section 2. A literature review is presented in Section 3. Formulations and valid inequalities are presented in Section 4, this is the key section on the paper. A reader who is pressed for time can focus on Section 4. Each component of the proposed methodology as well as the overall algorithm design are then detailed in Section 5. Experimental results are reported and discussed in Section 6. Finally Section 7 concludes the paper.

2 Problem definition and notation

The warehouse layout is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ contains two depots s and s' and two types of vertices: locations $\mathcal{V}_L$ and intersections $\mathcal{V}_I$. So, overall $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_I \cup \{s, s'\}$. Each location in the set $\mathcal{V}_L$ contains one or more products to be picked, whereas the intersections in $\mathcal{V}_I$ are used to encode the warehouse structure. Additionally, s denotes the depot where picking routes start and s' is the depot where routes end and picked orders are dropped off. Moreover, a travelling time t_{ij} is associated with each arc $(i, j) \in \mathcal{A}$.

The set of items is denoted by $\mathcal{I}$. An item $i \in \mathcal{I}$ is characterized by its location $v_i \in \mathcal{V}_L$. Note that here we consider an item has a unique location.

A set $\mathcal{O}$ of orders to collect is given. Each order $o \in \mathcal{O}$ is composed of a set $\mathcal{I}_o \subseteq \mathcal{I}$ of items to pick. Let us denote by $B_o = |\mathcal{I}_o|$ the size of order $o \in \mathcal{O}$.

An order $o \in \mathcal{O}$ is associated with a time t_o^{pick} that represents the time needed to pick the items composing order o. It is proportional to the size B_o of order o. We indicate as $\mathcal{V}_o \subseteq \mathcal{V}_L$ the set of locations that have to be visited to retrieve order $o \in \mathcal{O}$. Moreover, an order $o \in \mathcal{O}$ is associated with a deadline $\bar{d}_o > 0$ that represents the latest instant of time to have the order ready at node s'.

The set of deadlines is denoted as $\mathcal{D}$, and in practice the number of different deadlines in $\mathcal{D}$ is much lower than the number of orders since deadlines correspond to the departure of a vehicle from the warehouse to deliver orders. Hence many orders share the same deadline.

A batch is a set of orders that are collected together in a single route by a picker. Each batch is associated to a route k in the warehouse starting from s, terminating in s' and visiting a subset of locations in $\mathcal{V}_L$. The orders collected along the route k are denoted as $\mathcal{O}_k \subseteq \mathcal{O}$. The deadline $\bar{d}_k$ of a route k is the minimum deadline among the deadlines of all its orders, that is $\bar{d}_k = \min_{o \in \mathcal{O}_k} \bar{d}_o$.

The time t_k to perform a route k is the sum of three terms: the picking times of all order to be collected in the route, the travel times to go from each location to the next one, and, finally, a setup time t^{setup}. The setup time is supposed constant and represents the time needed by the picker to get the trolley ready.

The routes are performed by a set $\mathcal{P}$ of pickers, each of whom pushes a trolley of capacity B. A route k must respect the trolley's capacity, that is needs to satisfy $\sum_{o \in \mathcal{O}_k} B_o \leq B$.

A solution of the JOBPRSP-D is a sequence of routes $\mathcal{K}_p$ for each picker $p \in \mathcal{P}$ such that:

- all orders are picked up;

- each route respects the capacity constraint of the trolley;
- in each sequence $\mathcal{K}_p = \{r_p^1, r_p^2, \ldots, r_p^{n_p}\}$, route r_p^{i+1} starts after route r_p^i ends;
- each route arrives at the final depot not later than the minimum deadline of the collected orders.

The objective of the JOBPRSP-D is to minimize the sum of the times to perform the determined routes. Indeed, since the number of available pickers and orders to be collected is an input, minimizing the total time improves the productivity of the system. Moreover, deadlines being considered here as hard constraints, there is no need to consider tardiness aspect in the objective function.

In the following, we will indicate by $\mathcal{K}$ the set of all possible feasible routes. Moreover, given two routes k and $k' \in \mathcal{K}$, we may write $k \subseteq k'$ instead of $\mathcal{O}_k \subseteq \mathcal{O}_{k'}$ for ease of notation. Similarly, given an order $o \in \mathcal{O}$ and a route $k \in \mathcal{K}$, we may write $o \in k$ instead of $o \in \mathcal{O}_k$.

3 Literature review

The literature on problems related to Order Batching and Picker Routing is large and an exhaustive review is out of the scope of the present work. The interested reader is referred to the literature reviews of de Koster et al. (2007), van Gils et al. (2018), Çağla Cergibozan and Tasan (2019), Vanheusden et al. (2022).

We thus limit the review presented in this section to works directly related to this paper. First, we review works on the JOBPRP proposing algorithms that are exact or that produce both lower and upper bounds on the studied problem. Then, we review works that consider the sequencing of pickers.

3.1 Solving the JOBPRP exactly or producing lower and upper bounds

Some exact algorithms have been proposed for the JOBPRP but are rarely extended to more complex integrated problems. Algorithmic techniques relevant to the present work have also been investigated on the JOBPRP and are mostly based on exponential Integer Programming (IP) formulations inspired by vehicle routing.

In particular, to the best of our knowledge, Gademann and van de Velde (2005) are the first to design a Branch-and-Price algorithm where variables are related to batches and their corresponding routes in the warehouse. The proposed approach remains limited to the case of a single block warehouse where all orders have the same size. The pricing algorithm is a dedicated Branch-and-Bound algorithm that uses the dynamic program of Ratliff and Rosenthal (1983) to compute the optimal route of a batch. Muter and Öncan (2015) generalize the pricing problem of Gademann and van de Velde (2005) to deal with orders of different sizes, and also consider heuristic routing policies. Valle et al. (2017) propose a Branch-and-Cut approach to solve the JOBPRP in a two blocks warehouse and an optimal routing policy. The optimal routing is based on sub-tour elimination constraints typical of the Traveling Salesman Problem formulation of Dantzig. The algorithm proposed is able to optimally solve instances with 20 orders. The dynamic programming algorithm of Ratliff and Rosenthal (1983) is then generalized to any number

of cross-aisles and used by Briant et al. (2020) to design another column generation technique. The pricing is solved with integer programming and the addition of cutting planes based on the dynamic programs for computing optimal routes. Recently, Heßler and Irnich (2022) propose to extend the state space procedure of Ratliff and Rosenthal (1983) to formulate the pricing problem as a shortest path problem with side constraints, namely the grouping requirement of orders and the trolley capacity. It also naturally handles scattered storage *i.e.* an item may be stored at more than one pick location. The pricing problem can then be efficiently solved with an IP model. The proposed algorithm remains limited to a one block warehouse. Finally, Wahlen and Gschwind (2023) propose a Branch-and-Price-and-Cut algorithm to solve the JOBPRP. The pricing problem is formulated as a Shortest Path Problem with Resource Constraints (SPPRC) on a linear graph where the nodes represent the customer orders in a given sequence, and two arcs between two consecutive nodes represent the inclusion or exclusion of the order in the batch. The SPPRC is solved by a labelling algorithm with strong completion bounds ; the results on instances with one block warehouses outperforms the results of Muter and Öncan (2015).

3.2 Problems with picker sequencing considerations

The sequencing of pickers, which is at the heart of the present paper, is closely related to the wave-based picking issue. Picking is often performed in the so-called wave-based manner where a group of orders (a wave) is released at the same time and needs to be picked. The goal is to minimize the makespan of the wave which requires to assign batches to pickers. This is a natural extension of the JOBPRP where the exact sequence of each picker is not needed since the makespan is only dependent on the assignment of batches to the pickers and not on their sequencing. Wave picking problems with makespan minimisation are addressed by Ardjmand et al. (2018, 2020); Muter and Öncan (2022).

Sequencing really matters when due dates or deadlines are considered for orders. A number of variants of this integrated batching, routing and sequencing problem are addressed in the literature ; they vary on the objective and in the way they consider the due dates either as hard or soft constraints. Table 1 provides an overview of the related literature. Columns `time` and `tardiness` indicate if the objective function considers to minimize the time and the tardiness respectively. Column `Due date constraints` indicates if due dates are considered as hard or soft : note that a hard due date corresponds to a deadline. Finally, column `Solution method` indicates the proposed solution methods.

Reference	Objective		Due date	Solution method
	time	tardiness	constraints	
Tsai et al. (2008)	✓	✓	soft	genetic algorithm
Henn and Schmid (2013)		✓	soft	iterated local search tabu search
Henn (2015)		✓	soft	variable neighborhood descent variable neighborhood search
Chen et al. (2015)		✓	soft	genetic algorithm ant colony
Scholz et al. (2017)		✓	soft	variable neighborhood descent
Menéndez et al. (2017)		✓	soft	variable neighborhood search
van Gils et al. (2019)	✓		hard	iterated local search
Haouassi et al. (2022)	✓		hard	heuristic + constraint programming
D'Haen et al. (2022)	✓	✓	soft	large neighborhood search

Table 1: Studies of variants of the integrated batching, routing and sequencing problem.

Table 1 shows that several works focus on the sum of the delays to the due dates whereas others consider them as deadlines *i.e* as hard constraints, and then focus on the minimization of the total time. Also, two papers consider the minimization of a weighted sum of tardiness and total time. Additionally, note that two real life aspects of the problem are taken into account in recent works: dynamic arrival of orders (D'Haen et al. (2022)) and scattered storage (Haouassi et al. (2022)). All the algorithms proposed so far are based on metaheuristics such as genetic algorithm, iterated local search, tabu search, variable neighborhood search, ant colony, variable neighborhood descent, large neighbourhood search. Note that Haouassi et al. (2022) use a constraint programming approach to efficiently build a feasible schedule for each picker. Henn (2015) mentions column generation as a possible resolution approach but turns to a heuristic. To the best of our knowledge, none of the algorithms proposed in the literature is exact or computes lower bounds for this integrated problem.

The algorithm proposed in this paper is meant to close this gap and contribute in assessing the numerous heuristics developed in this field. Moreover, order deadlines are a relevant set-up in real-life warehouse used by van Gils et al. (2019); Haouassi et al. (2022) and a shared benchmark is available.

4 Formulation, relaxation and valid inequalities for the JOBPRSP-D

In this section we first provide an extended formulation of the JOBPRSP-D as a (multiple) bin packing problem. Let us first recall that given a route $k \in \mathcal{K}$, t_k represents its total time, $\mathcal{O}_k \subseteq \mathcal{O}$ the batch of orders to be collected while performing route k and $\bar{d}_k = \min_{o \in \mathcal{O}_k} \bar{d}_o$ its deadline. Additionally, to easily state the model, we denote as:

- $\mathcal{K}(o) = \{k \in \mathcal{K} : o \in \mathcal{O}_k\}$ the set of routes retrieving order o, for all orders $o \in \mathcal{O}$;

- $\mathcal{K}(\bar{d}) = \{k \in \mathcal{K} : \bar{d}_k \leq \bar{d}\}$ the set of routes with a deadline that is lower than or equal to $\bar{d}$, for all $\bar{d} \in \mathcal{D}$.

We use binary variables ρ_{kp} that equal 1 if route $k \in \mathcal{K}$ is assigned to picker $p \in \mathcal{P}$, 0 otherwise. The JOBPRSP-D can be formulated as follows:

$$
(B(\mathcal{K}))\begin{cases}
\min & \displaystyle\sum_{k\in\mathcal{K}} t_k \sum_{p\in\mathcal{P}} \rho_{kp} & & (1)\\
s.t. & \displaystyle\sum_{k\in\mathcal{K}(o)} \sum_{p\in\mathcal{P}} \rho_{kp} \geq 1 & \forall o \in \mathcal{O} & (2)\\
& \displaystyle\sum_{k\in\mathcal{K}(\bar{d})} t_k\, \rho_{kp} \leq \bar{d} & \forall \bar{d} \in \mathcal{D}, \forall p \in \mathcal{P} & (3)\\
& \rho_{kp} \in \{0,1\} & \forall k \in \mathcal{K}, p \in \mathcal{P} & (4)
\end{cases}
$$

The objective function (1) minimizes the total time required by the pickers to retrieve all orders. Constraints (2) ensure that each order is collected. From a bin packing point of view, those constraints state that among the routes that compose the final packing there is at least one that contains a given order $o \in \mathcal{O}$.

Constraints (3) impose the respect of the deadlines. In particular, for each deadline $\bar{d} \in \mathcal{D}$, they impose that the total time of routes in $\mathcal{K}(\bar{d})$ assigned to a picker do not exceed the value of $\bar{d}$. From a bin packing point of view, these constraints impose that the total size (that is the picking time) of the objects (that is the routes) assigned to a bin (that is a picker) do not exceed its capacity (that is the deadline). Again here we emphasize the fact that the sequence in which the picker performs the routes that are assigned to him/her is not important as long as their total time is lower than or equal to the considered deadline. This allows to look at the problem as a (variant of the multiple) bin packing problem.

Note that a valid sequence for each picker is easily computed from an assignment of the routes to the pickers provided by $(B(\mathcal{K}))$ by sorting the routes by increasing deadlines. Moreover, this bin packing formulation of the problem is possible because there are no release dates for the orders, i.e. all orders can be collected from the beginning of the planning horizon.

Finally constraints (4) define the variables.

Note that this extended formulation contains an exponential number of variables with regards to the size of the instance, and solving directly such a formulation implies that all feasible routes have been generated beforehand.

4.1 A relaxation of formulation $B(\mathcal{K})$

Let us now consider the binary variables ρ_k defined as:

$$
\rho_k = \sum_{p\in\mathcal{P}} \rho_{kp} \qquad \forall k \in \mathcal{K} \qquad (5)
$$

Note that ρ_k equals 1 if route $k \in \mathcal{K}$ is selected to be part of the solution, 0 otherwise. By taking into account constraints (5) and aggregating constraints (3) in formulation $(B(\mathcal{K}))$, we obtain the following formulation:

$$
(M(\mathcal{K})) \begin{cases} \min & \displaystyle\sum_{k \in \mathcal{K}} t_k \, \rho_k & & (6) \\ s.t. & \displaystyle\sum_{k \in \mathcal{K}(o)} \rho_k \geq 1 & \forall o \in \mathcal{O} & (7) \\ & \displaystyle\sum_{k \in \mathcal{K}(\bar{d})} t_k \, \rho_k \leq \bar{d} \times |\mathcal{P}| & \forall \bar{d} \in \mathcal{D} & (8) \\ & \rho_k \in \{0,1\} & \forall k \in \mathcal{K} & (9) \end{cases}
$$

The objective function (6) minimizes the total time of the selected routes. Constraints (7) ensure that each order is collected. Constraints (8) are obtained by summing constraints (3) over all pickers $p \in \mathcal{P}$ and taking into consideration the relation defined in (5). Constraints (9) define the variables. Note that $(M(\mathcal{K}))$ is the formulation proposed in Briant et al. (2020) for the JOBPRP with the additional constraints (8).

Proposition 1 *Formulation* $(M(\mathcal{K}))$ *is a relaxation of* $(B(\mathcal{K}))$.

Proof: The proof is trivial and directly follows from the definition of variables $\rho_k, k \in \mathcal{K}$ and the aggregation of constraints (3) to obtain constraints (8). □

Proposition 2 *The linear relaxations of formulations* $(B(\mathcal{K}))$ *and* $(M(\mathcal{K}))$ *are equal.*

Proof: Let us denote by $(LB(\mathcal{K}))$ and $(LM(\mathcal{K}))$ the linear relaxations of formulations $(B(\mathcal{K}))$ and $(M(\mathcal{K}))$ respectively, that is, integrality requirements on the variables (constraints (4) and (9)) are removed. From any fractional solution $\tilde{\rho}_{kp}$ of $(LB(\mathcal{K}))$, a feasible solution to $(LM(\mathcal{K}))$ can be built by setting $\tilde{\rho}_k = \sum_{p \in \mathcal{P}} \tilde{\rho}_{kp}$, that is using relation (5). Reversely, from any solution $\tilde{\rho}_k$ of $(LM(\mathcal{K}))$, we can set $\tilde{\rho}_{kp} = \frac{1}{|P|}\tilde{\rho}_k$ for each picker $p \in P$ to get a feasible solution of $(LB(\mathcal{K}))$. □

Note that constraints (8) can be seen as enforcing in $(M(\mathcal{K}))$ the linear relaxation of the bin packing constraints (3) of $(B(\mathcal{K}))$. This bound is also known as the L_1 lower bound for bin packing. Stronger bounds (such as L_2) can be used to derive stronger inequalities following the framework of Dual-Feasible Functions.

4.2 Valid inequalities for the JOBPRSP-D from Dual-Feasible Functions

In this section we derive valid inequalities for the JOBPRSP-D by using Dual-Feasible Functions (DFF). The concept of DFF has been used to solve optimization problems involving knapsack

constraints, like for example bin packing, cutting stock, scheduling or routing problems. DFF are used either to compute lower bounds or to add valid inequalities in integer programs[1]. The interested reader is referred to Clautiaux et al. (2010) who propose a survey on this topic. In the following, we recall some basic notions about DFF.

Definition 1 *A function $f : [0,1] \to [0,1]$ is dual-feasible if for any finite set S of real non-negative numbers, we have the relation*

$$\sum_{x \in S} x \leq 1 \;\; \Rightarrow \;\; \sum_{x \in S} f(x) \leq 1.$$

From any Dual-Feasible Function, the following proposition allows to derive valid inequalities for integer programs.

Proposition 3 *If*

- $f : [0,1] \to [0,1]$ *is a DFF*
- $S = \left\{ x \in \mathbb{Z}_+^n \mid \sum_{j=1}^n a_{ij} x_j \leq b_i, \;\; \forall i = 1, ..., m \right\}$, *with* $b_i \geq a_{ij} \geq 0$, *for all* $i = 1, \ldots, m, j = 1, \ldots, n$ *and* $b_i > 0$ *for all* $i = 1, \ldots, m$

then for any $i = 1, \ldots, m$, $\sum_{j=1}^n f(\frac{a_{ij}}{b_i}) x_j \leq 1$ *is a valid inequality for* S.

Numerous DFF for the bin packing problem have been proposed in the literature. We present in Section 4.2.1 the so-called Martello and Toth's cuts and in Section 4.2.2 the so-called Fekete and Schepers's cuts.

4.2.1 Martello and Toth's cuts

As explained above, each deadline in the definition of the JOBPRSP-D can be seen as the size of a bin from a bin packing problem point of view. Thus, given a deadline $\bar{d} \in \mathcal{D}$, we can consider the L_2 bound for the related bin packing problem (Martello and Toth (1990)). This provides a family of valid inequalities for formulation $(M(\mathcal{K}))$. Note that similar families of inequalities can be proposed for formulation $B(\mathcal{K})$. Let us denote $\mathcal{Q}(\bar{d}) = \{q \in \mathbb{N} | q \geq 1, q \leq \left\lfloor \frac{1}{2} \bar{d} \right\rfloor\}$. Given a deadline $\bar{d} \in \mathcal{D}$ and an integer $q \in \mathcal{Q}(\bar{d})$, we define the following sets:

- $\mathcal{K}_1(\bar{d}, q) = \left\{ k \in \mathcal{K}(\bar{d}) \mid t_k > \bar{d} - q \right\}$ as the set containing all the routes in $\mathcal{K}(\bar{d})$ which durations are strictly greater than $\bar{d} - q$;
- $\mathcal{K}_2(\bar{d}, q) = \left\{ k \in \mathcal{K}(\bar{d}) \mid q \leq t_k \leq \bar{d} - q \right\}$ as the set containing all the routes in $\mathcal{K}(\bar{d})$ which durations are greater than or equal to q and lower than or equal to $\bar{d} - q$;
- $\mathcal{K}_3(\bar{d}, q) = \left\{ k \in \mathcal{K}(\bar{d}) \mid t_k < q \right\}$ as the set containing all the routes in $\mathcal{K}(\bar{d})$ which durations are strictly lower than q.

[1] A pedagogical video on DFF is available on this page: `https://youtu.be/5S5FOPW5mzA`

Proposition 4 *The following constraints*

$$\sum_{k\in\mathcal{K}_1(\bar{d},q)} \bar{d}\,\rho_k + \sum_{k\in\mathcal{K}_2(\bar{d},q)} t_k\,\rho_k \leq \bar{d}\times|\mathcal{P}| \quad \forall \bar{d}\in\mathcal{D}, q\in\mathcal{Q}(\bar{d}) \qquad (10) \qquad \leftarrow \gamma_{MT}^{\bar{d}q}\leq 0$$

are valid inequalities for $(M(\mathcal{K}))$.

Sketch of the proof: For a detailed proof see Appendix 8.1. The proposition is proven by applying Proposition 3 to Constraints (3) with DFF f_0^λ, $\lambda\in\left[0;\frac{1}{2}\right]$:

$$f_0^\lambda : [0,1]\to[0,1]$$
$$x\mapsto \begin{cases} 1, & \text{if } x > 1-\lambda \\ x, & \text{if } \lambda\leq x\leq 1-\lambda \\ 0, & \text{if } x<\lambda \end{cases}$$

□

The meaning of these valid inequalities is the following. Each picker $p\in\mathcal{P}$ is considered as a bin of size $\bar{d}$. The routes in $\mathcal{K}_1(\bar{d},q)$ of size strictly greater than $\bar{d}-q$ are accounted with size $\bar{d}$. The routes of medium size, that is those in $\mathcal{K}_2(\bar{d},q)$ which lengths are between q and $\bar{d}-q$, are accounted with their normal size t_k. Finally, the routes in $\mathcal{K}_3(\bar{d},q)$ of size strictly lower than q are ignored.

This is valid since a route of size greater than $\bar{d}-q$ and a route of size greater than q cannot be performed by the same picker without exceeding the deadline $\bar{d}$. Inequalities (10) will be called Martello and Toth's cuts in the following of the paper, and we denote by $\gamma_{MT}^{\bar{d}q}\leq 0$ the dual variable associated to the cut with parameters $\bar{d}$ and q.

4.2.2 Fekete and Schepers's cuts

Let us choose a deadline $\bar{d}\in\mathcal{D}$, an integer $q\in\{2,\dots,\bar{d}\}$ and for all integers $i\in\{0,\dots,q-1\}$ let us define the following set:

$$\mathcal{K}(\bar{d},i,q) \;=\; \left\{\, k\in\mathcal{K} : \bar{d}_k\leq\bar{d},\ \text{and } i\,\frac{\bar{d}}{q} < t_k \leq (i+1)\frac{\bar{d}}{q}\right\}$$

Now, let us observe that if we consider the interval $[0,\bar{d}]$ divided in q identical sub-intervals of length $\frac{\bar{d}}{q}$ and we select a route $k\in\mathcal{K}(\bar{d},i,q)$, it entirely covers i intervals of $[0,\bar{d}]$, plus a strictly non-empty part of the $(i+1)$-th interval since, by definition of $\mathcal{K}(\bar{d},i,q)$ it holds $t_k - i\frac{\bar{d}}{q} > 0$.

It follows that given a deadline $\bar{d}\in\mathcal{D}$ and an integer $q\in\{2,\dots,\bar{d}\}$, we have at most $q-1$ equally sized intervals to place potential routes in $[0,\bar{d}]$, since the last interval is dedicated to allocate the residuals. As this reasoning is valid for each picker $p\in\mathcal{P}$, it thus follows the following proposition:

Proposition 5 *The following inequalities*

$$\sum_{i=1}^{q-1} \sum_{k \in \mathcal{K}(\bar{d},i,q)} i\, \rho_k \leq (q-1) \times |\mathcal{P}| \quad \forall \bar{d} \in \mathcal{D}, q \in \{2, \ldots, \bar{d}\} \quad (11) \qquad \leftarrow \gamma_{FS}^{\bar{d}q} \leq 0$$

are valid for $(M(\mathcal{K}))$.

Sketch of the proof: For a detailed proof see Appendix 8.2. The proposition is proven by applying Proposition 3 to constraints (3) with the DFF $g^\lambda(x)$, $\lambda \in \mathbb{N} \setminus \{0\}$:

$$g^\lambda(x) = \begin{cases} 0 & \text{if } x = 0 \\ \frac{x(\lambda+1)-1}{\lambda} & \text{if } x(\lambda+1) \in \mathbb{Z} \setminus \{0\} \\ \lfloor (\lambda+1)x \rfloor \frac{1}{\lambda}, & \text{otherwise} \end{cases}$$

□

It can be shown that the DFF $g^\lambda(x)$ is a non-maximal DFF, meaning that a stronger cut may be obtained. This can be done by considering the DFF proposed in Fekete and Schepers (2001). However, we choose to consider the weaker version since it makes the integration into the pricing problem easier as will be detailed in Section 5.2.4. Inequalities (11) will be called Fekete and Schepers's cuts in the following of the paper, and we denote by $\gamma_{FS}^{\bar{d}q} \leq 0$ the dual variable associated to the cut with parameters $\bar{d}$ and q.

4.3 Valid inequalities for the JOBPRP

The problem described by formulation $(M(\mathcal{K}))$ is an extension of the JOBPRP, where packing constraints, that is constraints (8), are considered. Hence, the valid inequalities for the JOBPRP are also valid for the JOBPRSP-D and are thus valid for $(M(\mathcal{K}))$.

4.3.1 Strengthened capacity cuts

The strengthened capacity cuts (SCC) play with the size of the orders and the capacity of the trolley to impose the minimum number of routes needed to retrieve a given set of orders. They are defined as follows:

$$\sum_{k \in \mathcal{K} | \mathcal{O}_k \cap \mathcal{R} \neq \emptyset} \rho_k \geq \left\lceil \frac{\sum_{o \in \mathcal{R}} B_o}{B} \right\rceil \quad \forall \mathcal{R} \subseteq \mathcal{O} \qquad (12) \qquad \leftarrow \gamma_{SC}^{\mathcal{R}} \geq 0$$

This family of cuts has been first proposed by Baldacci et al. (2008) for the Capacitated Vehicle Routing Problem (CVRP), and has also been used by Briant et al. (2020) and Wahlen and Gschwind (2023) in the context of the JOBPRP. Note that if there exists a set $\mathcal{R}' \subset \mathcal{R}$ such that $\left\lceil \frac{\sum_{o \in \mathcal{R}'} B_o}{B} \right\rceil = \left\lceil \frac{\sum_{o \in \mathcal{R}} B_o}{B} \right\rceil$, then the SCC defined over $\mathcal{R}'$ dominates the one defined over $\mathcal{R}$, and the latter does not need to be considered. When $\mathcal{R} = \mathcal{O}$, the corresponding SCC states a constraint on the minimum number of routes required in any solution.

4.3.2 Rank-1 cuts

The Chvátal–Gomory rank-1 cuts (R1C) are obtained by considering a small subset of the rows defining the set covering inequalities, i.e. constraints (7). Given a subset of orders $\mathcal{R} = \{o_1, o_2, \ldots, o_q\} \subseteq \mathcal{O}$, with $q \geq 3$ and non-negative multipliers $\mathbf{p} = \{p_1, p_2, \ldots, p_q\}$, R1C are defined as:

$$\sum_{k \in \mathcal{K}} \left\lfloor \sum_{o \in \mathcal{R} \cap \mathcal{O}_k} p_o \right\rfloor \rho_k \leq \left\lfloor \sum_{o \in \mathcal{R}} p_o \right\rfloor \quad \forall \mathcal{R} \subseteq \mathcal{O}, \mathbf{p} \in \mathbb{R}_+^q \qquad (13) \qquad \leftarrow \gamma_{R1}^{\mathcal{R}\mathbf{p}} \leq 0$$

The constraints that are obtained in the particular case when all the multipliers have the same value $p_o = \frac{1}{h}$ for a given integer $h \in \{1, \ldots, |\mathcal{R}|\}$ are known as subset-row inequalities. They have been first proposed by Jepsen et al. (2008), and are used in the state-of-the-art exact algorithms for vehicle routing problems as they provide a significant improvement on the lower bound (Pecin et al., 2017a). R1Cs have been used by Muter and Öncan (2015) and Heßler and Irnich (2022) to solve the JOBPRP.

Note that Pecin et al. (2017b) performed a computational polyhedral study to determine the best possible vectors of multipliers $\mathbf{p}$ for cuts with 3 to 5 rows. For example, when $|\mathcal{R}| = 3$, there is a single optimal multiplier that is $\mathbf{p} = \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right\}$. The corresponding R1C is then the following:

$$\sum_{k \in \mathcal{K}: |\mathcal{O}_k \cap \mathcal{R}| \geq 2} \rho_k \leq 1 \quad \forall \mathcal{R} \subseteq \mathcal{O}, |\mathcal{R}| = 3 \qquad (14)$$

5 A column generation based heuristic to solve the JOBPRSP-D

In this section, we present the algorithm that we designed to tackle the JOBPRSP-D. It is based on the one proposed in Briant et al. (2020) to solve the JOBPRP, and adapted to tackle the sequencing aspect of the JOBPRSP-D. Briefly, it solves the linear relaxation ($LM(\mathcal{K})$) of formulation ($M(\mathcal{K})$) via column generation. ($LM(\mathcal{K})$) is thus the so-called master problem and details on it are provided in Section 5.1. An overview of the complete algorithm is provided in Algorithm 1.

First, a pool $\mathcal{K}_{pool}$ of routes is generated and an initial solution is determined (Section 5.6). Then, an initial set of routes $\mathcal{K}' \subseteq \mathcal{K}$ over which the master problem is defined to obtain the restricted master problem ($LM(\mathcal{K}')$) is generated (Section 5.7).

The restricted master problem ($LM(\mathcal{K}')$) is then solved and, given the dual information, new negative reduced cost routes are included in $\mathcal{K}'$, first by scanning the pool $\mathcal{K}_{pool}$, and then by solving the pricing problem ($Pr(\mathcal{K}')$) (Section 5.2) if no negative reduced cost route has been found in the pool. From the most negative reduced cost column found either in the pool or by the pricing problem, a rich column set $\mathcal{K}_{rich}$ is also included in $\mathcal{K}'$ (Section 5.3).

After each iteration of the pricing problem, the Lagrangian bound is computed and the value of the lower bound LB is possibly updated (Section 5.5). Moreover, in the hope of improving the upper bound, some columns of the rich column set are generated to provide a feasible solution for ($B(\mathcal{K})$).

To strengthen the lower bound provided by the resolution of formulation $(LM(\mathcal{K}))$, we consider the valid inequalities presented in the previous section: the Martello and Toth's cuts (10), the Fekete and Schepers's cuts (11), the strengthened capacity cuts (12) and the rank-1 cuts (13). Their separation is detailed in Section 5.4.

The procedure continues until $(LM(\mathcal{K}))$ is optimally solved or the time limit is reached. A final resolution of formulation $(\mathcal{B}(\mathcal{K}'))$ defined over the set of routes $\mathcal{K}'$ generated so far is launched with a time limit τ_l. This column generation based heuristic procedure presented in this section is called CGH in the following.

Since the proposed algorithm is based on the one proposed in Briant et al. (2020), the details about the solving of the pricing problem, the rich column set and the Lagrangian bound are not reported here. This choice has been made to concentrate on and highlight the elements that allow to extend the algorithm of Briant et al. (2020) to solve the JOBPRSP-D. In particular, the main contributions are:

- a Mixed Integer Program (MIP) formulation of the pricing problem to take into account the bin packing constraints (8) (Section 5.2.1);
- the strengthening of the tour constraints used in the MIP formulation of the pricing problem (Section 5.2.2);
- the impact of the Martello and Toth's cuts (10), and Fekete and Schepers's cuts (11) on the MIP formulation of the pricing problem (Section 5.2.4), and their separation (Section 5.4);
- the determination of an initial solution for the JOBPRSP-D and a pool of columns (Section 5.6).

Moreover, note that contrary to Briant et al. (2020) no stabilization is used in the proposed algorithm. Indeed, preliminary experimental results showed that adding stabilization did not permit to improve the quality of the results.

The next sections present in details each component of the procedure.

5.1 Master problem

The master problem that we solve across column generation is the linear relaxation of formulation $(M(\mathcal{K}))$, denoted as $(LM(\mathcal{K}))$, where the integrality requirements on the variables $\rho_k, k \in \mathcal{K}$ are

Algorithm 1 Column generation heuristic (CGH) pseudo code

```
Computation of pool of routes K_pool (Section 5.6)
UB ← value of the best solution found by the initial heuristic (Section 5.6)
Initialization of K' (Section 5.7)
LB ← 0
do
    while (LM) is not optimal and time limit is not reached do
        Solve (LM(K')) with columns in K' (Section 5.1)
        if K_pool contains columns with negative reduced cost then
            K' ← K' ∪ {negative reduced cost columns of K_pool}
        else
            Solve pricing Pr(K') (Section 5.2)
            K' ← K' ∪ {negative reduced cost columns found by Pr(K')}
            LB ← max{LB, Lagrangian bound} (Section 5.5)
        end if
        Compute K_rich from the most negative reduced cost column (Section 5.3)
        K' ← K' ∪ K_rich
        UB̄ ← Seek for a primal solution from K_rich
        if UB̄ < UB then
            UB ← UB̄
        end if
    end while
    Separate constraints (10), (11), (12) and (13) (Section 5.4)
while time limit not reached and at least one (10), (11), (12), (13) is violated
UB ← Solve (B(K'))
```

removed. The master problem $(LM(\mathcal{K}))$ reads as follows:

$$
(LM(\mathcal{K})) \begin{cases}
\min & \displaystyle\sum_{k\in\mathcal{K}} t_k\ \rho_k & & & (15) & \\
s.t. & \displaystyle\sum_{k\in\mathcal{K}(o)} \rho_k\ \geq\ 1 & \forall o\in\mathcal{O} & (16) & \leftarrow\ \alpha^o \geq 0 \\
& \displaystyle\sum_{k\in\mathcal{K}(\bar{d})} t_k\ \rho_k\ \leq\ \bar{d}\times|\mathcal{P}| & \forall \bar{d}\in\mathcal{D} & (17) & \leftarrow\ \beta^{\bar{d}} \leq 0 \\
& \rho_k\ \geq 0 & \forall k\in\mathcal{K} & (18) &
\end{cases}
$$

The dual variables associated with each family of constraint are reported next to the expression. Usually, the master problem is solved over a restricted set of routes $\mathcal{K}' \subseteq \mathcal{K}$. We thus refer to $(LM(\mathcal{K}'))$ as the restricted master problem, as it is usually done in the literature.

5.2 Pricing problem

In practice, the master problem $(LM(\mathcal{K}))$ is solved on a subset $\mathcal{K}^{'} \subseteq \mathcal{K}$ of the entire set of routes. The role of the pricing problem is to identify a new variable among those in $\mathcal{K} \setminus \mathcal{K}^{'}$ of negative

reduced cost, or to prove that none exists. The reduced cost $\bar{c}_k$ of variable $\rho_k \in \mathcal{K}$ is computed as follows:

$$\bar{c}_k \;=\; t_k \;-\; \sum_{o \in \mathcal{O}} \alpha^o \, \mathbb{1}_{\mathcal{K}(o)}(k) \;-\; \sum_{\bar{d} \in \mathcal{D}} \beta^{\bar{d}} \, t_k \, \mathbb{1}_{\mathcal{K}(\bar{d})}(k)$$

In the rest of the section, we first formulate the pricing problem as a MIP presented in Section 5.2.1. Sections 5.2.2 and 5.2.3 present the strengthening of the tour constraints of Briant et al. (2020) that are dynamically included in the MIP of the pricing. Section 5.2.4 discusses the impact of the valid inequalities on the modelling and resolution of the pricing problem.

5.2.1 A MIP formulation for the solving of the pricing problem

The solving of the pricing problem is performed as in Briant et al. (2020), i.e. it is a cutting plane algorithm based on a MIP formulation for the pricing problem. In this section we provide details on the MIP formulation of the pricing problem. Details about the solving of the pricing can be found in Section 5.3. For sake of simplicity, we state the MIP formulation without considering the dual contribution deriving from the valid inequalities (10), (11), (12) and (13) presented in Section 4.2. This topic will be covered apart in Section 5.2.4.

The MIP formulation of the pricing problem presented here extends the one in Briant et al. (2020) by taking into account constraints (17) of the master problem which capture the additional sequencing characteristic of the JOBPRSP-D. The following sets of decision variables are used:

- $t \geq 0$: real non-negative variable that represents a lower bound on the total time of the route (including picking and setup times);
- $\delta^{\bar{d}} \geq 0$: real non-negative variable that equals t if the deadline of the route is less than $\bar{d}$, 0 otherwise;
- $e^o \in \{0, 1\}$: binary variable that equals 1 if order o is in the route, 0 otherwise;
- $\mu^{\bar{d}} \in \{0, 1\}$: binary variable that equals 1 if the deadline of the route equals to $\bar{d}$, 0 otherwise.

We introduce the following notation: $\mathcal{O}^{\bar{d}} = \left\{ o \in \mathcal{O} : \bar{d}_o = \bar{d} \right\}$ is the set of orders with deadline $\bar{d}$.

The pricing is solved by a cutting plane algorithm based on the following MIP model $Pr(\mathcal{K}')$:

$$
Pr(\mathcal{K}^{'}) \begin{cases}
\min \quad t \; - \; \displaystyle\sum_{o \in \mathcal{O}} \alpha^o \, e^o \; - \; \sum_{\bar{d} \in \mathcal{D}} \beta^{\bar{d}} \, \delta^{\bar{d}} & & (19) \\
s.t. \quad \displaystyle\sum_{o \in \mathcal{O}} B_o e_o \leq B & & (20) \\
t_k \left(\displaystyle\sum_{o \in \mathcal{O}_k} e_o - |k| + 1 \right) \leq t & \forall k \in \mathcal{K}' & (21) \\
\displaystyle\sum_{\bar{d} \in \mathcal{D}} \mu^{\bar{d}} \; = \; 1 & & (22) \\
\displaystyle\sum_{o \in \mathcal{O}^{\bar{d}}} e^o \; \geq \; \mu^{\bar{d}} & \forall \bar{d} \in \mathcal{D} & (23) \\
\displaystyle\sum_{\bar{d} \leq \bar{d}_o} \mu^{\bar{d}} \; \geq \; e^o & \forall o \in \mathcal{O} & (24) \\
\delta^{\bar{d}} \; \geq \; t \; - \; \displaystyle\sum_{\bar{d}' > \bar{d}} \bar{d}' \mu^{\bar{d}'} & \forall \bar{d} \in \mathcal{D} & (25) \\
t \; \leq \; \displaystyle\sum_{\bar{d} \in \mathcal{D}} \bar{d} \, \mu^{\bar{d}} & & (26) \\
e^o \; \in \; \{0,1\} & \forall o \in \mathcal{O} & (27) \\
\mu^{\bar{d}} \; \in \; \{0,1\} & \forall \bar{d} \in \mathcal{D} & (28) \\
\delta^{\bar{d}} \; \geq \; 0 & \forall \bar{d} \in \mathcal{D} & (29) \\
t \; \geq \; 0 & & (30)
\end{cases}
$$

The objective function (19) of $Pr(\mathcal{K}^{'})$ minimizes the reduced cost expression. Constraints (20) enforce the respect of the capacity of the trolley. Variable t encodes a lower bound on the travel time of a route. It is strengthened by the dynamic generation of constraints (21) that are called the *tour constraints*. Typically, any selection of orders that is a superset of a known set of orders k must require a total time longer than t_k so that $t \geq t_k$. Constraints (22) state that a single deadline must be selected for the route. Constraints (23) impose that at least one order in the route must have the selected deadline. Constraints (24) impose that the deadline of the route is smaller than the one of any order picked by the route. Note that the two previous constraints (23) and (24) ensure that the deadline of the route is the minimum among the deadlines of all picked orders. Constraints (25) guarantee that the $\delta^{\bar{d}}$ variables are consistent with $\mu^{\bar{d}}$. Constraints (26) impose that the duration of the route is smaller than its deadline to avoid obviously inconsistent routes. Finally, constraints (27)–(30) define the variables.

It is important to notice that solving $Pr(\mathcal{K}')$ will provide a lower bound on the value of the lowest reduced cost route in $\mathcal{K}$, since Constraints (21) are only defined for routes in set $\mathcal{K}'$. When solving $Pr(\mathcal{K}')$, the value of variable t will also represent a lower bound on the exact time

needed to collect all the orders selected by the pricing problem ($e_o = 1$).

5.2.2 Tour constraints and their strengthening

As mentioned in the previous section, to enhance the resolution of the pricing problem, Briant et al. (2020) propose to add to $Pr(\mathcal{K}')$, for each route $k \in \mathcal{K}'$, the following so-called *tour constraint*:

$$t_k \left(\sum_{o \in \mathcal{O}_k} e_o - |k| + 1 \right) \leq t \qquad \forall k \in \mathcal{K}' \qquad (31)$$

We recall that the rationale behind these constraints is the following: any selection of orders that is a superset of a known set of orders k must require a total time longer than t_k so that $t \geq t_k$. Moreover, note that a tour constraint is active only when all variables $e_o, o \in \mathcal{O}_k$ that are involved in it take value 1. Otherwise, the term between brackets is lower than or equal to zero and, obviously, the constraint is inactive.

In the following, we propose to strengthen these constraints by taking into account, in a precise manner, the times that account for the total time t_k of a route. These are the setup time t^{setup}, the picking times t_o^{pick} for each order that is picked, and the travelling times.

To start, let us define generic coefficients $b \geq 0$ and $a_o \geq 0, \forall o \in \mathcal{O}$. Let us also assume that the following constraint is valid:

$$b + \sum_{o \in \mathcal{O}_k} a_o \, e_o \leq t \qquad (32)$$

Let us note

$$\Delta_k = t_k - b - \sum_{o \in \mathcal{O}_k} a_o$$

as the difference between t_k and the left-hand side of constraint (32) evaluated on the route k, i.e. when $e_o = 1$ for all $o \in \mathcal{O}_k$. Note that since t encodes in $Pr(\mathcal{K}')$ a lower bound on the travel time, it holds $t \leq t_k$ where k is the optimal route found by solving $Pr(\mathcal{K}')$. Note that $t = t_k$ is insured if $k \in \mathcal{K}'$. Thus, since constraints (32) are assumed to hold, it follows that $\Delta_k \geq 0$.

The following propositions hold.

Proposition 6 *If constraints* (32) *are valid, then the following constraints*

$$b + \sum_{o \in \mathcal{O}_k} a_o \, e_o \; + \; \Delta_k \left(\sum_{o \in \mathcal{O}_k} e_o - |k| + 1 \right) \leq t \qquad \forall k \in \mathcal{K}' \qquad (33)$$

are valid for $Pr(\mathcal{K}')$ and allow to exactly compute the value of t_k when $e_o = 1$ for all orders $o \in \mathcal{O}_k$.

Proof: If $e_o = 1$ for all $o \in \mathcal{O}_k$, then constraints (33) applied to route k become $t_k \leq t$. Otherwise, we have: $\sum_{o \in \mathcal{O}_k} e_o - |k| + 1 \leq 0$, and the term added to constraints (32) to obtain constraints (33) is non-positive since $\Delta_k \geq 0$. □

Proposition 7 *Constraints* (33) *provide a better estimation than constraints* (31) *of the total time of the route determined by the resolution of the pricing problem* $Pr(\mathcal{K}')$.

Proof: Let us suppose that the solution of $Pr(\mathcal{K}')$ provides a route k^*, and let us consider a route $k \in \mathcal{K}'$. If $k^* \cap k = k$, i.e. k^* is a superset of k, then constraints (31) and (33) both lead to $t_k \leq t$.

If $k^* \cap k \subsetneq k$, thus, the following relation holds:

$$t \quad \geq \quad \underbrace{b + \sum_{o \in \mathcal{O}_{k^*}} a_o \; + \; \Delta_k(|k^* \cap k| - |k| + 1)}_{\text{Constraint (33)}} \quad \geq \quad \underbrace{t_k(|k^* \cap k| - |k| + 1).}_{\text{Constraint (31)}}$$

Indeed, $b + \sum_{o \in \mathcal{O}_{k^*}} a_o \geq 0$, $\Delta_k \leq t_k$ and $(|k^* \cap k| - |k| + 1) \leq 0$. Then t takes a value that is closer to t_{k^*} when using constraints (33) instead of constraints (31). □

Note that it is possible to generalize constraints (33) to consider all orders $o \in \mathcal{O}$. To this end, let us consider a route k and let us assume that the following constraint

$$b + \sum_{o \in \mathcal{O}} a_o \, e_o \leq t \qquad (34)$$

is valid and let us consider coefficients a_o such that it holds the following condition:

$$t_k + \sum_{o \in \mathcal{O}_{k'} \setminus \mathcal{O}_k} a_o \leq t_{k'} \quad \forall k' \supseteq k \qquad (35)$$

We thus state the following proposition.

Proposition 8 *If constraint* (34) *is valid and hypothesis* (35) *holds, then the following constraints*

$$b + \sum_{o \in \mathcal{O}} a_o \, e_o \; + \; \Delta_k \left(\sum_{o \in \mathcal{O}_k} e_o - |k| + 1 \right) \leq t \qquad \forall k \in \mathcal{K}' \qquad (36)$$

are valid for $Pr(\mathcal{K}')$ *and allow to exactly compute* t_k.

Proof: See Appendix 8.3.

Note that since, as can be easily seen, constraints (33) are weaker than constraints (36), we only add the latter to the pricing problem.

5.2.3 Examples of coefficients to be used in constraints (36)

By setting $b = t^{setup}$, and $a_o = t_o^{pick}$ for all $o \in \mathcal{O}$, it is clear that constraint (34) is satisfied, and hypothesis (35) holds. Such coefficients can be easily used in constraint (36). Note that these coefficients use the fact that in the studied problem the duration t_k of a route k is not only related to travelling time, but also to setup and picking times.

It is then possible to provide larger coefficients a_o for orders $o \in k$, by considering the minimum additional traveling time to pick each order $o \in k$. To this end, given a sequence

$(o_1, \ldots, o_{|k|})$ of the orders in route k, we define:

$$a_{o_i} = t^{pick}_{o_i} + \min_{\mathcal{R} \subseteq \{o_1, \ldots, o_{i-1}\}} \left\{ \tilde{t}_{\mathcal{R} \cup o_i} - \tilde{t}_{\mathcal{R}} \right\} \qquad \forall i \in (1, \ldots, |k|);$$

where $\tilde{t}_{\mathcal{R}}$ is the optimal traveling time to collect all orders in $\mathcal{R}$, and $\tilde{t}_{\emptyset} = 0$. Note that we still define $b = t^{setup}$ and $a_o = t^{pick}_o$ for all $o \in \mathcal{O} \setminus k$.

Coefficient a_{o_i} for each order $o_i \in k$ corresponds to the picking time plus the minimum additional travelling time required when adding order o_i to any subset of orders in $\{o_1, \ldots, o_{i-1}\}$. However, note that the computation of each coefficient has a complexity of $O(2^i)$.

In order to avoid large computation times to compute the coefficients when the constraint considers a large number of orders, we propose to compute a lower bound on these coefficients. The latter is based on the computation of an estimation of the travelling time to collect a set of orders and is calculated by considering the rectangular bounding box of the locations to be visited. Details are provided thereafter.

Given a subset of orders $\mathcal{R} \subseteq \mathcal{O}$, let us denote by $\tilde{t}^{box}_{\mathcal{R}}$ the time required to travel the perimeter of the 2-dimensions bounding box of the locations of all items in $\mathcal{R}$ and the depot. It is interesting to note that an optimal route can go outside the bounding box if the item in $\mathcal{R}$ with the largest y coordinate is located in the second half of its block (i.e. nearest from the upper cross aisle of the block). In such a case, an optimal route could use the upper cross-aisle of the upper block. We name *large bounding box* the bounding box of the locations of all items in $\mathcal{R}$ and the depot, enlarged to the upper cross-aisle. We denote by $\tilde{t}^{boxLarge}_{\mathcal{R}}$ the time required to travel the perimeter of the large bounding box. Then, given a sequence $(o_1, \ldots, o_{|k|})$ of the orders in route k, the following coefficients can be used:

$$a_{o_i} = t^{pick}_{o_i} + \max \left\{ 0; \tilde{t}^{box}_{\{o_1, \ldots, o_i\}} - \tilde{t}^{boxLarge}_{\{o_1, \ldots, o_{i-1}\}} \right\}.$$

Coefficient a_{o_i} is equal to the picking time plus a lower bound on the additional travelling time required when adding order o_i to any subset of orders in $\{o_1, \ldots, o_{i-1}\}$.

5.2.4 Impact of valid inequalities on the pricing problem

Column generation-based algorithms are usually used to solve a problem when its mathematical formulation is defined over a set with an exponential amount of variables. For real-size instances, the resulting number of variables does not allow to generate them all a-priori, thus only the interesting ones are dynamically generated. Such formulations are usually called *extended* and can be obtained by applying Dantzig-Wolfe reformulation techniques to formulations for the same problem defined over a polynomial number of variables. Such formulations are usually called *compact*.

Cutting planes added to the extended formulation are called *robust* cuts when they do not increase the complexity of the pricing problem needed to generate interesting variables on the fly. This is usually the case of constraints that can be expressed with the variables of the compact formulation. Opposite to that, when the complexity of the pricing problem is increased by the consideration of a family of cutting planes, they are said *nonrobust*. Hence, each nonrobust

cut that is added to the extended formulation (that is the master problem) directly makes the pricing problem more difficult to be solved. On the opposite, nonrobust cuts are known for their great potential for reducing integrality gaps (Costa et al., 2019).

Note that the four families of cuts proposed in Sections 4.2 and 4.3 are *nonrobust* cuts. When the pricing problem is modelled as an elementary shortest path problem with resource constraints, as it is usually done for routing problems, and solved by dynamic programming via a labelling algorithm, the management of non-robust cuts can be cumbersome (Pecin et al., 2017a). However, when the pricing problem is modelled as a MIP, as is the case of the present work, and solved with a commercial solver, adding nonrobust cuts in the master problem implies adding extra decision variables and constraints in the MIP formulation. Even if this can make the MIP large and increase its resolution time, the coding effort to manage them is rather limited. Each family of cuts therefore requires to update the MIP formulation of the pricing problem and such modifications are detailed in Section 8.4 of the Appendix since it amounts to relatively standard modelling techniques in integer programming.

5.3 Pricing iteration

After the resolution of the restricted master problem ($LM(\mathcal{K}')$), we first check if pool $\mathcal{K}_{pool}$ (see Section 5.6) includes routes with negative reduced cost. If any, they are added to $\mathcal{K}'$. If not, the pricing problem is solved. As explained in Section 5.2, solving $Pr(\mathcal{K}')$ provides a lower bound on the most negative reduced cost column in set $\mathcal{K}$. When solving $Pr(\mathcal{K}')$, each time a feasible route with a negative value is found, such a route can potentially lead to a negative reduced cost column. An efficient dynamic programming algorithm (see Cambazard and Catusse (2018); Pansart et al. (2018)) is then applied on such promising routes to compute their exact total time and check whether they indeed correspond to negative reduced cost routes.

After solving $Pr(\mathcal{K}')$, it is possible that no real negative reduced cost route has been found, while the optimal value of $Pr(\mathcal{K}')$ was negative. In such a case, set $\mathcal{K}'$ is enlarged with the best route found by $Pr(\mathcal{K}')$, and $Pr(\mathcal{K}')$ is solved again. More details on this iterative solving of $Pr(\mathcal{K}')$ can be found in (Briant et al., 2020). Moreover, in this work, $Pr(\mathcal{K}')$ is solved with a time limit τ_{Pr}. When the solving of the pricing problem has identified at least one negative reduced cost route, we add to $\mathcal{K}'$ up to 10 negative reduced cost routes found during the resolution of $Pr(\mathcal{K}')$.

After founding negative reduced cost routes, either in the pool, or by solving $Pr(\mathcal{K}')$, we then determine a set of routes to add in $\mathcal{K}'$ with respect to the route with the most negative reduced cost. This route will be denoted as k^* in the following. In particular, as proposed in Briant et al. (2020), two different strategies are considered to propose other routes to insert in $\mathcal{K}'$: the first strategy is to complete column k^* with a set of columns that potentially constitute a feasible solution of ($B(\mathcal{K})$), and the second strategy is to propose columns similar to k^* with fewer orders. For the first strategy, we proceed as in Briant et al. (2020) to generate a set of promising columns such that each order is collected. However, note that because of the bin-packing constraints (3), there is no guarantee on the feasibility of the solution. Hence, we then solve formulation ($B(\mathcal{K})$) with this set of promising columns to evaluate the feasibility of the solution. If the solution is feasible and improves the current upper bound, then the upper bound

is updated accordingly. For the second strategy, if k^* has less than 8 orders, as in Briant et al. (2020), we generate all sub-tours of k^* that use no more than 75% of the capacity of the trolley. If k^* has 8 orders or more, we generate all sub-tours of k^* with one order less than k^*. All these columns generated by the two strategies compose the rich column set $\mathcal{K}_{rich}$ that is mentioned in Algorithm 1. The reader is referred to Briant et al. (2020) for precise details.

5.4 Separation of the valid inequalities

Note that the Martello and Toth's cuts (10) can be stronger than the relaxation of the bin-packing constraints (8) only if $\mathcal{K}_1(\bar{d}, q)$ contains at least one element k such that $\rho_k^* > 0$, where ρ^* is the optimal solution of the master problem $(LM(\mathcal{K}))$. Note that when q increases, the size of $\mathcal{K}_1(\bar{d}, q)$ increases, but the size of $\mathcal{K}_1(\bar{d}, q) \cup \mathcal{K}_2(\bar{d}, q)$ decreases. So, there is not a value of q that a priori provides the most violated Martello and Toth's cut (10). However, as mentioned in Martello and Toth (1990), it is not necessary to consider all the possible values of q in $\mathcal{Q}(\bar{d})$. The size of $\mathcal{K}_1(\bar{d}, q)$ increases when there exist $k \in \mathcal{K}(\bar{d})$ such that $q = \bar{d} - t_k + 1$. For a given size of $\mathcal{K}_1(\bar{d}, q)$, the most violated Martello and Toth's cut (10) is the one with the lowest q value, i.e. associated with the highest size of $\mathcal{K}_2(\bar{d}, q)$. After all these remarks, for a given $\bar{d}$, we just consider the $q = \bar{d} - t_k + 1$ values such that there is a route k in $\mathcal{K}(\bar{d})$ with $\rho_k^* > 0$ and $t_k \geq \frac{1}{2}\bar{d} + 1$.

For the Fekete and Schepers's cuts (11), given a value of $\bar{d}$, several inequalities can be generated with different values of q. We decided to consider the cut with the highest violation to be included in the model as a preliminary computational study showed that this was the best configuration.

For the SCC cuts (12), as done in Briant et al. (2020), we only consider those characterized by a right-hand side that equals the minimum number of trolleys to retrieve all orders. Note that given the benchmark of instances, it is possible in practice to enumerate all subsets of orders that require this minimum number of trolley to be picked. As a consequence, we do not call a separation algorithm, but scan the set of such subsets to look for violated constraints. Moreover, we only consider the SCC cuts that are minimum for inclusion. The master problem is initialized only with the SCC cuts (12) defined over the entire set of orders. The others are added to the master problem only if they are violated.

For the rank-1 cuts (13), after each resolution of the master problem, we check, in this order, if such constraints defined over subset of orders of size 3, then size 4 are violated. For small size instances (with 18 orders or less), we also check subset of orders of size 5. We stop the procedure when 40 violated constraints are found or when the list has been fully scanned. In any case, we do include in the master problem at most the 20 most violated cuts.

5.5 Computation of a lower bound

When the procedure reaches the time limit before $(LM(\mathcal{K}))$ is solved, we calculate the so-called Lagrangian bound to be able to provide a valid lower bound on the value of the optimal solution. The interested reader is referred to Briant et al. (2020) for a detailed explanation of its computation.

5.6 Initial solution and generation of a pool of routes

Before starting the resolution of the master problem via column generation, we provide an initial solution and generate a set of possible promising routes to include in a pool $\mathcal{K}_{pool}$.

The generation of the routes to include in $\mathcal{K}_{pool}$ starts by first creating a set of routes as follows: the list of orders is swept and the first order not yet considered is used to initialize a route k. The list of orders is then swept again and the order o that does not violate the capacity constraints and minimizes the following score

$$score(o,k) = \max_{l \in \mathcal{V}_o} \{ \min_{\bar{o} \in k, \bar{l} \in \mathcal{V}_{\bar{o}}} \{ t_{l,\bar{l}} \} \}$$

is inserted in the route. Note that the score provides an estimation of the increment in the travel time needed to pick order $o \in \mathcal{O}$ when added to route k (see Briant et al. (2020)). When route k cannot allocate other orders, a new route is initialized. This step of the procedure continues until all the orders have been assigned to a route.

Then, all the generated routes are concatenated to form a sequence σ made of all the orders. Now, an acyclic graph $G_\sigma = (V_\sigma, A_\sigma)$ is built over σ as follows. V_σ contains a node for each order plus a dummy node 0 and A_σ contains the arc (i,j) for all $0 = i < j = |\mathcal{O}|$, if picking items of orders $\sigma_{i+1}, \ldots, \sigma_j$ is feasible with respect to the capacity of the trolley. The cost associated with the arc is exactly the travel time needed to retrieve such orders. We then compute the shortest path on G_σ that starts from the dummy node 0 and ends at $\sigma_{|\mathcal{O}|}$ and store all the routes associated with arcs in the shortest path in a set $\mathcal{S}$. Note that considering the set of arcs of the shortest path provides a set of routes that is guaranteed to pick all the orders in $\mathcal{O}$. However such set is not guaranteed to provide a feasible solution with respect to the bin-packing constraints. This step of the procedure is inspired by the so-called *Split* procedure proposed in Prins (2004) to obtain a solution for the capacitated vehicle routing problem from a sequence of all the customers to serve.

Note that each of the route associated with each arc of A_σ goes in $\mathcal{K}_{pool}$. Formulation $(\mathcal{B}(\mathcal{K}'))$ defined over the set of routes $\mathcal{K}' = \mathcal{S} \cup \mathcal{S}_0$ is solved (with a time limit of 10 seconds) in order to find feasible solutions with the generated routes. Note that $\mathcal{S}_0$ contains all routes of size 1 and 2. The solution s^* that is obtained is then stored.

We finally apply a local search-based step, where we swap pairs of orders in σ. We make sure not to swap orders that belong to the same arc of the shortest path previously computed. After each swap, we call the split procedure again and update set $\mathcal{S}$ that is used to call the bin packing solver. If the provided solution is better than the previous one, s^* is updated and the swap is implemented, i.e. sequence σ is modified accordingly. Concurrently, $\mathcal{K}_{pool}$ is updated with all the new routes/arcs that are generated.

Note that a swap in σ implies a minor modification of the graph G_σ. We thus do not need to build it from scratch at every operation.

The procedure terminates when a time limit τ_{pool} is reached.

5.7 Initialization of the restricted master problem

The restricted master problem is initialized by adding in $\mathcal{K}'$ all routes of size 1. Then $\mathcal{K}'$ is completed with at most 2000 routes of size 2, 3 and 4 that fill the trolley at at least 90% of its capacity, starting by generating those of size 2, then size 3 and finally size 4. Then, for all the tours generated, if they do not use all the capacity, we add some orders if possible to fill the remaining capacity. We also add in $\mathcal{K}'$ the routes that compose the best solution found by the procedure described in Section 5.6.

Concerning the valid inequalities, we only add from the beginning the SCC cut defined over the entire set of orders $\mathcal{O}$.

6 Experimental results

The experiments were performed on an Intel(R) Core(TM) i7-8650U CPU @ 2.11 GHz processor with 16 GB of RAM and each algorithm ran on a single thread. The code was written in Java and CPLEX 20.1 was used to solve the LPs and the MIPs.

6.1 Parameters

The CGH uses a set of parameters that we list here. The procedure that fills pool $\mathcal{K}_{pool}$ is run for $\tau_{pool} = 600$ seconds. The resolution of $Pr(\mathcal{K}')$ has a time limit τ_{Pr} of 180 seconds. When CGH terminates, $(B(\mathcal{K}'))$ is solved with a time limit τ_l of the available remaining time to solve the problem. In case τ_l is less than 600 seconds, τ_l is set to 600 seconds. The time limit to solve the small instances is set to 3 600 seconds while the time limit to solve the large instances is set to 14 400 seconds.

6.2 Benchmark of instances

To evaluate the performances of procedure CGH proposed in Section 5, we use the benchmark of instances generated to evaluate the iterated local search (ILS) algorithm proposed in van Gils et al. (2019). This benchmark of instances is divided in two sets: the *small instances* made of 6, 12 and 18 orders, and the *large instances* composed of 100, 200 and 300 orders. Other four parameters, each taking three different values, are used to generated the instances. These parameters specify the layout of the warehouse, the storage policy, the trolley capacity, and the deadline distribution (see Table 3 of van Gils et al. (2019)). In the case of the small (resp. large) instances, for each set of the five parameters (the size of the orders plus the other four), 10 (resp. 30) instances are generated. This leads to a total of 2430 small (resp. 7290 large) instances. Thus, the benchmark of instances is made of 9720 instances. We report the results obtained with the procedure presented in Section 5 on the following subsets of instances:

- small instances: we select only one replication for each set of parameters, thus 243 instances are considered;
- large instances: we select only instances with 100 orders and trolley capacity in $\{15, 30\}$, for a total of 54 instances.

We limit the small instances to one replication due to similar behaviour of the procedure over the other instances. For the large instances, the non-considered instances are clearly too large to be tackle with our approach.

Finally, we report that some issues were found in the data sets of van Gils et al. (2019). We contacted the authors and we had kind and constructive exchanges with Kris Braekers who helped in correcting the problems[2]. After the correction, one instance of the small as well as one of the large set turned out to be infeasible. We thus removed them for sake of comparison with the ILS proposed by van Gils et al. (2019). Note that Kris Braekers let the algorithm proposed in van Gils et al. (2019) run on the corrected benchmark. We thus compare here the performances of CGH against the new results obtained by the ILS. Note that we provide optimal results on all small instances, so these results make it possible to assess the quality of the solutions obtained by the ILS.

The remainder of this section is organized as follows. In Sections 6.3 and 6.4, we first evaluate the interest of the proposed valid inequalities and strengthened tour constraints in the pricing, respectively. Then, optimal results for the small instances are provided in Section 6.5. Finally, the results of CGH on large instances are reported in Section 6.6.

6.3 Analysis on the contribution of valid the inequalities

In this section, we evaluate the potential of each family of inequalities presented in Sections 4.2 and 4.3. To this end we solve $(M(\mathcal{K}))$, the linear relaxation of $(B(\mathcal{K}))$ on the small instances, with different configurations. Note that since the instances are small, we can generate all columns in $\mathcal{K}$ and evaluate their duration by calling the dynamic programming of Cambazard and Catusse (2018).

Table 2 provides details on the contribution of each family of cuts on the value of $(M(\mathcal{K}))$ the linear relaxation of $(B(\mathcal{K}))$. The first two columns labelled *Data sets* report the number of orders ($|\mathcal{O}|$) and the number of instances (#Inst). The remaining columns, labelled *Average Root Gap (%)* report the optimality gap at the root node of the branch and bound tree for six different configurations: configuration labelled *None* solves $(\mathcal{M}(\mathcal{K}))$ without the use of any cut; configuration *All* uses all families of cuts; configuration *No cut*, with $cut \in \{MT, FS, R1, SC\}$ uses all the families of cuts but one, that is indicated by parameter *cut*. When $cut = MT$, it means Martello and Toth's cuts (10) (Section 4.2.1); when $cut = FS$ it means Fekete and Schepers's cuts (11) (Section 4.2.2); when $cut = R1$ it means rank-1 cuts (13) (Section 4.3.2); when $cut = SC$ it means the strengthened capacity cuts (12) (Section 4.3.1). The *Average Root Gap (%)* is computed as: $100\frac{z^*_{B(\mathcal{K})} - z^*_{M(\mathcal{K})}}{z^*_{M(\mathcal{K})}}$, where $z^*_{B(\mathcal{K})}$ is the optimal value of $B(\mathcal{K})$, and $z^*_{M(\mathcal{K})}$ is the optimal value of $M(\mathcal{K})$ after adding the corresponding violated cuts.

[2]The updated data-sets as well as the best solutions found are available on this page: `https://pagesperso.g-scop.grenoble-inp.fr/~cambazah/sequencing/`

Data sets		Average Root Gap (%)					
$\|\mathcal{O}\|$	#Inst	None	All	No MT	No FS	No R1	No SC
6	81	0.35	0	0.14	0	0	0
12	81	0.84	0.05	0.06	0.06	0.14	0.15
18	80	1.00	0.08	0.14	0.08	0.27	0.33

Table 2: Average root optimality gap for different sets of valid inequalities.

Overall the strengthened capacity cuts and the rank-1 cuts seem to be the most effective while the Fekete and Schepers's cuts have only a small impact on instances with 12 orders. It is also interesting to note that for instances with 6 orders, not adding the Martello and Toth's cuts yields a positive root gap while not adding other families of cuts has no impact. Moreover, from our experiments, we also note that the strengthened capacity cut defined over the whole set of orders $\mathcal{O}$ is a very effective cut. In conclusion, since all the families of inequalities help in solving the considered set of instances, we decided to consider all of them in the final configuration of CGH.

6.4 Interest of the strengthened tour constraints (36)

In this section, we assess the interest of using the strengthened tour constraints in the MIP formulation of the pricing problem (see Section 5.2.2). On the 26 instances with a 100 orders and a trolley capacity of 15, we have run CGH without considering the strengthened tour constraints, i.e. by considering only the classical tour constraints (31) as already proposed in Briant et al. (2020). The time limit is set to 14 400 seconds. Table 3 reports overall results on this set of 26 instances. Column *Strengh.TourCst.* indicates if CGH has been run with or without strengthened tour constraints. The other columns report the number of instances that reach the time limit (*#TimeLimit*), the average computation time in seconds to run all the instances (*Avg. Cpu(s)*), the number of instances solved to proven optimality (*#Opt*), the number of instances with a final gap that is lower than 1% (*#Gap≤1%*) and the average optimality gap (*Avg. Gap(%)*).

Strengh.TourCst.	#TimeLimit	Avg. Cpu(s)	#Opt	#Gap≤1%	Avg. Gap(%)
yes	10	8 433	4	21	1.73
no	16	11 748	3	11	5.80

Table 3: Overall comparison of CGH on data sets of 100 orders and batch capacity 15, with and without strengthened tour constraints in the pricing.

From the results of Table 3, it is clear that adding the strengthened tour constraints in the MIP formulation of the pricing problem enables to reduce the computation time and to obtain optimality gaps of higher quality. These results thus reflect the theoretical dominance of the strengthened tour constraints (36) over the tour constraints (33) from a computational point of view. We then consider them in the final configuration of CGH.

6.5 Exact algorithm using $(B(\mathcal{K}))$ for small size instances

On the small instances a complete enumeration of the feasible routes is possible in a reasonable amount of time. The duration of each route is computed with the dynamic programming of Cambazard and Catusse (2018). Therefore formulation $(B(\mathcal{K}))$, strengthened with all the valid inequalities presented in Sections 4.2 and 4.3, can be solved as a compact integer programming formulation[3].

The results on small instances are shown in Table 4. The first two columns labelled *Data sets* report the number of orders ($|\mathcal{O}|$) and the number of instances (#Inst). The next five columns labelled *Cpu (s)* report, in seconds, the following computational times: the average (*avg*), the minimum (*min*) and the maximum (*max*) computational time to solve one instance with the corresponding characteristics, the average time to generate all the routes (*InitGen*) and the average time to solve the MIP formulation (*MIP*).

We first note that all instances can be solved to optimality within the given time limit. The maximum time needed is 410.2 seconds. The generation of all columns is the most time-consuming part and takes in average 40.0 seconds for instances with 18 orders. We note that the resulting MIP formulation is often solved relatively quickly: it indeed takes 4.9 seconds in average for instances with 18 orders. The remaining time (that is not detailed in the table) is used to separate the family of cuts that we consider. Note that, as could be expected, this considerably outperforms the exact approach proposed in van Gils et al. (2019) which fails to solve to proven optimality 40.6% of the small instances within a time limit of 4 hours. Note that having optimal solutions for all small instances now enables to assess the quality of the ILS proposed in van Gils et al. (2019).

Data sets		Cpu (s)				
$\|\mathcal{O}\|$	#Inst	avg	min	max	InitGen	MIP
6	81	0	0.0	0.1	0	0
12	81	1.5	0.0	6.5	1.4	0
18	80	49.4	0.0	410.2	40.0	4.9

Table 4: Overall results on the small instances.

To conclude, formulation $(B(\mathcal{K}))$ can solve to optimality all instances with up to 18 orders. This is thanks to the fact that all the routes can be generated beforehand.

6.6 Evaluation of CGH on large size instances

We turn our attention to the ability of CGH to provide high quality lower and upper bounds on the large instances. A time limit of 14 400 seconds (4 hours) is imposed. Note that the final resolution of $B(\mathcal{K}')$ is not included in the time limit, hence the final cpu time may exceed the 4 hours.

Tables 5 and 6 report the results obtained on instances of 100 orders and a trolley capacity B equal to 15 and 30.

[3]Note that when ignoring the sequencing part of the problem, it boils down to the batching and picker routing problem. All these small instances can also be optimally solved in this case.

The first five columns labelled *Data set* report the identification of the instance (*Id*), the number of pickers ($|\mathcal{P}|$), the number of aisles in the warehouse (*#Aisles*), the storage location policy that is used during the generation of the instance (*Loc*) and the deadline distribution (*DD*). Note that values reported in the columns *Loc* and *DD* are in $\{1, 2, 3\}$ with the following meaning: *i.e* 1='Random', 2='Within aisle', 3='Across aisle' and 1='Uniform', 2='Triangular progressive', 3='Triangular degressive', respectively.

The sixth column (*UB*) reports the value of the best solution found by the ILS algorithm of van Gils et al. (2019). The last five columns labelled *CGH* concern the performances of CGH. In particular column *UB* reports the best upper bound found by CGH, column *Cpu (s)* indicates the computation time in seconds, column *Gap (%)* reports the optimality gap computed as: $100\frac{z^{UB}-z^{LB}}{z^{LB}}$, where z^{UB} and z^{LB} respectively denote the best upper and lower bounds obtained at the end of CGH. The optimality gap is reported in italic when it is zero, to emphasize that the optimality on the respective instance has been proven. Finally, columns *#Cols* and *#It* report the number of columns in $\mathcal{K}'$ at the end of the procedure and the number of calls to the pricing problem (see line 11 of Algorithm 1), respectively. Note that when some negative reduced cost columns are found in the pool, this is not counted as an extra iteration.

Data set					ILS	CGH				
Id	|P|	#Aisles	Loc	DD	UB	UB	Cpu (s)	Gap (%)	#Cols	#It
7	4	12	1	1	360 314	**359 348**	3 361	0.02	6 386	73
8	3	12	1	2	308 226	**307 070**	7 931	0.04	13 367	99
9	4	12	1	3	337 286	**332 952**	4 006	0.01	7 325	74
34	3	12	2	1	285 768	**281 688**	3 448	0.03	6 031	86
35	3	12	2	2	279 886	**278 818**	1 129	*0.00*	5 104	38
36	4	12	2	3	265 444	**264 568**	2 990	0.02	8 500	79
61	3	12	3	1	292 240	**290 648**	10 064	0.02	12 935	124
62	4	12	3	2	337 164	**336 990**	1 434	*0.00*	453	37
63	4	12	3	3	294 490	**290 808**	8 205	0.01	9 669	120
88	5	24	1	1	510 786	**504 552**	14 408	0.45	11 182	111
89	5	24	1	2	453 518	**448 960**	14 404	0.71	9 645	120
90	6	24	1	3	533 726	**531 584**	14 423	4.18	9 818	91
115	5	24	2	1	440 786	**437 212**	2 515	0.01	5 511	48
116	5	24	2	2	487 356	**482 316**	2 373	0.02	4 469	36
117	6	24	2	3	426 670	**426 226**	4 596	0.06	5 874	74
142	5	24	3	1	452 266	**446 276**	14 405	0.05	9 235	99
143	5	24	3	2	449 182	**443 084**	6 089	*0.00*	6 667	78
144	6	24	3	3	471 632	**470 294**	14 434	4.61	10 655	95
169	6	36	1	1	708 648	**693 786**	15 003	1.72	9 874	113
170	6	36	1	2	583 298	**582 794**	14 462	24.7	15 797	71
196	6	36	2	1	559 908	**553 320**	14 406	0.57	8 940	110
197	6	36	2	2	652 180	**650 470**	3 486	0.01	4 938	28
198	7	36	2	3	547 630	**541 890**	10 868	0.02	10 086	120
223	6	36	3	1	525 258	**517 924**	14 419	7.60	14 267	108
224	6	36	3	2	614 394	**610 220**	14 413	0.03	8 491	93
225	7	36	3	3	654 416	**649 296**	1 990	*0.00*	5 032	51

Table 5: Detailed results on data sets of 100 orders ($|\mathcal{O}| = 100$), and trolley capacity $B = 15$.

From the results in Table 5, we point out the following observations.

- 16 out of the 26 instances solved the linear relaxation of $B(\mathcal{K})$ before the time limit of 4 hours.
- 21 out of the 26 instances are solved to near optimality and the optimality gap at the end of the computation is lower than 1%. 4 instances, that is Id35, Id62, Id143 and Id225, are solved to optimality. We can claim that the lower bounds provided by $(LM(\mathcal{K}))$ are of very high quality even on large size data sets.
- The lower bound computed by CGH allows to assert the quality of ILS on this benchmark which provides overall very good solutions. Note that all upper bounds are slightly improved by CGH but with a significantly longer time (ILS is run with a limited number of iterations and its runtime is 22 seconds in average with a maximum of 70 seconds).
- The number of calls to the pricing problem is rather small for the given time limit. Moreover, more than 95% of the time is spent to solve the pricing problem, hence the solving of the pricing problem is the bottleneck of the proposed approach.

Table 6 reports results obtained on instances with trolley capacity of 30. Note that a '-' in column *UB* means that a feasible solution has not been found for the respective instance. Similarly a '-' in column *Gap (%)* means that the lower bound is zero. In this case we do not report optimality gaps. Note that for instance Id180, CGH fails in both providing a valid UB and a strictly positive LB.

Data set					ILS	CGH				
Id	|P|	#Aisles	Loc	DD	UB	UB	Cpu (s)	Gap (%)	#Cols	#It
16	4	12	1	1	305 840	**305 016**	14 918	9.25	18 199	92
17	3	12	1	2	268 896	**268 846**	15 046	22.44	20 045	53
18	4	12	1	3	**256 420**	260 508	15 042	27.95	20 326	65
43	3	12	2	1	219 278	**218 018**	14 492	2.53	19 107	123
44	3	12	2	2	203 742	**203 030**	14 538	4.92	22 551	100
45	4	12	2	3	201 350	**200 998**	15 111	1.75	25 709	131
70	3	12	3	1	232 510	**232 036**	14 870	22.31	19 356	71
71	3	12	3	2	**228 188**	229 908	15 006	15.65	20 963	77
72	4	12	3	3	**239 470**	240 434	15 002	20.14	19 233	64
97	4	24	1	1	**343 062**	361 130	15 006	878.73	15 722	43
98	5	24	1	2	**458 204**	458 814	14 971	63.07	17 855	59
99	5	24	1	3	**417 336**	446 658	15 026	228.18	20 151	67
124	4	24	2	1	**322 434**	331 250	15 036	96.64	20 757	79
125	4	24	2	2	**321 718**	324 468	15 025	57.83	21 913	45
126	5	24	2	3	349 442	**348 542**	14 631	70.17	23 785	53
151	4	24	3	1	**358 618**	378 824	15 003	194.68	20 663	59
152	4	24	3	2	**361 010**	366 308	15 013	81.21	17 877	49
153	5	24	3	3	**383 112**	387 088	15 018	69.15	17 587	50
178	5	36	1	1	**480 888**	510 588	15 010	-	15 645	71
179	5	36	1	2	**531 724**	553 044	15 037	-	17 443	60
180	6	36	1	3	**600 422**	-	15 042	-	16 480	56
205	5	36	2	1	**451 856**	470 242	15 030	192.73	20 133	66
206	6	36	2	2	**502 722**	507 516	14 740	108.39	18 221	35
207	7	36	2	3	**503 854**	508 732	15 012	125.61	19 487	50
232	6	36	3	1	**523 364**	529 706	14 483	351.96	20 450	42
233	6	36	3	2	**497 086**	504 674	15 025	148.86	17 395	45
234	6	36	3	3	**450 494**	467 620	15 024	644.05	17 532	44

Table 6: Detailed results on data sets of 100 orders ($|\mathcal{O}| = 100$), and trolley capacity $B = 30$.

From the results in Table 6, we point out the following observations.

- All instances ran up to the time limit of 4 hours.
- No instance is solved to near optimality, only four instances report a gap of lass than 10%. Moreover, for three instances (Id178, Id179 and Id180) we report an infinite optimality gap. Hence, instances with $B = 30$ are much more difficult to be solved with CGH.
- CGH is able to improve the value of the upper bound with respect to the ILS only for 7 instances. For fairness of the analysis we note that the improvement is rather small. On the other hand, when ILS performs better than CGH, the UB provided by the latter is sometime of poor quality with respect to the one provided by ILS. Moreover CGH cannot find a feasible solution on instance Id180.

As a conclusion, it can observed that CGH provided overall very good results on large size instance with a small trolley capacity ($B = 15$). However, the performances of CGH deteriorate drastically when enlarging the capacity of the trolley to $B = 30$.

7 Conclusion

In this paper we showed that the Joint Order Batching, Picker Routing and Sequencing Problem with Deadlines (JOBPRSP-D) is better captured by formulating it as a bin packing problem rather than a scheduling problem. We believe that the heuristic approaches as proposed in van Gils et al. (2019) can be improved based on this analysis. This observation lead to the design of an algorithm with performance guarantee (i.e., able to provide valid lower and upper bounds), for a very complex integrated logistic problem. Experiments results showed that the proposed approach correctly scales on instances of reasonable sizes.

A key contribution is the design of valid inequalities for the master problem as well as for the pricing problem. The first family of cutting planes takes advantage of the bin packing analysis and relies on Dual-Feasible Functions. Regarding the pricing problem, the inequalities proposed in Briant et al. (2020) (namely the tour constraints) are also strengthened. This allows to efficiently solve the pricing problem that can be seen as the bottleneck of the approach due to the hardness of the picker routing problem. As a result, the work of this paper shows that the column generation based heuristic (CGH) proposed for the Joint Order Batching and Picker Routing Problem (JOBPRP) in Briant et al. (2020) can be extended to consider deadlines associated with orders to be prepared.

The proposed algorithm is able to solve optimally all the small instances generated by van Gils et al. (2019) and involving up to 18 orders. Surprisingly, it provides very tight intervals of the optimal values (and improved upper bounds) for some of the large instances of 100 orders. It therefore contributes in asserting the quality of the heuristic techniques often proposed in this area of research due to the hardness of these integrated problems.

Future research on the topic may involve the JOBPRSP-D in the dynamic setting where orders are dynamically released during the working day (see for example D'Haen et al. (2022)). The proposed approach may be used to compute an initial solution that collects already known orders. A quick insertion heuristic would then need to be developed to consider the inclusion of dynamic orders in the current plan.

Another interesting research perspective is to integrate more features in the scheduling aspect of the JOBPRSP-D. The consideration of breaks in the schedule of the pickers can typically be of interest, as well as the possibility to associate release dates to orders, if an order cannot be collected too early to avoid its items staying too much time in the staging area (Rijal et al., 2021). Considering such features would certainly require a scheduling formulation, rather than a bin backing formulation as presented in the current work.

Another interesting problem would be to integrate to the JOBPRP or to the JOBPRSP-D storage location assignment decisions at the operational level. It is indeed known that in the e-commerce context, the forward area of the warehouse is supplied on a daily basis. Historical information coming from previous orders can be used to efficiently fill the forward area. From a computational point of view, another interesting perspective is to develop a more efficient algorithm to optimaly solve the pricing problem, which is the bottleneck of the proposed approach.

References

Ardjmand, E., Ghalehkhondabi, I., Young II, W. A., Sadeghi, A., Weckman, G. R., and Shakeri, H. (2020). A hybrid artificial neural network, genetic algorithm and column generation heuristic for minimizing makespan in manual order picking operations. *Expert Systems with Applications*, 159:113566.

Ardjmand, E., Shakeri, H., Singh, M., and Bajgiran, O. S. (2018). Minimizing order picking makespan with multiple pickers in a wave picking warehouse. *International Journal of Production Economics*, 206:169–183.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385.

Briant, O., Cambazard, H., Cattaruzza, D., Catusse, N., Ladier, A., and Ogier, M. (2020). An efficient and general approach for the joint order batching and picker routing problem. *Eur. J. Oper. Res.*, 285(2):497–512.

Cambazard, H. and Catusse, N. (2018). Fixed-parameter algorithms for rectilinear steiner tree and rectilinear traveling salesman problem in the plane. *European Journal of Operational Research*, 270(2):419–429.

Çağla Cergibozan and Tasan, A. (2019). Order batching operations: an overview of classification, solution techniques, and future research. *Journal of Intelligent Manufacturing*, 30:335–349.

Chen, T. L., Cheng, C. Y., Chen, Y. Y., and Chan, L. K. (2015). An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. *International Journal of Production Economics*, 159:158–167.

Clautiaux, F., Alves, C., and de Carvalho, J. V. (2010). A survey of dual-feasible and superadditive functions. *Annals of Operations Research*, 179(1):317–342.

Costa, L., Contardo, C., and Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985.

de Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.

D'Haen, R., Braekers, K., and Ramaekers, K. (2022). Integrated scheduling of order picking operations under dynamic order arrivals. *International Journal of Production Research*, pages 1–22.

Fekete, S. P. and Schepers, J. (2001). New classes of fast lower bounds for bin packing problems. *Mathematical programming*, 91(1):11–31.

Gademann, N. and van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 37(1):63–75.

Haouassi, M., Kergosien, Y., Mendoza, J. E., and Rousseau, L.-M. (2022). The integrated orderline batching, batch scheduling, and picker routing problem with multiple pickers: the benefits of splitting customer orders. *Flexible Services and Manufacturing Journal*, 34(3):614–645.

Henn, S. (2015). Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. *Flexible Services and Manufacturing Journal*, 27(1):86–114.

Henn, S. and Schmid, V. (2013). Metaheuristics for order batching and sequencing in manual order picking systems. *Computers and Industrial Engineering*, 66(2):338–351.

Heßler, K. and Irnich, S. (2022). Modeling and exact solution of picker routing and order batching problems. Technical Report LM-2022-03, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Hintsch, T., Irnich, S., and Kiilerich, L. (2021). Branch-price-and-cut for the soft-clustered capacitated arc-routing problem. *Transportation Science*, 55(3):687–705.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

Martello, S. and Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete applied mathematics*, 28(1):59–70.

Menéndez, B., Bustillo, M., Pardo, E. G., and Duarte, A. (2017). General variable neighborhood search for the order batching and sequencing problem. *European Journal of Operational Research*, 263(1):82–93.

Muter, İ. and Öncan, T. (2022). Order batching and picker scheduling in warehouse order picking. *Iise Transactions*, 54(5):435–447.

Muter, I. and Öncan, T. (2015). An exact solution approach for the order batching problem. *IIE Transactions*, 47(7):728–738.

Pansart, L., Catusse, N., and Cambazard, H. (2018). Exact algorithms for the order picking problem. *Comput. Oper. Res.*, 100:117–127.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017a). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., and Santos, H. (2017b). Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, 45(3):206–209.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computer & Operations Research*, 31(12):1985–2002.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations research*, 31(3):507–521.

Rijal, A., Bijvank, M., Goel, A., and de Koster, R. (2021). Workforce scheduling with order-picking assignments in distribution facilities. *Transportation Science*, 55(3):725–746.

Scholz, A., Schubert, D., and Wäscher, G. (2017). Order picking with multiple pickers and due dates – Simultaneous solution of Order Batching, Batch Assignment and Sequencing, and Picker Routing Problems. *European Journal of Operational Research*, 263(2):461–478.

Tsai, C. Y., Liou, J. J., and Huang, T. M. (2008). Using a multiple-GA method to solve the batch picking problem: Considering travel distance and order due time. *International Journal of Production Research*, 46(22):6533–6555.

Valle, C. A., Beasley, J. E., and Da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, 262(3):817–834.

van Gils, T., Caris, A., Ramaekers, K., and Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, 277(3):814–830.

van Gils, T., Ramaekers, K., Caris, A., and de Koster, R. B. (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, 267(1):1–15.

Vanheusden, S., van Gils, T., Ramaekers, K., Cornelissens, T., and Caris, A. (2022). Practical factors in order picking planning: state-of-the-art classification and review. *International Journal of Production Research*.

Wahlen, J. and Gschwind, T. (2023). Branch-price-and-cut-based solution of order batching problems. *Transportation Science*, 57(3):756–777.

8 Appendix

8.1 Proof of Proposition 4

Proposition 4 *The following constraints*

$$\sum_{k \in \mathcal{K}_1(\bar{d},q)} \bar{d}\,\rho_k + \sum_{k \in \mathcal{K}_2(\bar{d},q)} t_k\,\rho_k \leq \bar{d} \times |\mathcal{P}| \quad \forall \bar{d} \in \mathcal{D}, q \in \mathcal{Q}(\bar{d}) \qquad (10)$$

are valid inequalities for $(M(\mathcal{K}))$.

Proof: Consider the function f_0^λ from the L_2 lower bound proposed by Martello and Toth (1990). Let $\lambda \in \left[0; \frac{1}{2}\right]$:

$$f_0^\lambda : [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} 1, & \text{if } x > 1-\lambda \\ x, & \text{if } \lambda \leq x \leq 1-\lambda \\ 0, & \text{if } x < \lambda \end{cases}$$

By applying Proposition 3 to constraints (3) with DFF f_0^λ, the following valid inequalities are obtained:

$$\sum_{k \in \mathcal{K}(\bar{d})} f_0^\lambda\left(\frac{t_k}{\bar{d}}\right)\rho_{kp} \leq 1, \quad \forall \bar{d} \in \mathcal{D}, p \in \mathcal{P}$$

This can be written as:

$$\sum_{k \in \mathcal{K}(\bar{d}) \mid \frac{t_k}{\bar{d}} < \lambda} 0\rho_{kp} + \sum_{k \in \mathcal{K}(\bar{d}) \mid \lambda \leq \frac{t_k}{\bar{d}} \leq 1-\lambda} \frac{t_k}{\bar{d}}\rho_{kp} + \sum_{k \in \mathcal{K}(\bar{d}) \mid \frac{t_k}{\bar{d}} > 1-\lambda} 1\rho_{kp} \leq 1, \quad \forall \bar{d} \in \mathcal{D}, p \in \mathcal{P}$$

By multiplying by $\bar{d}$, and by setting $\lambda = \frac{q}{\bar{d}}$, we obtain the following:

$$\sum_{k \in \mathcal{K}(\bar{d}) \mid q \leq t_k \leq \bar{d}-q} t_k\rho_{kp} + \sum_{k \in \mathcal{K}(\bar{d}) \mid t_k > \bar{d}-q} \bar{d}\rho_{kp} \leq \bar{d}, \quad \forall \bar{d} \in \mathcal{D}, p \in \mathcal{P}$$

By summing over all $p \in \mathcal{P}$ and taking into consideration (5), we get exactly constraints (10). □

8.2 Proof of Proposition 5

Proposition 5 *The following inequalities*

$$\sum_{i=1}^{q-1} \sum_{k \in \mathcal{K}(\bar{d},i,q)} i\,\rho_k \leq (q-1) \times |\mathcal{P}| \quad \forall \bar{d} \in \mathcal{D}, q \in \{2, \ldots, \bar{d}\} \quad (11)$$

are valid for $(M(\mathcal{K}))$.

Proof: Fekete and Schepers (2001) proposed a DFF denoted by $f_{FS,1}^\lambda$ (see Clautiaux et al.

(2010)). Given a $\lambda \in \mathbb{N} \setminus \{0\}$, the function is the following:

$$f^{\lambda}_{FS,1} : [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} x, & \text{if } x(\lambda+1) \in \mathbb{Z} \\ \lfloor (\lambda+1)x \rfloor \frac{1}{\lambda}, & \text{otherwise} \end{cases}$$

We consider a weaker version of $f^{\lambda}_{FS,1}$ referred to as g^{λ} which handles differently the case $x(\lambda+1) \in \mathbb{Z}$:

$$g^{\lambda}(x) = \begin{cases} 0 & \text{if } x = 0 \\ \frac{x(\lambda+1)-1}{\lambda} & \text{if } x(\lambda+1) \in \mathbb{Z} \setminus \{0\} \\ \lfloor (\lambda+1)x \rfloor \frac{1}{\lambda}, & \text{otherwise} \end{cases}$$

Note that $g^{\lambda}(x) \leq f^{\lambda}_{FS,1}(x)$ for any $x \in [0,1]$ so that g^{λ} is a non-maximal DFF. By applying Proposition 3 to constraints (3) with $g^{\lambda}(x)$, the following valid inequalities are obtained:

$$\sum_{k \in \mathcal{K}(\bar{d}) \mid \frac{t_k}{\bar{d}}(\lambda+1) \in \mathbb{Z}} \frac{\frac{t_k}{\bar{d}}(\lambda+1) - 1}{\lambda} \rho_{kp} + \sum_{k \in \mathcal{K}(\bar{d}) \mid \frac{t_k}{\bar{d}}(\lambda+1) \notin \mathbb{Z}} \left\lfloor (\lambda+1)\frac{t_k}{\bar{d}} \right\rfloor \frac{1}{\lambda} \rho_{kp} \leq 1, \quad \forall \bar{d} \in \mathcal{D}, p \in \mathcal{P}$$

Let us now set $\lambda = q-1$, and $\frac{t_k}{d}(\lambda+1) = j$ and multiply by $q-1$. We obtain:

$$\sum_{k \in \mathcal{K}(\bar{d}) \mid t_k = j\frac{\bar{d}}{q}, j \in \mathbb{Z}} (j-1)\rho_{kp} + \sum_{k \in \mathcal{K}(\bar{d}) \mid t_k = j\frac{\bar{d}}{q}, j \notin \mathbb{Z}} \lfloor j \rfloor \rho_{kp} \leq q-1, \quad \forall \bar{d} \in \mathcal{D}, p \in \mathcal{P}$$

Consider $i = \lfloor j \rfloor$, we can replace $t_k = j\frac{\bar{d}}{q}, j \notin \mathbb{Z}$ by $i\frac{\bar{d}}{q} < t_k < (i+1)\frac{\bar{d}}{q}, i \in \mathbb{Z}$. This gives the following constraints:

$$\sum_{k \in \mathcal{K}(\bar{d}) \mid t_k = i\frac{\bar{d}}{q}, i \in \mathbb{Z}} (i-1)\rho_{kp} + \sum_{k \in \mathcal{K}(\bar{d}) \mid i\frac{\bar{d}}{q} < t_k < (i+1)\frac{\bar{d}}{q}, i \in \mathbb{Z}} i\rho_{kp} \leq q-1, \quad \forall \bar{d} \in \mathcal{D}, p \in \mathcal{P}$$

Thus the following inequality is valid:

$$\sum_{k \in \mathcal{K}(\bar{d}) \mid i\frac{\bar{d}}{q} < t_k \leq (i+1)\frac{\bar{d}}{q}, i \in \mathbb{Z}} i\rho_{kp} \leq q-1, \quad \forall \bar{d} \in \mathcal{D}, p \in \mathcal{P} \qquad (37)$$

By summing (37) over all $p \in \mathcal{P}$ and taking into consideration (5), we get exactly constraints (11).

□

8.3 Proof of Proposition 8

Proposition 8 *If constraint* (34) *is valid and hypothesis* (35) *holds, then the following con-*

straint

$$b + \sum_{o \in \mathcal{O}} a_o \, e_o \; + \; \Delta_k \left(\sum_{o \in \mathcal{O}_k} e_o - |k| + 1 \right) \leq t \qquad (36)$$

is valid for $Pr(\mathcal{K}')$ *and allows to exactly compute* t_k.

Proof: Let k' be the route defined by the set of orders $o \in \mathcal{O}$ such that $e_o = 1$. We need to prove that constraint (36) is satisfied for k', i.e.

$$b + \sum_{o \in k'} a_o \; + \; \Delta_k \left(|k \cap k'| - |k| + 1 \right) \leq t_{k'}.$$

Let us consider the three possible cases:

case 1 : $k' = k$, then constraint (36) becomes $t \geq t_k$. It is valid and also permits to exactly compute $t_{k'}$.

case 2 : $k \cap k' = k$, i.e. $k \subseteq k'$, then constraint (36) becomes
$t \geq b + \sum_{o \in k'} a_o \; + \; \Delta_k = t_k + \sum_{o \in k' \setminus k} a_o$
This is valid since from hypothesis (35), we obtain $t_{k'} \geq t_k + \sum_{o \in k' \setminus k} a_o$.

case 3 : $k \cap k' \neq k$, then $|k \cap k'| \leq |k| - 1$, so $\Delta_k \left(|k \cap k'| - |k| + 1 \right) \leq 0$. However, we have $b + \sum_{o \in k'} a_o \leq t_{k'}$ from constraint (34) applied to route k'. Hence constraint (36) is valid.

□

8.4 Modifications of the pricing problem due to each type of non-robust cut

Martello and Toth's cuts. For each Martello and Toth's cut (10) defined by the pair $(\bar{d}^*, q^*)$, we consider the following additional variables. For the sake of clarity we will not index them by $(\bar{d}^*, q^*)$.

- $w \geq 0$: real non-negative variable that equal $\bar{d}^*$ if the route is in $\mathcal{K}_1(\bar{d}^*, q^*)$, and equal t if the route is in $\mathcal{K}_2(\bar{d}^*, q^*)$, and 0 otherwise
- b_1: binary variable that equal 1 if the route is in $\mathcal{K}_1(\bar{d}^*, q^*)$, 0 otherwise
- b_2: binary variable that equal 1 if the route is in $\mathcal{K}_2(\bar{d}^*, q^*)$, 0 otherwise
- b_3: binary variable that equal 1 if the route is in $\mathcal{K}_3(\bar{d}^*, q^*)$, 0 otherwise

The objective function of $Pr(\mathcal{K}')$ is modified by subtracting $\gamma_{MT}^{\bar{d}^* q^*} \, w$, where $\gamma_{MT}^{\bar{d}^* q^*}$ is the dual value associated to the cut. The following constraints are added to $Pr(\mathcal{K}')$:

$$b_1 + b_2 + b_3 + \sum_{\bar{d} > \bar{d}^*} \mu^{\bar{d}} = 1 \quad (38)$$

$$w \geq \bar{d}^*\, b_1 \quad (39)$$

$$w \geq t - \max_{\bar{d} \in \mathcal{D}} \left\{ \bar{d} \right\} (1 - b_2) \quad (40)$$

$$t \geq q^*\, b_2 + \left(\bar{d}^* - q^* + 1 \right) b_1 \quad (41)$$

$$t \leq \bar{d}^*\, b_1 + \left(\bar{d}^* - q^* \right) b_2 + (q^* - 1)\, b_3 + \sum_{\bar{d} > \bar{d}^*} \bar{d}\, \mu^{\bar{d}} \quad (42)$$

$$w \geq 0 \quad (43)$$

$$b_1, b_2, b_3 \in \{0, 1\} \quad (44)$$

Constraint (38) ensures that the route is in one of the three sets $\mathcal{K}_1(\bar{d}^*, q^*)$, $\mathcal{K}_2(\bar{d}^*, q^*)$, $\mathcal{K}_3(\bar{d}^*, q^*)$, or the route has a deadline that is grater than $\bar{d}^*$. Constraint (39) ensures that w takes value $\bar{d}^*$ if the route is in $\mathcal{K}_1(\bar{d}^*, q^*)$, while constraint (40) ensures that w takes value t if the route is in $\mathcal{K}_2(\bar{d}^*, q^*)$. Constraints (41) and (42) ensure the total time of the route is consistent with the set that contains the route. Constraints (43) and (44) define the domain of the decision variables.

Fekete et Schepers cuts. For each Fekete and Schepers' cut (11) defined by the pair $(\bar{d}^*, q^*)$, we need to consider the following additional variable:

- v : non-negative integer variable that represents the number of intervals of size $\bar{d}^*/q^*$ covered by the total time of the route (with a positive residual), i.e. what is denoted by i in constraint (11).

For the sake of clarity we will not indexed it by $(\bar{d}^*, q^*)$.

The objective function of $Pr(\mathcal{K}^{'})$ is modified by subtracting $\gamma_{FS}^{\bar{d}^* q^*}\, v$ where $\gamma_{FS}^{\bar{d}^* q^*}$ is the dual value associated to the cut. The following constraints are added to $Pr(\mathcal{K}^{'})$:

$$t \leq (v+1)\frac{\bar{d}^*}{q^*} + \sum_{\bar{d} > \bar{d}^*} \left(\bar{d} - \frac{\bar{d}^*}{q^*} \right) \mu^{\bar{d}} \quad (45)$$

$$t \geq \frac{\bar{d}^*}{q^*} v + \frac{1}{q^*} \quad (46)$$

$$v \in \mathbb{N} \quad (47)$$

Constraints (45) and (46) ensure that v corresponds to the number of intervals of size $\bar{d}^*/q^*$ covered by the travel time of the route, if the deadline of the route is less or equal than $\bar{d}^*$. Note that the value $1/q^*$ in constraint (46) is set to ensure $t_k > i\, \bar{d}^*/q^*$ in the definition of $\mathcal{K}(\bar{d}^*, i, q^*)$. Finally, constraint (47) defines the domain of the variable.

Strengthened capacity cuts. For each SCC defined by the set of orders $\mathcal{R}^*$, we need to consider the following additional variable:

- z : binary variable that equal 1 if the route contains at least one order in $\mathcal{R}^*$, 0 otherwise.

Fort he sake of clarity we do not index the variable by $\mathcal{R}^*$.

The objective function of $Pr(\mathcal{K}')$ is modified by subtracting $\gamma_{SC}^{\mathcal{R}^*} z$ where $\gamma_{SC}^{\mathcal{R}^*}$ is the dual value associated to the cut. The following constraints are added to $Pr(\mathcal{K}')$:

$$\sum_{o \in \mathcal{R}^*} e_o \geq z \qquad (48)$$

Rank-1 cuts. The management of R1Cs in a MIP formulation of the pricing problem has been recently proposed by Hintsch et al. (2021). They propose two sets of constraints to be included in the MIP, where each set does not dominate the other and are thus worth consideration. We detail hereafter these two sets of constraints.

For each R1C cut (12) defined by the pair $(\mathcal{R}^*, \mathbf{p}^*)$, we need to consider the following additional variable: $u \in \mathbb{N}$ that represents the coefficient $\left\lfloor \sum_{o \in \mathcal{R}^*} p_o^* e_o \right\rfloor$ of the route computed by the pricing problem. For the sake of clarity we do not index the variable by $(\mathcal{R}^*, \mathbf{p}^*)$.

The objective function of $Pr(\mathcal{K}')$ is modified by subtracting $\gamma_{R1}^{\mathcal{R}^* \mathbf{p}^*} u$ where $\gamma_{R1}^{\mathcal{R}^* \mathbf{p}^*}$ is the dual value associated to the cut.

Let us consider that the multipliers $\mathbf{p}^* = \left\{p_1^*, p_2^*, \ldots, p_q^*\right\}$ can be written as $\left\{\frac{s_1^*}{t^*}, \frac{s_2^*}{t^*}, \ldots, \frac{s_q^*}{t^*}\right\}$ with $s_1^*, s_2^*, \ldots, s_q^*, t^* \in \mathbb{N}^*$.

The first constraint to be added to $Pr(\mathcal{K}')$ is the following:

$$\sum_{o \in \mathcal{R}^*} s_o^* e_o - t^* u \leq t^* - 1 \qquad (49)$$

For the second set of constraints, let us introduce the notion of *minimal subset* defined as follows: a set $\mathcal{M} \subseteq \mathcal{R}^*$ is a minimal subset for $\mathcal{R}^*$ and multipliers $\mathbf{p}^*$ if there exits an integer $m \geq 1$ such that:

$$\sum_{o \in \mathcal{M}} p_o \geq m \qquad \text{and} \qquad \sum_{o \in \mathcal{M}'} p_o < m \qquad \forall \mathcal{M}' \subsetneq \mathcal{M}.$$

Let us denote by $\mathcal{N}^*$ the set of all minimal subsets for $\mathcal{R}^*$ and multipliers $\mathbf{p}^*$. The second set of constraints, one for each element in $\mathcal{N}$, is the following:

$$\sum_{o \in \mathcal{M}} e_o - u \leq |\mathcal{M}| - \left\lfloor \sum_{o \in \mathcal{M}} p_o^* \right\rfloor \qquad \forall \mathcal{M} \in \mathcal{N}^* \qquad (50)$$

The interested reader is referred to Hintsch et al. (2021) for a detailed explanation, especially on minimal subsets $\mathcal{N}^*$ for each optimal vector of multipliers for R1Cs defined over sets of orders of cardinality between 3 and 5.