DeLuxing: Deep Lagrangian Underestimate Fixing for Column-Generation-Based Exact Methods

Yu Yang
Department of Industrial and Systems Engineering, University of Florida, yu.yang@ise.ufl.edu

In this paper, we propose an innovative variable fixing strategy called *deep Lagrangian underestimate fixing* (DeLuxing). It is a highly effective approach for removing unnecessary variables in column-generation (CG)-based exact methods used to solve challenging discrete optimization problems commonly encountered in various industries, including vehicle routing problems (VRPs). DeLuxing employs a novel linear programming (LP) formulation with only a small subset of the enumerated variables, which is theoretically guaranteed to yield qualified dual solutions for computing Lagrangian underestimates (LUs). Due to their small sizes, DeLuxing can efficiently solve a sequence of similar LPs to generate multiple high-quality LUs, and thus can, in most cases, remove over 75% of the variables from the enumerated pool. We extend the fundamental concept underpinning the new formulation to contexts beyond variable fixing, namely variable type relaxation and cutting plane addition. We demonstrate the effectiveness of the proposed method in accelerating CG-based exact methods via the capacitated multi-trip vehicle routing problem with time windows (CMTVRPTW), its two important variants with loading times or release dates, and the classic CVRP and VRPTW. Enhanced by DeLuxing and the extensions, one of the best exact methods for solving the CMTVRPTW developed in Yang (2023) doubles the size of instances solved optimally for the first time while being more than 7 times on average and up to over 20 times as fast as top-performing exact methods reported in the literature. It further accelerates RouteOpt, one of the world's fastest VRP solvers (You et al. 2023), by 45% and 71%, respectively, for solving the 200-node CVRP and 300-node VRPTW instances. While DeLuxing is less effective for longer routes, it still contributes to an effective primal heuristic.

Keywords: column generation · variable fixing · Lagrangian underestimate · multi-trip vehicle routing

1. Introduction

The textbook Dantzig-Wolfe decomposition (DWD; Dantzig and Wolfe 1960) naturally gives rise to a column generation (CG) approach for solving challenging linear programs (LPs), where "promising" variablesare generated and added to the restricted master program (RMP) as needed throughout the solution process. This idea of implicitly dealing with variables when there are too many of them dates back to Ford and Fulkerson (1958) and has expanded well beyond the original context of LP solving. In particular, it has been successfully combined with the well-known branch-and-bound framework Land and Doig (2010) into the branch-and-price (BP) approach (Barnhart et al. 1998) and additionally with problem-specific cutting planes into the branch-price-and-cut (BPC) approach

(e.g., Kohl et al. 1999, Fukasawa et al. 2006, Jepsen et al. 2008) for solving integer programs (IPs). BP and BPC methods are now the leading exact algorithms to approach various challenging discrete optimization problems arising in the industry, including vehicle routing problems (VRPs; Pessoa et al. 2020, Desaulniers et al. 2016a), inventory routing problems (Desaulniers et al. 2016b, Engineer et al. 2012), and crew rostering problems (CRPs; Breugem et al. 2022, Quesnel et al. 2020).

A persistent challenge associated with CG is its tendency to generate an excessive number of columns when solving large-scale instances, which slows down the solution process and consumes a large amount of memory. To alleviate this problem, it is possible to adopt a straightforward column clean-up procedure that drops columns with large reduced costs (see Section 5.2 of Pessoa et al. 2020) at the expense of more CG iterations required to solve the LPs optimally. To further mitigate the issue, most BPC methods incorporate reduced cost fixing (RCF), which fixes variables based on their reduced costs, as a default functionality to remove variables (Pecin et al. 2017a,b). RCF builds upon the fact that the reduced cost of a given variable x_i lower bounds the absolute change in the optimal value of the LP relaxation for each unit change in the value of x_i. The variable bound can thus be tightened accordingly to prevent the LP from achieving objective values worse than a known primal bound of the mixed integer program (MIP; for more details, see Wolsey and Nemhauser 1999, p. 389). For problems involving binary variables (e.g., Crowder et al. 1983, Johnson et al. 1985), RCF can directly fix variables, and those fixed to 0 can be safely removed from the formulation without compromising solution optimality.

1.1. Motivation

Compared to classic compact formulations (e.g., vehicle or commodity flow-based formulations, see Baldacci et al. 2004, Cappanera and Gallo 2004), an extensive formulation, such as a set partitioning formulation or DWD reformulation, usually produces much tighter lower bounds (by default, we consider minimization problems in this paper) but needs to be solved by a CG approach due to the exponential number of variables. Such tight lower bounds make it possible to enumerate all columns with reduced costs no larger than the current integrality gap at an early stage. The enumeration implicitly applies RCF and was first recommended by Baldacci et al. (2008) for solving VRPs, which has led to remarkable acceleration (Yang 2023, Sadykov et al. 2021, Paradiso et al. 2020).

The enumeration is usually activated when the current integrality gap drops blew a threshold Δ. Thus, the aggressiveness of enumeration is directly controlled by Δ, with larger values indicating higher aggressiveness. Increasing the aggressiveness within a certain range helps to close an open branch-and-bound node (BBN) faster, while too large a Δ results in the enumeration of an excessively large column pool or even failure of enumeration due to hitting limits on, e.g., time, memory, or the

pool size, causing deterioration in overall performance (Costa et al. 2019). For challenging instances with a reasonable Δ, it is common to have millions of columns or more enumerated.

Although RCF can also be applied after enumeration to reduce the pool size gradually (see Pessoa et al. 2020), when the route length is not long (usually no more than 15 customers per route), its effectiveness is limited for three major reasons. First, the columns in the pool are promising ones and tend to be difficult to remove because they have implicitly passed the initial screening by RCF in the enumeration phase. Second, the effectiveness of RCF relies heavily on the changes in the lower and upper bounds. Multiple rounds of cutting plane addition and branching are typically required before the pool can be shrunk to a tractable size such that the BBN can be closed by directly solving an IP with all columns left in the pool. Consequently, RCF may iterate through the entire pool many times in this process, incurring a substantial increase in computational load. Finally, RCF usually uses only one optimal dual solution from the most recent LP, which can be somewhat arbitrary within the optimal face of the corresponding polyhedron and thus results in fluctuating and mostly mediocre performance.

Unfortunately, to the best of our knowledge, not much progress has been made in addressing the said causes of ineffectiveness, which motivates this research. Specifically, we seek to unlock the full potential of variable fixing for CG-based exact methods when an enumeration procedure is employed.

1.2. Contributions, Limitations, and Outline

This paper proposes a *deep Lagrangian underestimate fixing* (DeLuxing) method widely applicable to accelerate CG-based exact methods. We summarize our contributions as follows.

- *We introduce a novel LP formulation that yields high-quality Lagrangian underestimates (LUs) and rigorously prove its validity.* The LP includes only a small subset of variables (i.e., those with reduced costs no larger than half of the current gap), allowing for a rapid search for promising dual solutions. The variable fixing induced by employing such dual solutions addresses the ineffectiveness of RCF from three perspectives: (i) it imposes much less restriction on qualified dual solutions while the standard RCF generally necessitates the use of optimal dual solutions; (ii) it takes effect at the current BBN by reducing the strong reliance on the quality of the lower bound, avoiding repeated branching or addition of cutting planes before achieving its significance; (iii) it proactively seeks multiple dual solutions to generate LUs of high quality that fix a large number of variables with mild computational overhead.
- *We expand the fundamental idea behind the proposed LP formulation to achieve additional acceleration beyond variable fixing.* More precisely, following this principle, we prove that, under mild conditions, a large proportion of the integer variables can be relaxed to continuous ones in the RMP, thereby expediting the computation. Additionally, we enhance the iterative process of

adding cutting planes by conducting computations on a restricted reformulation, which includes only a subset of all variables — those with reduced costs not exceeding half of the current gap. This new approach to cut addition further accelerates the computation while resulting in minimal or no deterioration in the quality of the obtained lower bound. It is worth mentioning that these extensions preserve the exactness of the entire algorithm.

- *We propose a straightforward yet effective algorithmic framework that systematically directs the exploration for promising dual solutions, and we provide insights into the mechanisms contributing to its success.* The key idea involves bundling columns with similar characteristics into a reference point to guide the search. A clustering procedure is initially employed to identify Euclidean distance-based similarities among columns. The algorithm then starts from each cluster and conducts a deep search using a reference point computed with newly identified removable columns at each iteration until a stopping criterion is reached. One can view this iterative procedure as implicitly revealing the similarities among columns via the reference points.
- *We demonstrate that DeLuxing, as a versatile variable screening tool, effectively removes unnecessary variables and can be flexibly applied throughout a BPC method.* Our experiments on the capacitated multi-trip vehicle routing problem with time window (CMTVRPTW) show that DeLuxing can remove over 75% of the variables in most cases. Its effectiveness is even more pronounced as the problem size increases, achieving a reduction of up to 99%. In addition to its standard usage of removing variables after an exact enumeration, DeLuxing can serve as a crucial component in a highly effective primal heuristic.
- *We conduct an extensive numerical study and show that DeLuxing, along with several acceleration techniques inspired by it, takes the performance of BPC methods to an entirely new level.* One of the best exact methods for solving the CMTVRPTW in Yang (2023) enhanced by the proposed DeLuxing can solve all instances with 140 customers for the first time, doubling the size of instances that can be solved to optimality. Furthermore, it achieves near-optimal solutions with an average optimality gap of 0.5% for instances with up to 200 customers. Additionally, remarkable acceleration (45% and 71%, respectively) is achieved on top of RouteOpt when solving the 200-node CVRP and 300-node VRPTW instances.

The effectiveness of DeLuxing relies on enumerating all necessary columns early in the optimization process, as seen in the CMTVRPTW and its two variants. For classic VRPs (e.g., CVRP and VRPTW) with short to moderate routes, considerable computation is still required after enumeration, and DeLuxing can significantly accelerate the solution process. However, in instances where the route length is excessively long, enumeration may not succeed until a very late stage. A node can be closed rapidly without much branching after a successful enumeration. In such cases, DeLuxing may

not be particularly effective. Nonetheless, the primal heuristic inspired by DeLuxing demonstrates remarkable effectiveness even for long-route instances.

The rest of the paper is structured as follows. Section 2 provides a review of some variable fixing techniques related to RCF. Section 3 describes preliminaries on the enumeration procedure and variable fixing techniques. Section 4 introduces the theoretical foundations and relevant formulations for dual picking, and gives an overview of DeLuxing. A detailed explanation of DeLuxing is provided in Section 5. Three extensions inspired by DeLuxing are presented in Section 6. We report the results of five sets of numerical experiments in Section 7. Finally, in Section 8, we make concluding remarks and identify potential avenues for future research. The detailed numerical results, the source code, and the data needed to reproduce these results can be found in the e-companion.

2. Literature Review

In this section, we review variable fixing techniques that rely on the well-known RCF, specifically focusing on those integrated into customized CG-based algorithms. Additionally, we discuss some recent efforts to enhance the effectiveness of RCF in more general settings. Finally, we review the current state-of-the-art exact methods for solving the CMTVRPTW, its two variants, CVRP, and VRPTW to facilitate the comprehension of our numerical results in Section 7.

The idea of what is now known as RCF was originally introduced in the seminal work Dantzig et al. (1954) for solving the traveling salesman problem. Its practical effectiveness and ease of implementation have made it a standard procedure in cutting-edge MIP solvers such as Gurobi (Achterberg 2018), CPLEX (Bixby et al. 2000), and SCIP (Achterberg et al. 2008). Moreover, RCF has been applied to leverage the strengths of MIP in a constraint programming (CP) framework (Yunes et al. 2010, Bacchus et al. 2017). Beyond MIP and CP, RCF has also been employed in two-stage stochastic programming (Crainic et al. 2018), semidefinite relaxation (Posta et al. 2012), and many others.

However, applying RCF to fix nominal variables in an extensive formulation solved by a CG-based method cannot be done blindly as it necessitates restructuring the pricing subproblem to prevent the regeneration of eliminated variables. Therefore, fixing by reduced costs is typically applied to implicit variables, which is equivalent to removing a subset of the variables in the RMP. Irnich et al. (2010) propose to use path-reduced cost to remove arcs from the underlying network of routing and scheduling problems without sacrificing optimality. They derive analytical results on which dual solutions *to the pricing subproblem* can achieve the largest reduced cost for an arc-flow variable under a given dual solution to the LP relaxation of the RMP. The authors conclude that approximately 80% of the arcs can be eliminated when the optimality gap is around 1%. This arc elimination technique has also been successfully applied in Pecin et al. (2017a) for solving the VRPTW.

Pessoa et al. (2010) and Pecin et al. (2017b) refine this approach to a resource-value-dependent arc elimination procedure for solving parallel machine scheduling problems and the CVRP, respectively. Using a similar approach, Sadykov et al. (2021) perform the so-called bucket arc elimination on a sophisticated way of organizing labels in the labeling algorithm called bucket graph. They report a 6% speedup compared to a standard arc elimination procedure independent of resources and conclude that hard instances with small primal-dual gaps benefit more from this new bucket arc elimination. Desaulniers et al. (2020) propose to generalize the idea to fix sequences of two arcs with a modification in the labeling algorithm for pricing. Experiments on the VRPTW and four variants of the electric VRPTW show that single-arc fixing can eliminate more than 90% of the feasible two-arc sequences, and two-arc sequence fixing can fix approximately half of the remaining ones, achieving an overall reduction of around 19% in the BPC computation time.

Enumeration, which identifies all potential columns with reduced costs not exceeding the current integrality gap, is another effective way to utilize RCF. This procedure has been employed in many high-performing BPC approaches (e.g., Yang 2023, Pessoa et al. 2020, Baldacci et al. 2013, 2011a,b,c,d) since its inception in Baldacci et al. (2008). After enumeration, RCF can be applied to the nominal variables in the conventional manner as columns are no longer generated by the labeling algorithm. Nonetheless, the efficacy of RCF is limited, especially at the current BBN, due to the three reasons explained in Section 1.1, which leaves room for improving RCF when applied in this way. It has been observed in Sellmann (2004) that distinct dual solutions can result in significantly different effectiveness and sub-optimal dual solutions could potentially result in even more variable fixing than optimal ones, which suggests a promising research avenue.

Bajgiran et al. (2017) take a step in exploring such improvement and propose to search for a dual solution maximizing the number of variables that can be fixed by solving a MIP. Their experiments demonstrate that the new dual picking method yields an average speedup of 20% in geometric mean over the default CPLEX. The authors also observe that by limiting the search to the optimal dual face instead of the entire dual feasible space, almost the same amount of fixing can be achieved while being orders of magnitude faster. However, solving the MIP constructed by Bajgiran et al. (2017) can be challenging as it includes n binary variables, where n is the number of variables in the original problem. The authors thus set a time limit of 10 minutes and use all feasible solutions obtained in the process for variable fixing. In Yang (2023), the author proposes to solve a new auxiliary problem that computes a second dual solution for fixing variables after enumeration in the price-cut-and-enumerate method (EPCEM) for the CMTVRPTW. Based on a similar idea, de Lima et al. (2023) develop two strategies to compute alternative dual solutions for variable fixing when dealing with network flow models. It is worth mentioning that since they consider the DWD reformulation, the variable fixing is conducted on the arcs of the underlying network. These methods can be performed

iteratively, with each iteration building upon the previous round of variable fixing. They demonstrate that these techniques speed up the proof of optimality despite their high computational overhead. Lastly, Mingozzi et al. (2013) propose four bounding procedures to enhance lower bounds in solving multi-trip VRPs, where the dual solution from each procedure is additionally utilized for RCF. Note that they do not actively search for new dual solutions for variable fixing purposes.

Our proposed DeLuxing method eliminates nominal variables in the RMP via multiple dual solutions. It fundamentally differs from existing approaches in several key aspects. First, DeLuxing uses a novel small-sized LP formulation to search for qualified dual solutions that are not restricted to be (sub)-optimal or feasible for the original dual problem. Second, it uses a completely new way of searching that exploits the underlying column similarities revealed iteratively. Last, DeLuxing does not rely on specific problem structures, unlike previous approaches such as those in de Lima et al. (2023), making it generally applicable to both CG-based exact methods and MIPs.

The CMTVRPTW is an increasingly popular extension of the well-known capacitated VRPTW by allowing each vehicle to perform multiple trips, catering to real-world scenarios with constraints such as limited vehicles, driver availability, vehicle capacity, or trip duration (Cattaruzza et al. 2016b). In this study, we also consider two notable variants of the CMTVRPTW: the CMTVRPTW *with loading times* (CMTVRPTW-LT; Hernandez et al. 2016) and CMTVRPTW *with release dates* (CMTVRPTW-R; Cattaruzza et al. 2016a). In the CMTVRPTW-LT, a loading time, computed as a fixed percentage of the total service time of all customers visited in the route, is required at the depot before a vehicle can start the trip. In the CMTVRPTW-R, the requested goods by each customer become available at the depot at possibly different times, called release dates. Consequently, a vehicle cannot depart from the depot until all the goods to be aboard are available, i.e., its departure time should be no earlier than the latest release date among the goods it ships. Although the loading times and release dates do not affect the objective, which is to minimize the total distance traveled by all vehicles, they impact the solution space and thus the problem difficulty.

Paradiso et al. (2020) develop the first exact solution framework (ESF) capable of solving the CMTVRPTW and its four variants including the CMTVRPTW-LT and CMTVRPTW-R. The ESF, operating as a BPC type of algorithm, solves a novel structure-based formulation, which significantly outperforms all previously leading exact methods such as those proposed by Hernandez et al. (2014, 2016), and Cheng et al. (2020). Inspired by Paradiso et al. (2020), Yang (2023) introduce a superstructure-based formulation with fewer variables while maintaining the same theoretical tightness. The EPCEM developed therein demonstrates substantially superior numerical performance compared to the ESF. Two very recent studies achieve remarkable advancements, representing the state-of-the-art. Roboredo et al. (2023) propose a new graph-based model, effectively solved by the VRP-Solver developed in Pessoa et al. (2020) with a simple parameterization. Zhang (2022) propose

a novel three-phase exact method for solving all five problems. In the initial two phases, a route-based and a trip-based model are solved, respectively, to obtain tight lower bounds, followed by solving a trip-based model with dynamic time discretization in the last phase. In contrast, our approach works only with route-based models. Specifically, incorporating DeLuxing and the three extensions developed in this paper into the EPCEM (see Section 7.2), we obtain an enhanced exact method, denoted as Default, expanding the frontiers of this field even further. For the classic CVRP and VRPTW, the VRP-Solver (Pessoa et al. 2020) consistently achieves one of the best results in the literature and thus serves as a benchmark for experiments in Section 7.5.

3. Preliminaries

In this section, we first formally describe the enumeration procedure that is now widely applied in the BPC framework for solving challenging discrete optimization problems. Then, we review the general variable fixing technique by Lagrangian bounds. Lastly, we discuss a special variable fixing strategy via dual picking introduced in Yang (2023) and its major drawbacks, which serve as a natural motivation for this study. Throughout the paper, we use $\mathbb{R}_+$, $\mathbb{Z}$, and $\mathbb{Z}_+$ to denote the set of non-negative real numbers, integers, and non-negative integers, respectively. Little letters in bold are used to represent vectors. The inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$.

3.1. The Enumeration Procedure

Consider the following extensive formulation that can be a standard set partitioning formulation or a DWD reformulation. Since the proposed method will be applied exclusively to fix integer variables, we omit continuous variables in the presentation without loss of generality. Moreover, we only consider that all variables are non-negative in our presentation. This requirement is not necessary and can be removed through a straightforward variable substitution.

$$
\begin{aligned}
\mathrm{F}(\mathcal{R}): \qquad z^* \;=\; \min \quad & \sum_{r \in \mathcal{R}} c_r x_r & & \\
\text{s.t.} \quad & \sum_{r \in \mathcal{R}} a_{ir} x_r = b_i, & & \forall i \in \mathcal{N}, \\
& x_r \in \mathbb{Z}_+, & & \forall r \in \mathcal{R}.
\end{aligned}
$$

In the context of VRPs, $\mathcal{N}$ represents the set of customers, each of which should be visited exactly once (i.e., $b_i = 1$ for $i \in \mathcal{N}$), $\mathcal{R}$ consists of all feasible routes and possibly some relaxed routes that are not necessarily feasible (e.g., *ng*-routes from Baldacci et al. 2012), x_r is a binary decision variable taking the value of one if route $r \in \mathcal{R}$ is used and zero otherwise, c_r and a_{ir} denote the cost and the number of times customer i is visited by route r, respectively. Due to the exponential size of $\mathcal{R}$, formulation $\mathrm{F}(\mathcal{R})$ is typically solved by a BPC method (Fukasawa et al. 2006, Pecin et al. 2017b).

Let lb and ub, separately, be a lower bound and an upper bound of the optimal value z^*. An lb is usually obtained by solving some LP relaxations of $\mathrm{F}(\mathcal{R})$, and an ub is usually set to the objective value of the best feasible solution found so far. The optimality gap, denoted by g, is computed as the difference between ub and lb, i.e., $g := ub - lb$. A BPC method can try to enumerate all variables with reduced costs no larger than g with respect to (w.r.t.) the current dual solution when the gap g falls below some prespecified threshold. This idea was first proposed for solving VRPs in Baldacci et al. (2008) and has been successfully applied in most state-of-the-art BPC methods.

A labeling algorithm, slightly modified from the one used for generating columns in the pricing procedure, is typically employed to enumerate all qualified columns. Specifically, for VRPs, the dominance rule must be changed to allow dominance only between partial routes (labels) that have visited exactly the same subset of customers, ensuring that no qualified routes are missed. Additionally, the focus should be solely on elementary routes, as relaxed routes (e.g., ng-routes) that visit some customers more than once are not feasible. More details can be found in Baldacci et al. (2008) and Pecin et al. (2017b). However, due to this weaker dominance rule and the requirement for elementariness, the enumeration process generally requires significantly more time than the pricing procedure. Arc elimination based on RCF (Irnich et al. 2010, Pessoa et al. 2020) can substantially reduce the network size, and therefore, performing enumeration on this reduced network can significantly shorten the computational time. This arc elimination has been implemented in RouteOpt for the CVRP and VRPTW. However, we did not apply it in our code for solving the CMTVRPTW and its two variants since enumeration only takes up a noticeable portion of the total time in easy instances that can be solved optimally within 5 minutes. We believe arc elimination has the potential to further accelerate the solution process for these instances and will be considered in future studies.

Let $\underline{\mathcal{R}}$ denote the set of variables that have been enumerated. In the case of VRPs, the set $\underline{\mathcal{R}}$ only consists of qualified elementary routes. RCF guarantees that solving $\mathrm{F}(\underline{\mathcal{R}})$ will yield an optimal solution to $\mathrm{F}(\mathcal{R})$ because a variable with a reduced cost greater than g cannot take a positive integer value in any optimal solution. When the cardinality of $\underline{\mathcal{R}}$ is in the tens of thousands, solving $\mathrm{F}(\underline{\mathcal{R}})$ as an IP by a general MIP solver such as Gurobi (Gurobi Optimization, LLC 2023) can yield an optimal solution within a reasonable time frame. In case of $|\underline{\mathcal{R}}|$ being too large, the algorithm can continue the BPC procedure using inspection for CG (Contardo and Martinelli 2014). Specifically, instead of running the dynamic programming-based labeling algorithm, which can be computationally intensive, especially when many non-robust cuts (Pecin et al. 2017b) have been added, pricing is done by evaluating the reduced costs of the columns in the pool. In both cases, the pool size significantly impacts the time required to prove optimality.

3.2. Variable Fixing by Lagrangian Bounds

A natural way to reduce the computational burden after enumeration is to remove variables from $\mathrm{F}(\underline{\mathcal{R}})$. Consider the following LP relaxation of $\mathrm{F}(\underline{\mathcal{R}})$ with cutting planes added in the solution process.

$$\begin{aligned}
\overline{\mathrm{F}}(\underline{\mathcal{R}}): \qquad \bar{z}^* \;=\; \min \quad & \sum_{r\in\underline{\mathcal{R}}} c_r x_r & & \\
\text{s.t.} \quad & \sum_{r\in\underline{\mathcal{R}}} a_{ir} x_r = b_i, & \forall i\in\mathcal{N}, & \qquad (1)\\
& \sum_{r\in\underline{\mathcal{R}}} a_{kr} x_r \leqslant b_k, & \forall k\in\mathcal{K}, & \qquad (2)\\
& x_r \geqslant 0, & \forall r\in\underline{\mathcal{R}}, &
\end{aligned}$$

where $\mathcal{K}$ denotes the index set of the added cuts. For VRPs, they typically include the rounded capacity cuts (RCCs; Laporte and Nobert 1983, Lysgaard et al. 2004), the (limited memory) subset row cuts (SRCs; Jepsen et al. 2008, Pecin et al. 2017b) and problem-specific feasibility cuts, e.g., the relaxed (super)structure feasibility cuts for the CMTVRPTW (Yang 2023, Paradiso et al. 2020).

Fixing variables by Lagrangian bounds is a widely applied technique in solving discrete optimization problems (e.g., Balas and Saltzman 1991, Balas and Carrera 1996, Holmberg and Yuan 2000). The general idea is that when a variable is set to a given value, if the Lagrangian bound is larger than the best upper bound, then this value can be excluded from the variable's feasible region. More precisely, consider the following Lagrangian dual function obtained from $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ by dualizing the constraints.

$$\mathcal{L}(\mathbf{y},\mathbf{x}) \;=\; \sum_{i\in\mathcal{I}} b_i y_i + \sum_{r\in\underline{\mathcal{R}}} \left(c_r - \sum_{i\in\mathcal{I}} a_{ir} y_i \right) x_r,$$

where $\mathcal{I} := \mathcal{N} \cup \mathcal{K}$, y_i for $i \in \mathcal{I}$ are the dual variables associated with constraints (1) and (2), $\mathbf{y} = (y_i)_{i\in\mathcal{I}}$, and $\mathbf{x} = (x_r)_{r\in\underline{\mathcal{R}}}$. For convenience, we define the set $\mathcal{Y} := \{\mathbf{y} \in \mathbb{R}^{|\mathcal{I}|} : y_k \leqslant 0,\ \forall k \in \mathcal{K}\}$. Let $\overline{\mathrm{F}}(\underline{\mathcal{R}})|_{x_j=v}$ be the formulation obtained by adding an additional constraint $x_j = v$ to $\overline{\mathrm{F}}(\underline{\mathcal{R}})$, and $\bar{z}^*|_{x_j=v}$ be its optimal value, which is set to $+\infty$ in case of infeasibility. Due to LP weak duality, it follows that $\max_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{x}\geqslant 0, x_j=v} \mathcal{L}(\mathbf{y},\mathbf{x}) \leqslant \bar{z}^*|_{x_j=v}$. If, for any given dual vector $\hat{\mathbf{y}} \in \mathcal{Y}$, we have $\min_{\mathbf{x}\geqslant 0, x_j=v} \mathcal{L}(\hat{\mathbf{y}},\mathbf{x}) > ub$, then it immediately leads to $\bar{z}^*|_{x_j=v} \geqslant \max_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{x}\geqslant 0, x_j=v} \mathcal{L}(\mathbf{y},\mathbf{x}) \geqslant \min_{\mathbf{x}\geqslant 0, x_j=v} \mathcal{L}(\hat{\mathbf{y}},\mathbf{x}) > ub$. Therefore, x_j cannot equal v in any optimal solution to $\mathrm{F}(\underline{\mathcal{R}})$. RCF can be viewed as a special case of this general technique. Specifically, if $\min_{\mathbf{x}\geqslant 0, x_j=1} \mathcal{L}(\mathbf{y}^*,\mathbf{x}) > ub$ for an optimal dual solution $\mathbf{y}^*$, then the binary variable x_j can be fixed to 0 and thus removed.

3.3. Variable Fixing by Dual Picking

The Lagrangian bound $\min_{\mathbf{x}\geqslant 0, x_j=1} \mathcal{L}(\mathbf{y},\mathbf{x})$ depends on the dual $\mathbf{y}$ used and it reduces to $\bar{z}^* + \bar{c}_j + \min_{\mathbf{x}\geqslant 0} \sum_{r\in\underline{\mathcal{R}}} \bar{c}_r x_r$ when $\mathbf{y}$ is an optimal dual solution to $\overline{\mathrm{F}}(\underline{\mathcal{R}})$, where $\bar{c}_r = c_r - \sum_{i\in\mathcal{I}} a_{ir} y_i = c_r - \langle \mathbf{y}, \mathbf{a}_r \rangle$

is the corresponding reduced cost of x_r, and $\mathbf{a}_r = (a_{ir})_{i\in\mathcal{I}}$. Let $\mathcal{Y}^*$ be the set of optimal dual solutions to $\overline{\mathrm{F}}(\underline{\mathcal{R}})$. In Yang (2023), the author proposes to pick a special point in $\mathcal{Y}^*$ to obtain large reduced costs, thereby fixing a large number of columns to 0. This involves solving the following LP, denoted by DF, that maximizes the sum of the reduced costs of all variables. According to the LP duality theorem, it is equivalent to solving AUX in the primal space (see Section 6.5 of Yang 2023 for details).

$$
(\mathrm{DF}): \begin{cases} \max & \displaystyle\sum_{r\in\underline{\mathcal{R}}}\left(c_r - \sum_{i\in\mathcal{I}} a_{ir}y_i\right) \\ \text{s.t.} & \displaystyle\sum_{i\in\mathcal{I}} a_{ir}y_i \leqslant c_r, \quad \forall r\in\underline{\mathcal{R}}, \\ & \displaystyle\sum_{i\in\mathcal{I}} b_i y_i = \bar{z}^*, \\ & y_i \leqslant 0, \quad \forall i\in\mathcal{K}. \end{cases}
\qquad
(\mathrm{AUX}): \begin{cases} \min & \displaystyle\sum_{r\in\underline{\mathcal{R}}} c_r(x_r+1) + \bar{z}^* w \\ \text{s.t.} & \displaystyle\sum_{r\in\underline{\mathcal{R}}} a_{ir}x_r + b_i w = -\sum_{r\in\underline{\mathcal{R}}} a_{ir}, & \forall i\in\mathcal{N}, \\ & \displaystyle\sum_{r\in\underline{\mathcal{R}}} a_{kr}x_r + b_k w \leqslant -\sum_{r\in\underline{\mathcal{R}}} a_{kr}, & \forall k\in\mathcal{K}, \\ & x_r \geqslant 0, & \forall r\in\underline{\mathcal{R}}. \end{cases}
$$

3.3.1. Major Drawbacks The above approach has several drawbacks. First, by design, AUX searches within the dual optimal face, using a dual solution from $\mathcal{Y}^*$ to update the reduced costs. However, $\mathcal{Y}^*$ only constitutes a small portion of all feasible dual solutions to $\overline{\mathrm{F}}(\underline{\mathcal{R}})$, so the number of variables that can be removed by solving AUX may be limited. Second, solving AUX, possibly with different objective coefficients, to obtain multiple dual solutions can lead to more variable fixings. However, AUX has $(|\underline{\mathcal{R}}|+1)$ variables, which can easily top tens of millions for challenging instances, making AUX time-consuming and memory-intensive to solve, particularly for interior point methods that are known to outperform the simplex method for large-sized LPs. Consequently, it is computationally prohibitive to repeatedly solve AUX with varied objective coefficients.

In fact, for each individual $r\in\underline{\mathcal{R}}$, we want to maximize the reduced cost $\bar{c}_r$, which can be achieved by solving an AUX with $\bar{c}_r$ as the objective function. Thus, it requires solving AUX by a total of $|\underline{\mathcal{R}}|$ times and is computationally intractable. Instead, maximizing the sum $\sum_{r\in\underline{\mathcal{R}}}\bar{c}_r$ can be viewed as a coarse approximation that works reasonably well when the set of $|\mathcal{I}|$-dimensional vectors, $\{-\mathbf{a}_r:\ r\in\underline{\mathcal{R}}\}$, have some nice structure. For example, when they are close to the ray generated by $\mathbf{y}$ as depicted in the left subfigure of Figure 1, where $\mathbf{y}$ is an extreme point of the polyhedron $\mathcal{Y}^*$, almost all reduced costs $\bar{c}_r$ are maximized at the same extreme point $\mathbf{y}$.

However, this approach can be problematic, particularly when $-\mathbf{a}_r$ for $r\in\underline{\mathcal{R}}$ are scattered in the $|\mathcal{I}|$-dimensional Euclidean space. Let $\mathbf{o}' := \frac{1}{|\underline{\mathcal{R}}|}\sum_{r\in\underline{\mathcal{R}}}(-\mathbf{a}_r)$ be the center and $r' := \max_{r\in\underline{\mathcal{R}}}\|-\mathbf{a}_r-\mathbf{o}'\|$ be the radius. Maximizing the sum $\sum_{r\in\underline{\mathcal{R}}}\bar{c}_r$ essentially maximizes the inner product $\langle\mathbf{y},\mathbf{o}'\rangle$ for $\mathbf{y}\in\mathcal{Y}^*$, which is achieved at extreme point $\mathbf{y}^1$. However, as shown in the right subfigure of Figure 1, when the radius r' is relatively large, $\mathbf{y}^1$ may not be the maximizer for a majority of $\bar{c}_r$. For instance, all the purple and yellow points are maximized at extreme points $\mathbf{y}^2$ and $\mathbf{y}^3$, respectively. As a result, some variables could have been fixed if a better dual solution, such as $\mathbf{y}^2$ or $\mathbf{y}^3$ in this example, had been used to compute the reduced costs.

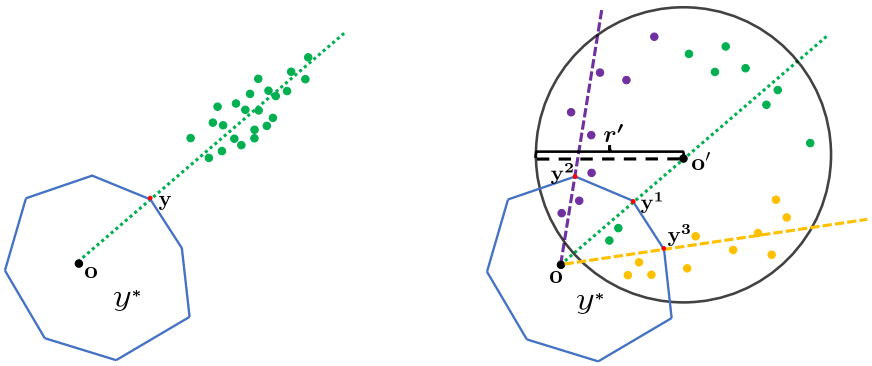

Figure 1 An example illustrating that a direct maximization of the sum of all reduced costs can be problematic, where $\mathbf{o}$ represents the origin, $\mathcal{Y}^*$ represents the feasible region of DF, $\mathbf{y}$, $\mathbf{y}^1$, $\mathbf{y}^2$, and $\mathbf{y}^3$ are extreme points of $\mathcal{Y}^*$, and the points in green, purple, and yellow represent $-\mathbf{a}_r$ for some $r \in \underline{\mathcal{R}}$.

4. The DeLuxing

The proposed DeLuxing aims to overcome the previously mentioned limitations. More specifically, DeLuxing enables the removal of variables by using LUs computed with dual solutions that are not necessarily optimal and may even be infeasible, which enlarges the search space substantially. In this process, a sequence of carefully crafted LPs of much smaller size than the AUX is solved instead of just solving a single AUX, significantly increasing the chance of an unnecessary variable being removed. According to our numerical experiments detailed in Section 7, DeLuxing can remove more than 75% of the columns in most cases, reducing $\underline{\mathcal{R}}$ to a quarter or less of its original size.

4.1. Theoretical Foundations

Constructing small-sized LPs to obtain multiple dual solutions fast is one of the key ideas behind DeLuxing, which is motivated by the observation that the number of variables with reduced costs no greater than $\frac{g}{2}$ only comprises a small proportion (mostly less than 15%) of the elements in $\underline{\mathcal{R}}$. This ratio is observed to be even smaller for larger instances. In other words, $|\underline{\mathcal{R}}_2^{\leqslant}|$ is much smaller than $|\underline{\mathcal{R}}|$, where $\underline{\mathcal{R}}_2^{\leqslant} := \{r \in \underline{\mathcal{R}} : \bar{c}_r^{\boldsymbol{\pi}} \leqslant \frac{g}{2}\}$, $\boldsymbol{\pi}$ is a given optimal dual solution to $\overline{\mathrm{F}}(\underline{\mathcal{R}})$, and $\bar{c}_r^{\boldsymbol{\pi}} = c_r - \langle \boldsymbol{\pi}, \mathbf{a}_r \rangle$ is the reduced cost of variable x_r w.r.t. $\boldsymbol{\pi}$. Although we use the simplified notation $\underline{\mathcal{R}}_2^{\leqslant}$ that does not explicitly show its dependence on $\boldsymbol{\pi}$, it should not cause any confusion. Once $\underline{\mathcal{R}}_2^{\leqslant}$ is computed using a given $\boldsymbol{\pi}$, it will not be recomputed w.r.t. another dual solution unless otherwise specified. Thus, it is expected that substantial acceleration will be achieved if the computation can be performed using solely variables x_r for r in set $\underline{\mathcal{R}}_2^{\leqslant}$ instead of the whole set $\underline{\mathcal{R}}$. This is made possible by the following Lemma 1 (Proposition 4 from Yang 2023), which ensures that any optimal solution to $\mathrm{F}(\underline{\mathcal{R}})$ can have $(k-1)$ variables with reduced costs larger than $\frac{g}{k}$ taking positive integer values.

LEMMA 1 **(Proposition 4 in Yang 2023)**. *For any given positive integer k, the inequality $\sum_{r \in \underline{\mathcal{R}}_k^{>}} x_r \leqslant k-1$ is valid for $\mathrm{F}(\underline{\mathcal{R}})$, where $\underline{\mathcal{R}}_k^{>} := \{r \in \underline{\mathcal{R}} : \bar{c}_r^{\boldsymbol{\pi}} > \frac{g}{k}\}$.*

Evidently, Lemma 1 also reduces to the standard RCF when $k = 1$. We consider the case when $k = 2$, and work with the formulation $\overline{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant})$ obtained from $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ with the set of variables x_r for $r \in \underline{\mathcal{R}}$ replaced by $r \in \underline{\mathcal{R}}_2^{\leqslant}$. Let $\mathcal{Y}^{\boldsymbol{\pi}}$ be the set of all feasible dual solutions to $\overline{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant})$, i.e., $\mathcal{Y}^{\boldsymbol{\pi}} := \{\mathbf{y} \in \mathbb{R}^{|\mathcal{I}|} : \sum_{i\in\mathcal{I}} a_{ir} y_i \leqslant c_r,\ \forall r \in \underline{\mathcal{R}}_2^{\leqslant},\ y_i \leqslant 0,\ \forall i \in \mathcal{K}\}$.

PROPOSITION 1. *For any given $\hat{\mathbf{y}} \in \mathcal{Y}^{\boldsymbol{\pi}}$ and $j \in \underline{\mathcal{R}}_2^{>}$, if $\langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle > ub - c_j$ is satisfied, where $\mathbf{f}^j = (b_i - a_{ij})_{i\in\mathcal{I}}$, then variable x_j can be removed from formulation* $\mathrm{F}(\underline{\mathcal{R}})$.

Proof Let $\mathrm{F}'(\underline{\mathcal{R}})$ be the formulation obtained from $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ by adding the additional constraint $\sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1$. Due to Lemma 1, we know $\sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1$ is valid for $\mathrm{F}(\underline{\mathcal{R}})$. Therefore, $\mathrm{F}'(\underline{\mathcal{R}})$ can be viewed as a relaxation of $\mathrm{F}(\underline{\mathcal{R}})$. Let $z'|_{x_j=1}$ and $z^*|_{x_j=1}$ be the optimal value of $\mathrm{F}'(\underline{\mathcal{R}})$ and $\mathrm{F}(\underline{\mathcal{R}})$, respectively, when $x_j = 1$ is enforced for a given $j \in \underline{\mathcal{R}}_2^{>}$. Then we have $z'|_{x_j=1} \leqslant z^*|_{x_j=1}$. For convenience, we define $\bar{c}_r^{\hat{\mathbf{y}}} := c_r - \langle \hat{\mathbf{y}}, \mathbf{a}_r \rangle$. Note that $\underline{\mathcal{R}}_2^{\leqslant} = \underline{\mathcal{R}} \backslash \underline{\mathcal{R}}_2^{>}$.

$$\begin{aligned}
\min_{\substack{\mathbf{x}\geqslant 0, x_j=1,\\ \sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{x}) &= \min_{\substack{\mathbf{x}\geqslant 0, x_j=1,\\ \sum_{r\in\underline{\mathcal{R}}_2^{>}\backslash\{j\}} x_r = 0}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{x}) \\
&= \sum_{i\in\mathcal{I}} b_i \hat{y}_i + \min_{\substack{\mathbf{x}\geqslant 0, x_j=1,\\ \sum_{r\in\underline{\mathcal{R}}_2^{>}\backslash\{j\}} x_r = 0}} \sum_{r\in\underline{\mathcal{R}}} \left(c_r - \sum_{i\in\mathcal{I}} a_{ir} \hat{y}_i \right) x_r \\
&= \sum_{i\in\mathcal{I}} b_i \hat{y}_i + \bar{c}_j^{\hat{\mathbf{y}}} + \min_{x_r\geqslant 0, \forall r\in\underline{\mathcal{R}}_2^{\leqslant}} \sum_{r\in\underline{\mathcal{R}}_2^{\leqslant}} \bar{c}_r^{\hat{\mathbf{y}}} x_r \\
&\geqslant \sum_{i\in\mathcal{I}} b_i \hat{y}_i + \bar{c}_j^{\hat{\mathbf{y}}} = c_j + \sum_{i\in\mathcal{I}} (b_i - a_{ij}) \hat{y}_i = c_j + \langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle > ub
\end{aligned}$$

where the inequality is due to $\bar{c}_r^{\hat{\mathbf{y}}} \geqslant 0$ for $r \in \underline{\mathcal{R}}_2^{\leqslant}$, given $\hat{\mathbf{y}}$ is a feasible dual to $\overline{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant})$. Consequently,

$$z^*|_{x_j=1} \geqslant z'|_{x_j=1} = \bar{z}^*|_{x_j=1,\ \sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1} \geqslant \max_{\mathbf{y}\in\mathcal{Y}} \min_{\substack{\mathbf{x}\geqslant 0, x_j=1,\\ \sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1}} \mathcal{L}(\mathbf{y}, \mathbf{x}) \geqslant \min_{\substack{\mathbf{x}\geqslant 0, x_j=1,\\ \sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{x}) > ub,$$

where $\bar{z}^*|_{x_j=1,\ \sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1}$ is the optimal value of $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ when $x_j = 1$ and $\sum_{r\in\underline{\mathcal{R}}_2^{>}} x_r \leqslant 1$ are enforced, and the second inequality is due to weak duality. We conclude that any feasible solution to $\mathrm{F}(\underline{\mathcal{R}})$ with x_j equal to 1 must have an objective value larger than ub, and thus x_j can be removed from the formulation, which completes the proof. ■

Remarks: According to Proposition 1, any $\hat{\mathbf{y}} \in \mathcal{Y}^{\boldsymbol{\pi}}$ can be used for removing unnecessary variables, even if it may be infeasible to the dual of $\overline{\mathrm{F}}(\underline{\mathcal{R}})$. It provides an easily verifiable criterion to decide if a variable x_j for $j \in \underline{\mathcal{R}}_2^{>}$ can be removed for a given $\hat{\mathbf{y}}$. Note that vectors $\mathbf{f}^j$ and values $ub - c_j$ for all $j \in \underline{\mathcal{R}}_2^{>}$ need to be calculated only once and can be reused throughout the computation. Upon obtaining a new feasible dual solution $\hat{\mathbf{y}}$ to $\overline{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant})$, it suffices to compute the inner product $\langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle$ and make the comparison. Two distinct variables, $x_i, x_j \in \underline{\mathcal{R}}$, are said to be compatible if there exists a feasible solution to $\mathrm{F}(\underline{\mathcal{R}})$ with both x_i and x_j equal to 1. In other words, x_i and x_j are incompatible if $x_i + x_j \leqslant 1$ is valid for $\mathrm{F}(\underline{\mathcal{R}})$. The following Proposition 2 provides a sufficient condition for removing a variable x_j for $j \in \underline{\mathcal{R}}_2^{\leqslant}$.

PROPOSITION 2. *For any given* $\hat{\mathbf{y}} \in \mathcal{Y}^{\boldsymbol{\pi}}$ *and* $j \in \underline{\mathcal{R}}_2^{\leqslant}$, *if* $\langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle > ub - c_j - \min\{\eta_j, 0\}$ *is satisfied, where* $\eta_j = \min_{r \in \mathcal{S}^j}\{c_r - \langle \hat{\mathbf{y}}, \mathbf{a}_r \rangle\}$ *and* $\mathcal{S}^j = \{r \in \underline{\mathcal{R}}_2^{>} : x_r$ *is compatible with* $x_j\}$, *then variable* x_j *can be removed from formulation* $\mathrm{F}(\underline{\mathcal{R}})$.

Proof We use the notation defined in the above proof of Proposition 1. Note that $j \in \underline{\mathcal{R}}_2^{\leqslant}$ in this case. Additionally, let $\overline{\mathcal{S}^j} = \underline{\mathcal{R}}_2^{>} \backslash \mathcal{S}^j$. By the definition of $\mathcal{S}^j$, it follows that for any $j' \in \overline{\mathcal{S}^j}$, $x_j + x_{j'} \leqslant 1$ is valid for $\mathrm{F}(\underline{\mathcal{R}})$. Let $\mathrm{F}''(\underline{\mathcal{R}})$ be the formulation obtained from $\mathrm{F}'(\underline{\mathcal{R}})$ by adding the additional constraints $x_j + x_{j'} \leqslant 1, \forall j' \in \overline{\mathcal{S}^j}$. Let $\mathcal{X} := \left\{\mathbf{x} \in \mathbb{R}_+^{|\underline{\mathcal{R}}|} : \sum_{r \in \underline{\mathcal{R}}_2^{>}} x_r \leqslant 1, x_j = 1, x_j + x_{j'} \leqslant 1, \forall j' \in \overline{\mathcal{S}^j}\right\} = \left\{\mathbf{x} \in \mathbb{R}_+^{|\underline{\mathcal{R}}|} : \sum_{r \in \underline{\mathcal{R}}_2^{>}} x_r \leqslant 1, x_j = 1, x_{j'} = 0, \forall j' \in \overline{\mathcal{S}^j}\right\}$, and $\mathcal{X}' := \mathcal{X} \cap \{\mathbf{x} \in \mathbb{R}_+^{|\underline{\mathcal{R}}|} : x_r = 0, \forall r \in \underline{\mathcal{R}}_2^{\leqslant}, r \neq j\}$. Let $z''|_{x_j=1}$ be the optimal value of $\mathrm{F}''(\underline{\mathcal{R}})$ when $x_j = 1$ is enforced. Since $\mathrm{F}''(\underline{\mathcal{R}})$ again can be view as a relaxation of $\mathrm{F}(\underline{\mathcal{R}})$, we have $z''|_{x_j=1} \leqslant z^*|_{x_j=1}$.

$$\begin{aligned}
\min_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{x}) &= \sum_{i \in \mathcal{I}} b_i \hat{y}_i + \min_{\mathbf{x} \in \mathcal{X}} \sum_{r \in \underline{\mathcal{R}}} \left(c_r - \sum_{i \in \mathcal{I}} a_{ir} \hat{y}_i\right) x_r \\
&\geqslant \sum_{i \in \mathcal{I}} b_i \hat{y}_i + \min_{\mathbf{x} \in \mathcal{X}'} \sum_{r \in \underline{\mathcal{R}}} \bar{c}_r^{\hat{\mathbf{y}}} x_r \\
&= \sum_{i \in \mathcal{I}} b_i \hat{y}_i + \bar{c}_j^{\hat{\mathbf{y}}} + \min_{\substack{x_r \geqslant 0, \forall r \in \underline{\mathcal{R}}_2^{>}, \\ \sum_{r \in \mathcal{S}^j} x_r \leqslant 1}} \sum_{r \in \underline{\mathcal{R}}_2^{>}} \bar{c}_r^{\hat{\mathbf{y}}} x_r \\
&= \sum_{i \in \mathcal{I}} b_i \hat{y}_i + \bar{c}_j^{\hat{\mathbf{y}}} + \min\left\{0, \min_{r \in \mathcal{S}^j} \bar{c}_j^{\hat{\mathbf{y}}}\right\} \\
&= c_j + \langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle + \min\left\{0, \min_{r \in \mathcal{S}^j} \bar{c}_j^{\hat{\mathbf{y}}}\right\} > ub
\end{aligned}$$

where the first inequality is again due to the fact that $\bar{c}_r^{\hat{\mathbf{y}}} \geqslant 0$ for $r \in \underline{\mathcal{R}}_2^{\leqslant}$. Finally,

$$z^*|_{x_j=1} \geqslant z''|_{x_j=1} = \bar{z}^*|_{\mathbf{x} \in \mathcal{X}} \geqslant \max_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\mathbf{y}, \mathbf{x}) \geqslant \min_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{x}) > ub,$$

where $\bar{z}^*|_{\mathbf{x} \in \mathcal{X}}$ is the optimal value of $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ when $\mathbf{x}$ is restricted to $\mathcal{X}$. Therefore, any feasible solution to $\mathrm{F}(\underline{\mathcal{R}})$ with x_j equal to 1 must have an objective value larger than ub, and thus x_j can be removed from the formulation, which completes the proof. ■

Remarks: Applying Proposition 2 requires identifying variables with index in $\underline{\mathcal{R}}_2^{>}$ that can take a positive value simultaneously with x_j. For VRPs, $\mathcal{S}^j$ can be defined as the set $\{r \in \underline{\mathcal{R}}_2^{>} : a_{ir} + a_{ij} \leqslant 1, \forall i \in \mathcal{N}\}$, which essentially identifies routes in $\underline{\mathcal{R}}_2^{>}$ that do not conflict with the given route j while ensuring that each customer is visited only once. It is worth mentioning that if additional information, such as time windows for the CMTVRPTW and battery constraints for EV or drone routing (e.g., Desaulniers et al. 2016a, Roberti and Ruthmair 2021), is available to tell that a route $j' \in \underline{\mathcal{R}}_2^{>}$ is incompatible with route j, then it can be removed from $\mathcal{S}^j$. As a result, more variables might be removed from $\mathrm{F}(\underline{\mathcal{R}})$ because the condition in Proposition 2 becomes easier to satisfy. In addition, the conflict graph constructed in this process can help solve $\mathrm{F}(\underline{\mathcal{R}})$ as an IP at the end. To accelerate

each iteration, it is possible to skip computing $\mathcal{S}^j$ exactly and set $\mathcal{S}^j = \underline{\mathcal{R}}_2^>$ instead, which potentially leads to fewer variables being removed each time. Since each iteration is faster now, we can afford to run more iterations and thus find more feasible dual solutions to $\overline{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant})$ for variable fixing.

4.2. Novel LP Formulation for Dual Picking

In this paper, we refer to $\ell_j^{\mathbf{y}} := c_j + \langle \mathbf{f}^j, \mathbf{y} \rangle$ as the Lagrangian underestimate of variable x_j w.r.t. $\mathbf{y}$ and use the number of variables deemed removable by $\boldsymbol{\ell}^{\mathbf{y}} = (\ell_j^{\mathbf{y}})_{j \in \underline{\mathcal{R}}}$ as a measure of the quality of $\mathbf{y}$. Our numerical experiments show that $\boldsymbol{\ell}^{\mathbf{y}}$ significantly differs depending on $\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}$, and thus the quality of $\mathbf{y}$ varies substantially, which aligns with the observation from Sellmann (2004).

4.2.1. High Level Idea Finding a $\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}$ of the best quality is NP-hard in general because it involves satisfying the maximum number of linear constraints defined in Propositions 1 and 2, which is equivalent to solving a generalized maximum feasible subsystem problem that is known to be NP-hard (Amaldi and Kann 1995). Nonetheless, finding a single best-quality $\mathbf{y}$ is overkill since we do not have to be restricted to using a single $\mathbf{y}$ for this purpose. In the extreme case, we can solve $\max_{\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}} \langle \mathbf{f}^j, \mathbf{y} \rangle$ for each $j \in \underline{\mathcal{R}}$ to decide individually if x_j can be removed, which requires solving $|\underline{\mathcal{R}}|$ linear programs in total and is polynomial in time complexity. This suggests that we should use multiple $\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}$ to compute different LUs. Now the question becomes how to efficiently obtain multiple $\mathbf{y}$ from $\mathcal{Y}^{\boldsymbol{\pi}}$ that yield high-quality LUs.

Based on the discussion in Section 3.3.1, we propose to iteratively identify a subset $\mathcal{J}$ of $\underline{\mathcal{R}}$ such that the vectors $\mathbf{f}^j$ for $j \in \mathcal{J}$ are close to each other, and then compute $\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}$ that maximizes the inner product $\langle \sum_{j \in \mathcal{J}} \mathbf{f}^j, \mathbf{y} \rangle$. The intuition is that when the vectors $\mathbf{f}^j$ for $j \in \mathcal{J}$ are sufficiently similar, a solution $\mathbf{y}$ maximizing $\sum_{j \in \mathcal{J}} \langle \mathbf{f}^j, \mathbf{y} \rangle$ is likely to also achieve a close-to-maximum value for each individual inner product $\langle \mathbf{f}^j, \mathbf{y} \rangle$, leading to LUs that can potentially eliminate many variables.

4.2.2. Potential Issue and Fix The unboundedness of $\mathcal{Y}^{\boldsymbol{\pi}}$ implies that there might exist $j \in \mathcal{J}$ such that $\max_{\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}} \langle \mathbf{f}^j, \mathbf{y} \rangle$ goes to $+\infty$. In this case, $d^{\mathcal{J}} := \max_{\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}} \sum_{j \in \mathcal{J}} \langle \mathbf{f}^j, \mathbf{y} \rangle$ is also unbounded. By LP strong duality, it is equivalent to the optimization problem LP^j being infeasible, where LP^j is defined as minimizing $\sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} c_r x_r$ subject to $\sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} a_{ir} x_r = f_i^j,\ \forall i \in \mathcal{N}$, $\sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} a_{kr} x_r \leqslant f_k^j,\ \forall k \in \mathcal{K}$, and $x_r \geqslant 0,\ \forall r \in \underline{\mathcal{R}}_2^{\leqslant}$. By the definition of $\mathbf{f}^j$, this means no feasible solution can be constructed using variables from $\underline{\mathcal{R}}_2^{\leqslant}$ when $x_j = 1$, which occurs infrequently in our experiments. A plausible explanation is that all enumerated variables, particularly those in $\underline{\mathcal{R}}_2^{\leqslant}$, are promising ones due to their relatively small reduced costs. Therefore, the chance that any $j \in \mathcal{J} \subseteq \underline{\mathcal{R}}$ cannot form a feasible solution along with variables in $\underline{\mathcal{R}}_2^{\leqslant}$ is slim.

However, when LP^j is indeed infeasible for some $j \in \mathcal{J}$, solving $\max_{\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}} \sum_{j \in \mathcal{J}} \langle \mathbf{f}^j, \mathbf{y} \rangle$ by an LP solver terminates once infeasibility is detected, yielding a possibly low-quality $\hat{\mathbf{y}} \in \mathcal{Y}^{\boldsymbol{\pi}}$ due to the

somewhat arbitrary termination. To address this issue, we only consider bounded $\mathcal{Y}^{\boldsymbol{\pi}}$. More precisely, for $i \in \mathcal{I}$, we lower and upper bound y_i by $-ub$ and ub, respectively, and let $\widehat{\mathcal{Y}^{\boldsymbol{\pi}}} := \mathcal{Y}^{\boldsymbol{\pi}} \cap \{\mathbf{y} \in \mathbb{R}^{|\mathcal{I}|} : -ub \leqslant y_i \leqslant ub,\ \forall i \in \mathcal{I}\}$. Note that for VRPs, lower bounding y_i for $i \in \mathcal{I}$ suffices to make $\mathcal{Y}^{\boldsymbol{\pi}}$ bounded since $a_{ir} \in \{0,1\}, \forall i \in \mathcal{N}, r \in \underline{\mathcal{R}}_2^{\leqslant}$. Our dual picking thus involves the following LPs.

$\widehat{\mathrm{DF}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$:

$$\begin{cases} \max \ \sum_{j \in \mathcal{J}} \langle \mathbf{f}^j, \mathbf{y} \rangle \\ \quad \text{s.t.} \ \sum_{i \in \mathcal{I}} a_{ir} y_i \leqslant c_r,\ \forall r \in \underline{\mathcal{R}}_2^{\leqslant}, \\ \qquad y_i \geqslant -ub,\ \forall i \in \mathcal{I}, \\ \qquad y_i \leqslant ub,\ \forall i \in \mathcal{N}, \\ \qquad y_i \leqslant 0,\ \forall i \in \mathcal{K}. \end{cases} \xrightarrow{\text{Dual}}$$

$\widehat{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$:

$$\begin{cases} \min \ \sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} c_r x_r + ub \cdot \left(\sum_{i \in \mathcal{I}} w_i + \sum_{i \in \mathcal{N}} v_i \right) \\ \quad \text{s.t.} \ \sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} a_{ir} x_r + v_i - w_i = \sum_{j \in \mathcal{J}} f_i^j, & \forall i \in \mathcal{N}, \\ \qquad \sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} a_{kr} x_r - w_i \leqslant \sum_{j \in \mathcal{J}} f_k^j, & \forall k \in \mathcal{K}, \\ \qquad x_r \geqslant 0, & \forall r \in \underline{\mathcal{R}}_2^{\leqslant}, \\ \qquad w_i \geqslant 0, \quad \forall i \in \mathcal{I}, \quad v_i \geqslant 0, \quad \forall i \in \mathcal{N}. \end{cases}$$

We choose to work with $\widehat{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$ instead of $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$ for implementation simplicity and computational efficiency. First of all, $\widehat{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$ can be modified from $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ more easily than $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$ inside a solver. Moreover, the dual simplex method has been empirically demonstrated to be superior to the primal simplex method (Bixby 2002). State-of-the-art LP solvers, such as Gurobi and CPLEX, almost always apply the dual simplex method in the default setting when running with a single thread. We iteratively vary the set $\mathcal{J}$ and solve $\widehat{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$ by the dual simplex to obtain an optimal dual solution $\hat{\mathbf{y}}$, which is subsequently used to compute LUs and identify the removable variables as per Propositions 1 and 2. Notably, it suffices to modify the right-hand side of $\widehat{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant}, \mathcal{J})$ when $\mathcal{J}$ is changed. Furthermore, the revised LP can be solved fast due to the warm-start effect of the dual simplex method in this case.

4.2.3. Further Discussion Changing $\mathcal{Y}^{\boldsymbol{\pi}}$ into $\widehat{\mathcal{Y}^{\boldsymbol{\pi}}}$ narrows the search region, which can potentially deteriorate the quality of $\hat{\mathbf{y}}$ obtained. However, according to our numerical experiments, such a presumed side effect is negligible. To provide some intuition for this observation, let us consider the formulation with $\mathcal{J} = \{j\}$, denoted by $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}_2^{\leqslant}, j)$, and its dual LP, denoted by $\widehat{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant}, j)$, for a given $j \in \underline{\mathcal{R}}$. Let $\hat{d}^{\{j\}}$ be the optimal value of $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}_2^{\leqslant}, j)$. Suppose $d^{\{j\}} := \max_{\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}} \langle \mathbf{f}^j, \mathbf{y} \rangle > ub$, then there exists $\mathbf{y} \in \mathcal{Y}^{\boldsymbol{\pi}}$ certifying that variable x_j can be removed. Let $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*)$ be an optimal solution to $\widehat{\mathrm{F}}(\underline{\mathcal{R}}_2^{\leqslant}, \{j\})$. When $\sum_{i \in \mathcal{I}} w_i^* + \sum_{i \in \mathcal{N}} v_i^* = 0$, we have that $\mathbf{x}^*$ is feasible to LP^j, and thus, according to strong duality, it holds that $\hat{d}^{\{j\}} \geqslant d^{\{j\}} > ub$. If $\sum_{i \in \mathcal{I}} w_i^* + \sum_{i \in \mathcal{N}} v_i^* \geqslant 1$, then again, we have $\hat{d}^{\{j\}} > ub$ when $c_r > 0$, suggesting that $\widehat{\mathcal{Y}^{\boldsymbol{\pi}}}$ still contains some elements which can certify that variable x_j is removable. In this sense, changing $\mathcal{Y}^{\boldsymbol{\pi}}$ to $\widehat{\mathcal{Y}^{\boldsymbol{\pi}}}$ causes a minimum difference. It is worth noticing that the above two cases (i.e., $\sum_{i \in \mathcal{I}} w_i^* + \sum_{i \in \mathcal{N}} v_i^* \leqslant 0$ or $\sum_{i \in \mathcal{I}} w_i^* + \sum_{i \in \mathcal{N}} v_i^* \geqslant 1$) are likely to happen because, for many problems including VRPs and CRPs, we have $f_i^j \in \mathbb{Z}$ for $i \in \mathcal{I}$.

4.3. Overview of DeLuxing

Algorithm 1 outlines the DeLuxing method, which consists of three steps explained in detail in Section 5. It is controlled by three input parameters: the number of clusters p and two threshold constants β_1 and β_2. In Step 1, an optimal dual solution to $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ is first obtained to compute reduced costs and initialize the index sets. In Step 2, the index set $\underline{\mathcal{R}}$ is first partitioned into p clusters via either the k-means++ clustering method (Arthur and Vassilvitskii 2007) or a simple but effective heuristic approach. In Step 3, a deep search for qualified dual solutions of good quality is performed using the centroid of each cluster as an initial reference point. This process involves calling the subroutine Algorithm 2 repeatedly with refined reference points, whose correctness is guaranteed by Propositions 1 and 2. Finally, Algorithm 1 outputs the index set of all variables certified as removable.

Algorithm 1: The Deep Lagrangian Underestimate Fixing (DeLuxing) Algorithm

Input: The number of clusters p, two threshold constants β_1 and β_2.

Step 1. *Initialization*: Solve $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ and obtain an optimal dual solution $\boldsymbol{\pi}$. Set $\underline{\mathcal{R}} \leftarrow \{r \in \underline{\mathcal{R}} : \bar{c}_r^{\boldsymbol{\pi}} \leqslant g\}$, $\mathcal{R}_1 \leftarrow \{r \in \underline{\mathcal{R}} : \bar{c}_r^{\boldsymbol{\pi}} \leqslant \frac{g}{2}\}$, $\mathcal{R}_2 \leftarrow \underline{\mathcal{R}} \backslash \mathcal{R}_1$, and $\mathcal{H} \leftarrow \varnothing$.

Step 2. *Clustering*: **If** $|\underline{\mathcal{R}}| \leqslant \beta_1$

Apply the k-means++ method to partition $\underline{\mathcal{R}}$ into p clusters, $\underline{\mathcal{R}}^1, \cdots, \underline{\mathcal{R}}^p$.

Else

Apply the ClustByNorm heuristic to partition $\underline{\mathcal{R}}$ into p clusters, $\underline{\mathcal{R}}^1, \cdots, \underline{\mathcal{R}}^p$.

Step 3. *Deep Search*:

For $i = 1$ to p

Set $\widetilde{\mathcal{J}} \leftarrow \underline{\mathcal{R}}^i \backslash \mathcal{H}$.

Do

Call the subroutine with input $\widetilde{\mathcal{J}}$, $\mathcal{R}_1$, $\mathcal{R}_2$, and obtain the output $\mathcal{D}$.

Set $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{D}$, $\widetilde{\mathcal{J}} \leftarrow \mathcal{D} \backslash \widetilde{\mathcal{J}}$, $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \backslash \mathcal{H}$, and $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \backslash \mathcal{H}$.

While $|\mathcal{D}| \geqslant \beta_2$

End

Output: The index set $\mathcal{H}$.

5. Elaboration on Every Step of DeLuxing

5.1. Step 1: Initialization

In this step, we first solve the linear program $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ to obtain an optimal dual solution $\boldsymbol{\pi}$. Then $\boldsymbol{\pi}$ is used to initialize two index sets $\mathcal{R}_1$ and $\mathcal{R}_2$, which keep track of the columns with reduced cost no larger than half of the current gap g and those within $(\frac{g}{2}, g]$ w.r.t. $\boldsymbol{\pi}$, respectively. It is worth noting that the dual solution used to enumerate $\underline{\mathcal{R}}$, referred to as $\tilde{\boldsymbol{\pi}}$, can also be used for

Algorithm 2: The Subroutine in DeLuxing

Input: Three index sets $\widetilde{\mathcal{J}}$, $\mathcal{R}_1$, and $\mathcal{R}_2$.

Substep 1. Solve $\widehat{\mathrm{F}}(\mathcal{R}_1, \widetilde{\mathcal{J}})$ and obtain an optimal dual solution $\hat{\mathbf{y}}$.

Substep 2. Compute $\mathcal{D} \leftarrow \{j \in \mathcal{R}_2 : \langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle > ub - c_j\} \cup \{j \in \mathcal{R}_1 : \langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle > ub - c_j - \min\{\eta_j, 0\}\}$, where $\eta_j = \min_{r \in \mathcal{S}^j} \{c_r - \langle \hat{\mathbf{y}}, \mathbf{a}_r \rangle\}$ and $\mathcal{S}^j = \{r \in \mathcal{R}_2 : x_r \text{ is compatible with } x_j\}$.

Output: The index set $\mathcal{D}$.

this purpose. However, we compute a new $\boldsymbol{\pi}$ because it updates the reduced costs and can help to remove some variables. We observe in our numerical experiments that, on average, about 10% of the enumerated variables can be certified to be removable using the updated reduced costs, i.e., $|\{r \in \underline{\mathcal{R}} : \bar{c}_r^{\boldsymbol{\pi}} > g\}| \approx 10\% \times |\underline{\mathcal{R}}|$. To improve computational efficiency, when the cardinality of $\underline{\mathcal{R}}$ is in the millions or higher, we skip solving $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ and directly set $\boldsymbol{\pi} = \tilde{\boldsymbol{\pi}}$ to initialize $\mathcal{R}_1$ and $\mathcal{R}_2$.

5.2. Step 2: Clustering

We maximize the inner product $\langle \sum_{j \in \mathcal{J}} \mathbf{f}^j, \mathbf{y} \rangle = |\mathcal{J}| \langle \bar{\mathbf{f}}, \mathbf{y} \rangle$ in the hope that the resulting $\mathbf{y}$ achieves close-to-optimal value for each individual $\langle \mathbf{f}^j, \mathbf{y} \rangle$, where $\bar{\mathbf{f}} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}}, \mathbf{f}^j$. Using the Cauchy-Schwarz inequality, we can derive $|\langle \mathbf{f}^j, \mathbf{y} \rangle - \langle \bar{\mathbf{f}}, \mathbf{y} \rangle| \leqslant \|\mathbf{f}^j - \bar{\mathbf{f}}\| \|\mathbf{y}\|$, which suggests we are likely to achieve our goal as long as $\|\mathbf{f}^j - \bar{\mathbf{f}}\|$ is small. This naturally leads us to the well-known k-means clustering problem that seeks to partition n observations $\{\mathbf{u}^1, \mathbf{u}^2, \cdots, \mathbf{u}^n\}$ in d dimension into k clusters $C_1, C_2, \cdots, C_k$ to minimize the within-cluster sum of squares, defined as $\sum_{i=1}^{k} \sum_{\mathbf{u} \in C_i} \|\mathbf{u} - \boldsymbol{\mu}^i\|^2$, where $\boldsymbol{\mu}^i$ is the mean (also called centroid) of points in the i-th cluster C_i.

While finding the optimal solution to the k-means clustering problem in d dimension is NP-hard even for two clusters (Aloise et al. 2009), many effective heuristics are available such as the Lloyd's algorithm (Lloyd 1982), refinement with Bradley and Fayyad's initialization (Bradley and Fayyad 1998), and the k-means++ (Arthur and Vassilvitskii 2007). Since k-means++ is known for its generally good performance (Celebi et al. 2013) and easy implementation, it has been used as the default method for determining initial cluster centroid positions in the "kmeans" function of Matlab. We also choose to use it in our C++ implementation.

We observe in our experiments that clustering vectors $\{\hat{\mathbf{f}}^j\}_{j \in \underline{\mathcal{R}}}$ instead of $\{\mathbf{f}^j\}_{j \in \underline{\mathcal{R}}}$ significantly improves the speed while yielding clusters that achieve nearly the same or sometimes even better overall performance for Algorithm 1. Here, $\hat{\mathbf{f}}^j := (\mathbf{f}_i^j)_{i \in \mathcal{N}}$ is a subvector of $\mathbf{f}^j$ that includes only the dimensions in $\mathcal{N}$. A possible explanation for this observation is that when two columns $j_1, j_2 \in \underline{\mathcal{R}}$ have similar coefficients a_{ij_1} and a_{ij_2} for $i \in \mathcal{N}$, it is likely that a_{ij_1} and a_{ij_2} are also close for $i \in \mathcal{K}$ since they correspond to coefficients of cutting planes. As a result, $\hat{\mathbf{f}}^j$ serves as a good representation of $\mathbf{f}^j$ for the purpose of clustering. In our implementation, we cluster $\{\hat{\mathbf{f}}^j\}_{j \in \underline{\mathcal{R}}}$. However, readers are encouraged to explore alternative options that may be more suitable for their specific problems.

5.2.1. The ClustByNorm Heuristic Although we can easily parallelize the computation using OpenMP (a library for parallel programming that supports C, C++, and Fortran) to achieve significant acceleration, the clustering process can still be time-consuming when the size of $\underline{\mathcal{R}}$ is in the millions. In such cases, we propose to use a simple but surprisingly effective heuristic, which we call ClustByNorm, to perform the clustering in place of the k-means++ method. It starts with computing the l_2 norm of each vector $\|\mathbf{f}^j\|$ for $j \in \underline{\mathcal{R}}$ and sorts them in non-increasing order. Then, we partition the sorted list into p clusters, each containing roughly $q = \lfloor |\underline{\mathcal{R}}|/p \rfloor$ vectors. Specifically, we assign the $(k-1)*q+1$ to $(k*q)$-th vectors in the sorted list to the k-th cluster for $k = 1, \cdots, p-1$, and the remaining vectors to the p-th cluster. Although ClustByNorm is slightly less effective than the k-means++ method in terms of the resulting DeLuxing's capability to remove columns, it leads to significant speedup when $|\underline{\mathcal{R}}|$ is large. We provide a detailed comparison of the performance of k-means++ and ClustByNorm in Section 7.1. By default, we use the k-means++ method for clustering and switch to ClustByNorm when $|\underline{\mathcal{R}}|$ exceeds a threshold constant β_1.

5.3. Step 3: Deep Search

For each cluster i, we first update it by $\widetilde{\mathcal{J}} \leftarrow \underline{\mathcal{R}}^i \backslash \mathcal{I}$ to exclude those identified as removable. Then its centroid $\boldsymbol{\mu} := \frac{1}{|\widetilde{\mathcal{J}}|} \sum_{j \in \widetilde{\mathcal{J}}} \mathbf{f}^j$ is used as a reference point to start the search. Specifically, we try to find a $\mathbf{y}$ that maximizes $\langle \boldsymbol{\mu}, \mathbf{y} \rangle$, which is accomplished by solving $\widehat{\mathrm{F}}(\mathcal{R}_1, \widetilde{\mathcal{J}})$ using the subroutine Algorithm 2. It returns the index set of removable variables $\mathcal{D}$ with the given input. However, bundling elements in a cluster according to this one-time clustering may not achieve the most desirable result. One reason is that the conditions in Propositions 1 and 2 aim to satisfy inequalities, whereas the clustering only concerns part of the inequalities, i.e., it tries to maximize the inner product on the left-hand side. Adding one extra dimension with a value of c_j to each vector $\mathbf{f}^j$ and clustering the updated vectors do not provide noticeable improvements, indicating the difficulty of incorporating information from the right-hand side. Moreover, the k-means++ method or the ClustByNorm is not perfect and is not likely to yield the best clusters.

The proposed deep search tries to address this concern. Essentially, it iteratively builds an artificial cluster by utilizing the most recently identified set $\mathcal{D}$ excluding $\widetilde{\mathcal{J}}$ (i.e., those that were used as input to Algorithm 2 to generate this $\mathcal{D}$). To the best of our knowledge, this idea is new in the literature. The rationale behind this approach is that the elements in a set $\mathcal{D}$ correspond to variables deemed removable by a common dual solution, which implicitly considers the whole inequalities and captures hidden similarities that might have been missed by the initial clustering. We remove $\widetilde{\mathcal{J}}$ from $\mathcal{D}$ because its information has already been used to generate $\mathcal{D}$ and is likely to be redundant and can adversely impact the next iteration. Multiple high-quality dual solutions are effectively picked in the do-while loop, and the total computational effort can be easily controlled by the input

threshold constant β_2. Specifically, the total number of calls to the subroutine is upper bounded by $(p + \lceil |\underline{\mathcal{R}}|/\beta_2 \rceil)$ because, by design, all the index sets $\mathcal{D}$ produced are non-overlapping.

6. Extensions

In this section, we expand upon the fundamental concept underlying DeLuxing to a broader range of contexts. Firstly, based on this concept, we prove that many integer variables can be relaxed into continuous ones in the IP solved to close a BBN, resulting in significant acceleration when the IP is large-scale and difficult to solve. Additionally, we demonstrate the concept can also be applied to enhance cutting plane addition. Furthermore, we propose an effective primal heuristic in which DeLuxing plays a crucial role. The effectiveness of these extensions is demonstrated individually using instances of the CMTVRPTW in Section 7.2. Moreover, we present the acceleration achieved by the new primal heuristic for solving the CVRP and VRPTW in Section 7.5. The first two extensions are not effective for the CVRP and VRPTW because a large number of relatively easy LPs and IPs are solved therein, while these extensions focus on dealing with a few more challenging LPs and IPs. Nonetheless, we believe the ideas behind them can potentially be applied to general MIP settings.

6.1. Variable Relaxation

Solving $\mathrm{F}(\underline{\mathcal{R}})$ as an IP by a solver is a convenient and effective way to close a BBN. The computational difficulty of $\mathrm{F}(\underline{\mathcal{R}})$ relies heavily on the number of integer variables, and it usually takes many rounds of branching and cutting plane additions before reducing the size of $\underline{\mathcal{R}}$ to a manageable level (e.g., Pessoa et al. 2020 requires $|\underline{\mathcal{R}}| \leqslant 10,000$). Relaxing the integer requirement for a large portion of variables is expected to bring substantial acceleration. If such relaxation is allowed, $\mathrm{F}(\underline{\mathcal{R}})$ can be solved as a MIP at a much earlier stage in the BPC method, saving a considerable amount of computational effort on branching and adding cutting planes to reduce the size of $\underline{\mathcal{R}}$. The following Proposition 3 indicates that in some cases, we can relax x_r for $r \in \underline{\mathcal{R}}_2^{>}$ in $\mathrm{F}(\underline{\mathcal{R}})$ as continuous variables without compromising optimality. Let $\breve{\mathrm{F}}(\underline{\mathcal{R}})$ be the formulation obtained by relaxing variables x_r for $r \in \underline{\mathcal{R}}_2^{>}$ in $\mathrm{F}(\underline{\mathcal{R}})$ to be continuous and adding the constraint $\sum_{r \in \underline{\mathcal{R}}_2^{>}} x_r \leqslant 1$. Let $\mathcal{P} \subset \mathbb{R}_+^{|\underline{\mathcal{R}}|}$ be the polyhedron corresponding to the feasible region of $\breve{\mathrm{F}}(\underline{\mathcal{R}})$, $\mathcal{E} \subset \mathcal{P}$ be the set of extreme points of $\mathcal{P}$, and $\mathcal{X}^* \subset \mathcal{P}$ be the set of optimal solutions to $\breve{\mathrm{F}}(\underline{\mathcal{R}})$.

Proposition 3. *If in* $\mathrm{F}(\underline{\mathcal{R}})$, $a_{ir} \in \{0,1\}$ *and* $b_i \in \mathbb{Z}$ $\forall i \in \mathcal{N}, r \in \underline{\mathcal{R}}$, *then it follows that* $\mathcal{X}^* \cap \mathcal{E} \subset \mathbb{Z}_+^{|\underline{\mathcal{R}}|}$.

Proof Let us consider any $\bar{\mathbf{x}} \in \mathcal{X}^* \cap \mathcal{E}$. It holds that $\bar{x}_r \in \mathbb{Z}_+$ for $r \in \underline{\mathcal{R}}_2^{\leqslant}$ due to the integer requirement in $\breve{\mathrm{F}}(\underline{\mathcal{R}})$. With the coefficients $a_{ir} \in \{0,1\}$ and $b_i \in \mathbb{Z}$ for all $i \in \mathcal{N}$ and $r \in \underline{\mathcal{R}}$, it follows that $\sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} a_{ir}\bar{x}_r \in \mathbb{Z}$ $\forall i \in \mathcal{N}$. Let $u_i := \sum_{r \in \underline{\mathcal{R}}_2^{>}} a_{ir}\bar{x}_r$. Consequently, $u_i = b_i - \sum_{r \in \underline{\mathcal{R}}_2^{\leqslant}} a_{ir}\bar{x}_r$ is integral for all $i \in \mathcal{N}$. For $r \in \underline{\mathcal{R}}_2^{>}$, $\bar{x}_r$ is non-negative, thus $u_i \in \mathbb{Z}_+$. Additionally, the constraint $\sum_{r \in \underline{\mathcal{R}}_2^{>}} x_r \leqslant 1$

in $\breve{\mathrm{F}}(\underline{\mathcal{R}})$ implies $\sum_{r\in\underline{\mathcal{R}}_2^>}\bar{x}_r \leqslant 1$. Let $\mathcal{Q} := \{r \in \underline{\mathcal{R}}_2^> :\ 0 < \bar{x}_r < 1\}$. If $\mathcal{Q}$ is an empty set, no further proof is needed. Let $\mathcal{N}^r := \{i \in \mathcal{N} :\ a_{ir} = 1\}$ for $r \in \mathcal{Q}$ and $\widetilde{\mathcal{N}} := \cup_{r\in\mathcal{Q}}\mathcal{N}^r$.

We claim that if $\mathcal{Q} \neq \varnothing$, then $\mathcal{N}^{r_1} = \mathcal{N}^{r_2}\ \forall r_1, r_2 \in \mathcal{Q}$. The claim is proved by contradiction. First, $\mathcal{Q} \neq \varnothing$ and $\sum_{r\in\underline{\mathcal{R}}_2^>}\bar{x}_r \leqslant 1$ imply there does not exist $r \in \underline{\mathcal{R}}_2^>$ such that $\bar{x}_r \geqslant 1$. Thus, we have $x_r = 0\ \forall r \in \underline{\mathcal{R}}_2^> \backslash \mathcal{Q}$. Suppose there exist $r_1, r_2 \in \mathcal{Q}$ such that $\mathcal{N}^{r_1} \neq \mathcal{N}^{r_2}$. As a result, there exist $k \in \widetilde{\mathcal{N}}$, $r', r'' \in \mathcal{Q}$ such that $k \in \mathcal{N}^{r'}$ and $k \notin \mathcal{N}^{r''}$. Therefore, we have $0 < x_{r'} \leqslant \sum_{r\in\underline{\mathcal{R}}_2^>} a_{kr}\bar{x}_r = \sum_{r\in\mathcal{Q}} a_{kr}\bar{x}_r < \sum_{r\in\mathcal{Q}}\bar{x}_r = \sum_{r\in\underline{\mathcal{R}}_2^>}\bar{x}_r \leqslant 1$. This implies $u_k \in (0,1)$, which contradicts the fact that u_k is an integer and thus proves the claim. Next we will show that if $\mathcal{Q} \neq \varnothing$ then $\bar{\mathbf{x}} \notin \mathcal{E}$.

Note that $\mathcal{Q}$ cannot be a singleton because if $\mathcal{Q} = \{r\}$, then $0 < \sum_{r\in\underline{\mathcal{R}}_2^>} a_{ir}\bar{x}_r = \bar{x}_r < 1$ for $i \in \mathcal{N}^r$, which again contradicts the fact that u_i is integral. Consider any two distinct $r_1, r_2 \in \mathcal{Q}$. According to the claim, we have $\mathcal{N}^{r_1} = \mathcal{N}^{r_2}$, i.e., $a_{ir_1} = a_{ir_2}$ for all $i \in \mathcal{N}$. Let $\tilde{\mathbf{x}}' = (\tilde{x}'_r)_{r\in\underline{\mathcal{R}}}$ and $\tilde{\mathbf{x}}'' = (\tilde{x}''_r)_{r\in\underline{\mathcal{R}}}$ be set to $\tilde{x}'_r = \tilde{x}''_r = \bar{x}_r$ for $r \in \underline{\mathcal{R}}\backslash\{r_1, r_2\}$ and $\tilde{x}'_{r_1} = \bar{x}_{r_1} - \epsilon$, $\tilde{x}'_{r_2} = \bar{x}_{r_2} + \epsilon$, $\tilde{x}''_{r_1} = \bar{x}_{r_1} + \epsilon$, and $\tilde{x}''_{r_2} = \bar{x}_{r_2} - \epsilon$, where $\epsilon > 0$ is small enough to ensure $\tilde{x}'_{r_1}$ and $\tilde{x}''_{r_2}$ are non-negative. Then $\sum_{r\in\underline{\mathcal{R}}} a_{ir}\bar{x}_r = \sum_{r\in\underline{\mathcal{R}}} a_{ir}\tilde{x}'_r = \sum_{r\in\underline{\mathcal{R}}} a_{ir}\tilde{x}''_r$ and thus $\tilde{\mathbf{x}}'$ and $\tilde{\mathbf{x}}''$ are feasible solutions to $\breve{\mathrm{F}}(\underline{\mathcal{R}})$. Furthermore, it follows that $\bar{\mathbf{x}} = (\tilde{\mathbf{x}}' + \tilde{\mathbf{x}}'')/2$, suggesting that $\bar{\mathbf{x}}$ is not an extreme point of $\mathcal{P}$, i.e., $\bar{\mathbf{x}} \notin \mathcal{E}$.

Therefore, the set $\mathcal{Q}$ has to be empty when $\bar{\mathbf{x}} \in \mathcal{E}$. The fact that $\sum_{r\in\underline{\mathcal{R}}_2^>} x_r \leqslant 1$ guarantees that $0 \leqslant x_r \leqslant 1$. Consequently, all the elements of $\bar{\mathbf{x}}$ take an integer value, which completes the proof. ■

Remarks: Proposition 3 guarantees that when the constraint coefficients a_{ir} are binaries and b_i are integers, any optimal solution to $\breve{\mathrm{F}}(\underline{\mathcal{R}})$ is also integral as long as it is an extreme point of the underlying polyhedron. For set partitioning formulations of VRPs and CRPs, the coefficients $a_{ir} \in \{0,1\}$ and $b_i = 1$ for all $i \in \mathcal{N}$ and $r \in \underline{\mathcal{R}}$, and thus the conditions are satisfied. However, it should be noted that when there exist two distinct $r_1, r_2 \in \underline{\mathcal{R}}$ such that $c_{r_1} = c_{r_2}$ and $a_{ir_1} = a_{ir_2}\ \forall i \in \mathcal{N}$, it is possible for $\breve{\mathrm{F}}(\underline{\mathcal{R}})$ to have an optimal solution that is not integral. This means there could be two identical columns that cannot be deleted due to the absence of certain feasibility requirements in the formulation, which is added as lazy cuts during the solution process. For the standard CVRP and VRPTW, this does not occur because there are no missing feasibility requirements in their formulations. In the case of CMTVRPTW, where the superstructure feasibility constraints are initially absent and are added dynamically using a callback function, such a scenario can happen. These constraints decide if there exists a feasible schedule to execute the selected vehicle routes by a given fleet, allowing each vehicle to make multiple trips (refer to Section 3.2 of Yang (2023) for details). Nonetheless, as long as the solver returns an optimal solution that represents an extreme point of $\mathcal{P}$, the proposed relaxation in Proposition 3 can still be applied without sacrificing optimality. Note that modern MIP solvers may yield an optimal solution that is not necessarily an extreme point via some primal heuristics. In this case, a callback function that cuts off such solutions by lazy constraints (so-called no-good cuts, Hooker et al. 1999) can be used to guarantee correctness.

The proposed relaxation can be performed even more aggressively with the help of a simple search and some lazy constraints to ensure optimality and integrality. Let $\widetilde{\mathrm{F}}(\underline{\mathcal{R}})$ be the formulation obtained by relaxing variables x_r for $r \in \underline{\mathcal{R}}_3^>$ in the original formulation $\mathrm{F}(\underline{\mathcal{R}})$ to be continuous and adding the constraints $\sum_{r\in\underline{\mathcal{R}}_2^>} x_r \leqslant 1$ and $\sum_{r\in\underline{\mathcal{R}}_3^>} x_r \leqslant 2$. We solve $\widetilde{\mathrm{F}}(\underline{\mathcal{R}})$ by a MIP solver with the callback function presented in Algorithm 3 that adds lazy constraints.

To ease the presentation, given a feasible solution $\bar{\mathbf{x}}$ to $\widetilde{\mathrm{F}}(\underline{\mathcal{R}})$, we define $\mathcal{R}^f := \{r \in \underline{\mathcal{R}}_3^> : \ \bar{x}_r > 0\}$, $\mathcal{R}^t := \{r \in \underline{\mathcal{R}}\backslash\underline{\mathcal{R}}_3^> : \ \bar{x}_r > 0\}$, $\tilde{c} := \sum_{r\in\mathcal{R}^t} c_r\bar{x}_r$, and $\tilde{\mathbf{b}} := (\tilde{b}_i)_{i\in\mathcal{N}}$, where $\tilde{b}_i = b_i - \sum_{r\in\mathcal{R}^t} a_{ir}\bar{x}_r$. The callback function is triggered whenever such a feasible solution $\bar{\mathbf{x}}$ is identified. It first examines whether all $\bar{x}_r$ values for $r \in \underline{\mathcal{R}}_3^>$ are integers. If they are, no further action is required, and the callback function terminates, returning control to the solver. If any $\bar{x}_r$ for $r \in \underline{\mathcal{R}}_3^>$ is not an integer, the callback function proceeds by searching for a feasible solution better than the one achieving the current upper bound (known as the incumbent). Specifically, it checks whether any two columns $r' \neq r''$ and $r', r'' \in \mathcal{R}^f$, together with columns with indices in $\mathcal{R}^t$ that have been selected an integral number of times in $\bar{\mathbf{x}}$, can form a superior feasible solution. This search can be conducted in a brute-force fashion, as we only need to consider $|\mathcal{R}^f|^2$ combinations. Notably, the size of $\mathcal{R}^f$ is small, typically less than a few dozen. After the search, a lazy constraint $\sum_{r\in\mathcal{R}^f\cup\mathcal{R}^t} x_r \leqslant |\mathcal{R}^t| + 1$ is added and the callback terminates.

Algorithm 3: The Callback Function

Input: A feasible solution $\bar{\mathbf{x}}$ to $\widetilde{\mathrm{F}}(\underline{\mathcal{R}})$ and current upper bound $\tilde{z}$

if $\exists r \in \underline{\mathcal{R}}_3^>$ *such that* $\bar{x}_r \in \mathcal{R}_+\backslash\mathbb{Z}_+$ **then**

Step 1. Search for $r', r'' \in \mathcal{R}^f$, $r' \neq r''$ with the smallest $c_{r'} + c_{r''}$ such that $\mathbf{a}_{r'} + \mathbf{a}_{r''} = \tilde{\mathbf{b}}$ and $c_{r'} + c_{r''} < \tilde{z} - \tilde{c}$. If one is found, update the incumbent solution to $\tilde{\mathbf{x}}$ by Equation (3).

Step 2. Add a lazy constraint $\sum_{r\in\mathcal{R}^f\cup\mathcal{R}^t} x_r \leqslant |\mathcal{R}^t| + 1$.

$$\tilde{\mathbf{x}} := (\tilde{x}_r)_{r\in\underline{\mathcal{R}}}, \text{ where } \tilde{x}_r = \begin{cases} 1, & \text{if } r \in \mathcal{R}^t \cup \{r', r''\}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The following Proposition 4 guarantees that under some mild conditions, we can obtain an optimal solution to $\mathrm{F}(\underline{\mathcal{R}})$ by solving $\widetilde{\mathrm{F}}(\underline{\mathcal{R}})$ with the proposed callback function in Algorithm 3.

Proposition 4. *Suppose in* $\mathrm{F}(\underline{\mathcal{R}})$*, all variables* x_r *for* $r \in \underline{\mathcal{R}}$ *are required to be binary,* $a_{ir} \in \{0, 1\}$*, and* $b_i \in \mathbb{Z}$ $\forall i \in \mathcal{N}, r \in \underline{\mathcal{R}}$*. Solving* $\widetilde{\mathrm{F}}(\underline{\mathcal{R}})$ *by a MIP solver that is equipped with the callback function described in Algorithm 3 and can find an optimal solution corresponding to an extreme point of the polyhedron (i.e., the feasible region of* $\widetilde{\mathrm{F}}(\underline{\mathcal{R}})$*) guarantees to find an optimal solution to* $\mathrm{F}(\underline{\mathcal{R}})$*.*

Proof If all $\bar{x}_r$ values for $r \in \underline{\mathcal{R}}_3^>$ are integers, then $\bar{\mathbf{x}}$ is feasible to $\mathrm{F}(\underline{\mathcal{R}})$. We only need to consider the case that there exists $\bar{x}_r$ taking a fractional value for some $r \in \underline{\mathcal{R}}_3^>$. Since $\sum_{r\in\mathcal{R}^f} x_r \leqslant \sum_{r\in\underline{\mathcal{R}}_3^>} x_r \leqslant 2$ and $x_r \in \{0,1\}$ for $r \in \mathcal{R}^t \subseteq \underline{\mathcal{R}}$, we have $\sum_{r\in\mathcal{R}^f\cup\mathcal{R}^t} x_r = \sum_{r\in\mathcal{R}^t} x_r + \sum_{r\in\mathcal{R}^f} x_r \leqslant |\mathcal{R}^t| + 2$. Since $\sum_{r\in\mathcal{R}^f\cup\mathcal{R}^t} x_r \in \mathbb{Z}_+$, the lazy constraint $\sum_{r\in\mathcal{R}^f\cup\mathcal{R}^t} x_r \leqslant |\mathcal{R}^t|+1$ will eliminate part of the feasible region with $\sum_{r\in\mathcal{R}^f\cup\mathcal{R}^t} x_r = |\mathcal{R}^t| + 2$. We claim that if an optimal solution exists in this eliminated part, it can be found, and thus, optimality can still be guaranteed.

Note that $\sum_{r\in\mathcal{R}^f\cup\mathcal{R}^t} x_r = |\mathcal{R}^t| + 2$ implies $x_r = 1$ for all $r \in \mathcal{R}^t$ and $\sum_{r\in\mathcal{R}^f} x_r = 2$. Thus, we only need to search for all $r', r'' \in \mathcal{R}^f$ and $r' \neq r''$, which, when combined with columns indexed by $r \in \mathcal{R}^t$, can form a superior feasible solution to the current incumbent solution. As shown in Algorithm 3, the callback function searches all such qualified pairs of columns and picks the best one, which completes the claim. It remains to show that integrality is also guaranteed.

It suffices to show the lazy constraints added by the callback can prevent any fractional solution from being considered feasible. We only need to consider fractional solutions $\bar{\mathbf{x}}$ with $\sum_{r\in\mathcal{R}^f} \bar{x}_r \leqslant 1$. By a similar argument to that provided in the above proof of Proposition 3, we can show that such $\bar{\mathbf{x}}$ cannot be an extreme point of the corresponding polyhedron, which completes the proof. ■

6.2. A New Way of Cutting Plane Addition

Cutting planes play a key role in modern branch-and-cut and BPC methods, which iteratively improve the dual bound and thus close the optimality gap. After obtaining an optimal solution to the current LP relaxation, cut separators are employed to identify violated valid inequalities. These inequalities can cut off the current fractional LP solution and are then incorporated as constraints in the LP. This process continues until some termination criteria are met.

For BPC methods, solving each LP after each round of cut addition requires repeated CG. When an enumerated pool is available, CG can be performed through inspection, which is much more efficient compared to a labeling algorithm. Nevertheless, this process can still be time-consuming when a considerable number of rounds of cut addition and CG are performed to solve the LP. Inspired by DeLuxing, we propose a new approach to streamline this procedure. Specifically, instead of starting with the columns currently present in the LP, we first include a moderate-sized (typically no larger than 50,000) subset of columns, denoted by $\widetilde{\mathcal{R}}$, from the enumerated pool $\underline{\mathcal{R}}$. We then obtain formulation $\overline{\mathrm{F}}(\widetilde{\mathcal{R}})$, with $\underline{\mathcal{R}}$ in formulation $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ from Section 3.2 replaced by $\widetilde{\mathcal{R}}$. One particular selection criterion that works well numerically is to include all columns from the enumerated pool with reduced costs not surpassing half of the gap, i.e., $\widetilde{\mathcal{R}} = \underline{\mathcal{R}}_2^{\leqslant}$. Subsequently, we generate cuts, such as subset-row inequalities (Jepsen et al. 2008) and rounded capacity cuts (Laporte and Nobert 1983), to tighten $\overline{\mathrm{F}}(\widetilde{\mathcal{R}})$ and directly resolve it without introducing new columns from the remaining pool. This process is repeated until tailing off or hitting prespecified limits. Before termination, all

remaining columns in the pool are introduced into the tightened $\overline{\mathrm{F}}(\widetilde{\mathcal{R}})$ to compute the final lower bound, ensuring its validity.

This approach presents two notable advantages. Firstly, due to the significantly smaller size of $\widetilde{\mathcal{R}}$ compared to $\underline{\mathcal{R}}$, we can circumvent the need to solve large LPs when CG adds an excessive number of columns. Secondly, when $\widetilde{\mathcal{R}}$ is selected such that $\underline{\mathcal{R}}_2^{\leqslant} \subseteq \widetilde{\mathcal{R}}$, additional opportunities for variable fixing emerge. More precisely, we can directly apply Propositions 1 and 2 to remove columns whenever the LP is resolved and a new dual solution is obtained. We acknowledge that inexact objective values are computed in the process since only a subset of columns are included in the current LP, which affects the cuts generated and potentially the quality of the lower bound obtained at the termination of the procedure. However, we emphasize the validity of the final lower bound, as all remaining columns in the pool will be added back to compute the bound. Our numerical experiments demonstrate that adding cuts in this manner yields almost identical bounds in our numerical experiments and results in significant speedup for some instances, as shown in Figure 5.

6.3. An Effective Primal Heuristic

Using a solver to solve the current RMP as an IP serves as a commonly employed primal heuristic within the BPC framework (Pessoa et al. 2020). The success of this approach is closely tied to the number of columns present in the RMP. An excessive number of columns can result in long computation times, whereas too few columns may produce feasible solutions of poor quality or result in infeasibility. To tackle this challenge, a straightforward approach is to only keep in the IP the smallest $\hat{\beta}$ columns in terms of reduced costs, where $\hat{\beta}$ is a constant. This approach leaves room for improvement, especially for the CMTVRPTW and its variants.

Similar to Pessoa et al. (2009), Marques et al. (2020) and Liguori et al. (2023), we first perform a trial enumeration with a small tentative gap. For the CMTVRPTW and its two variants tested in this paper, our experiments suggest that setting the trial gap to be $\min\{1\% \times lb, 10\}$ works well, where lb is the current lower bound. For the CVRP and VRPTW tested in Section 7.5, we used $\min\{0.2\% \times lb, 40\}$ for standard instances and $\min\{0.2\% \times lb, 20\}$ for long instances. In practice, this value can be straightforwardly determined through several trials aimed at creating a pool of moderate size (containing 50,000 to 500,000 columns) in most cases. Subsequently, DeLuxing is applied to remove unnecessary columns from this enumerated trial pool. Finally, we solve an IP using the columns remaining in the pool. In case the pool still contains more than $\hat{\beta}$ columns, we can keep the smallest $\hat{\beta}$ ones based on their reduced costs.

This simple heuristic has proven highly effective. One of the main factors contributing to its success is that some columns essential for constructing high-quality feasible solutions might be absent in the current RMP but can be generated through the trial enumeration. DeLuxing plays a crucial role in

this heuristic, as the trial enumeration can still produce a large number of columns. Nonetheless, a direct screening based solely on reduced costs, as described in the previous paragraph, performs badly. DeLuxing can often reduce the size of the column pool to be much smaller than $\hat{\beta}$ while ensuring that necessary columns are retained in the pool.

7. Numerical Results

In this section, we present an extensive numerical study comprising five sets of experiments. The first set aims to show the effectiveness of the key components of DeLuxing in removing columns. In the second set, we individually evaluate the effectiveness of DeLuxing and each extension introduced in Section 6 using CMTVRPTW instances. The third set compares our default method (with DeLuxing and the three extensions enabled) with the state-of-the-art algorithms on the CMTVRPTW and its two important variants, CMTVRPTW-LT and CMTVRPTW-R. We exclude the other two variants, the CMTVRPTW *with limited trip duration* (CMTVRPTW-LD) and the *drone routing problem* (DRP) considered in Yang (2023) as the difficulty of solving them does not stem from generating excessive variables in the solution process. In fact, the number of routes generated in solving the CMTVRPTW-LD and DRP instances is relatively small (mostly several thousand for the CMTVRPTW-LD and tens of thousands for the DRP) according to Tables EC.8 and EC.10 in Yang (2023). The fourth set of experiments seeks to further demonstrate the potential of the proposed approach by solving significantly larger multi-trip instances, with sizes twice as large as the largest ones currently documented in the literature. The last set of experiments is dedicated to showing that the proposed DeLuxing and the primal heuristic also substantially accelerate one of the world's leading solvers, RouteOpt (You et al. 2023), for solving the CVRP and VRPTW instances. Note that RouteOpt incorporates the bucket arc elimination (Sadykov et al. 2021), a sophisticated type of arc-flow fixing. Throughout the solution process, RouteOpt also applies standard RCF.

For the CMTVRPTW, we consider two datasets, totaling 171 instances. The first set comprises 81 instances described in Section 7.4.1. of Yang (2023), which are derived from the 27 type 2 Solomon instances. For each instance, we consider three cases: the first 70, 80, and all 100 customers. The second set consists of 90 large instances derived from the 30 instances (C2, R2, and RC2) in the G02 group (see Homberger and Gehring 2005). For each instance, we use the first 140, 170, and all 200 customers. The numbers of vehicles are set to 6, 7, 8, 12, 16, and 20 for instances with 70, 80, 100, 140, 170, and 200 customers, respectively, and the vehicle capacity is set to 100 for all the instances. For the CMTVRPTW-LT, we use the same 171 instances as the CMTVRPTW with the same parameters. The loading time of each customer is set to 20% of its service time following the procedure in Hernandez et al. (2016). For the CMTVRPTW-R, we use a total of 513 instances: 243 instances from Section 7.4.4. of Yang (2023) generated from the 81 CMTVRPTW instances via the procedure in

Cattaruzza et al. (2016a) with $\kappa \in \{0.25, 0.5, 0.75\}$, and an additional 270 instances generated from the 90 large CMTVRPTW instances using the same procedure. The number of vehicles and vehicle capacity are set to the same as those of the corresponding CMTVRPTW instances.

For classic VRPs, we employ the 200-node CVRP and 300-node VRPTW instances generated according to the procedure outlined in Uchoa et al. (2017). Specifically, the depot and customers are assigned integer coordinates within a $[0, 1000] \times [0, 1000]$ grid. The depot is placed at the center of the grid with coordinates (500, 500), and customer locations are randomly distributed across the grid. Customer demands are integer values uniformly drawn from [1, 100]. For the CVRP instances, the average route length, r, is uniformly selected from [6, 10], and the vehicle capacity is calculated by $Q = \lceil \frac{rD}{n} \rceil$, where D is the total demand of all customers and n is the number of customers. For the VRPTW instances, the vehicle capacity is set to 1000, the service time is 1000, and the maximum time resource H is 12000. For 80% of the customers, their time windows are of the form $[0,\ b_i]$, where b_i is an integer uniformly selected from $[0,\ \lfloor 0.8H \rfloor]$. The remaining 20% of the customers have time windows $[a_i,\ b_i]$, with a_i and b_i, respectively, uniformly selected from $[\lfloor 0.2H \rfloor,\ \lfloor 0.8H \rfloor]$ and $[a_i,\ \min\{H, \lfloor a_i + 0.25H \rfloor\}]$. These 200 CVRP and 200 VRPTW instances feature branch-and-bound trees with hundreds of nodes, and thus are generally challenging to solve. Meanwhile, the route length is moderate, making the pricing not overly difficult when many non-robust cuts, e.g., limited memory rank-1 cuts (Pecin et al. 2017b), are present. Instances from the well-known CVRPLIB (Lima et al. 2014) are not ideal for demonstrating the advantages of our proposed methods because most of them are either too easy (resulting in trees with several nodes or even being solved at the root) or too challenging (unsolvable within a day) for cutting-edge solvers.

All experiments are conducted on a workstation running Ubuntu 20.04 equipped with an Intel(R) Core(TM) i9-12900K CPU @ 3.90GHz and 128GB of RAM. The code is implemented in C++ language and compiled by g++ 9.4.0. Gurobi 9.1.1 is used as an LP and IP solver. All LPs are solved in the single-thread mode, and 8 threads are used to solve all IPs (MIPs). As a benchmark for our last set of experiments, the VRP-Solver is run on the same machine in the same mode (one thread for LPs and 8 threads for IPs/MIPs) using CPLEX 22.1.0. The k-means++ method is run parallelly with all available threads. The time limit for each multi-trip instance is set to 3 hours, and no limit is set for the CVRP and VRPTW instances. The source code and data needed to reproduce the results in this paper can be found in the electronic companion.

7.1. Effectiveness of the Key Components of DeLuxing

In this section, we aim to demonstrate the effectiveness of the key components of DeLuxing. The **Benchmark** is Algorithm 1 with $p = 50$, $\beta_1 = 500,000$, and $\beta_2 = 50$. We consider the following four variants, where the parameters p, β_1, and β_2 are set to the same values as Benchmark unless otherwise specified.

1. **FullForm**: This variant modifies Substep 1 of Algorithm 2 to solve the full formulation $\widehat{\mathrm{F}}(\underline{\mathcal{R}}, \widetilde{\mathcal{J}})$ instead of our formulation $\widehat{\mathrm{F}}(\mathcal{R}_1, \widetilde{\mathcal{J}})$ enabled by the proposed Propositions 1 and 2.
2. **ClustByNorm**: This variant sets β_1 to 0 and thus always applies the proposed ClustByNorm heuristic for the initial clustering.
3. **Random**: This variant randomly partitions $\underline{\mathcal{R}}$ into p clusters of equal size.
4. **NoDeepSearch**: This variant sets β_2 to $+\infty$ to skip the proposed deep search.

We compare the performance of these five methods on the column pools enumerated in the process of solving the CMTVRPTW instances. The advantages of the new formulation enabled by Propositions 1 and 2 can be demonstrated through a comparison of Benchmark and FullForm. The effectiveness of the straightforward heuristic approach, ClustByNorm, will be shown by comparing it with Benchmark and Random. Lastly, the benefits of the proposed deep search can be observed by comparing Benchmark with NoDeepSearch. The following information is included: the number of customers n, the average percentage of columns removed, and the average computing time (in seconds; CPU). Each average value is taken over all instances of the same size.

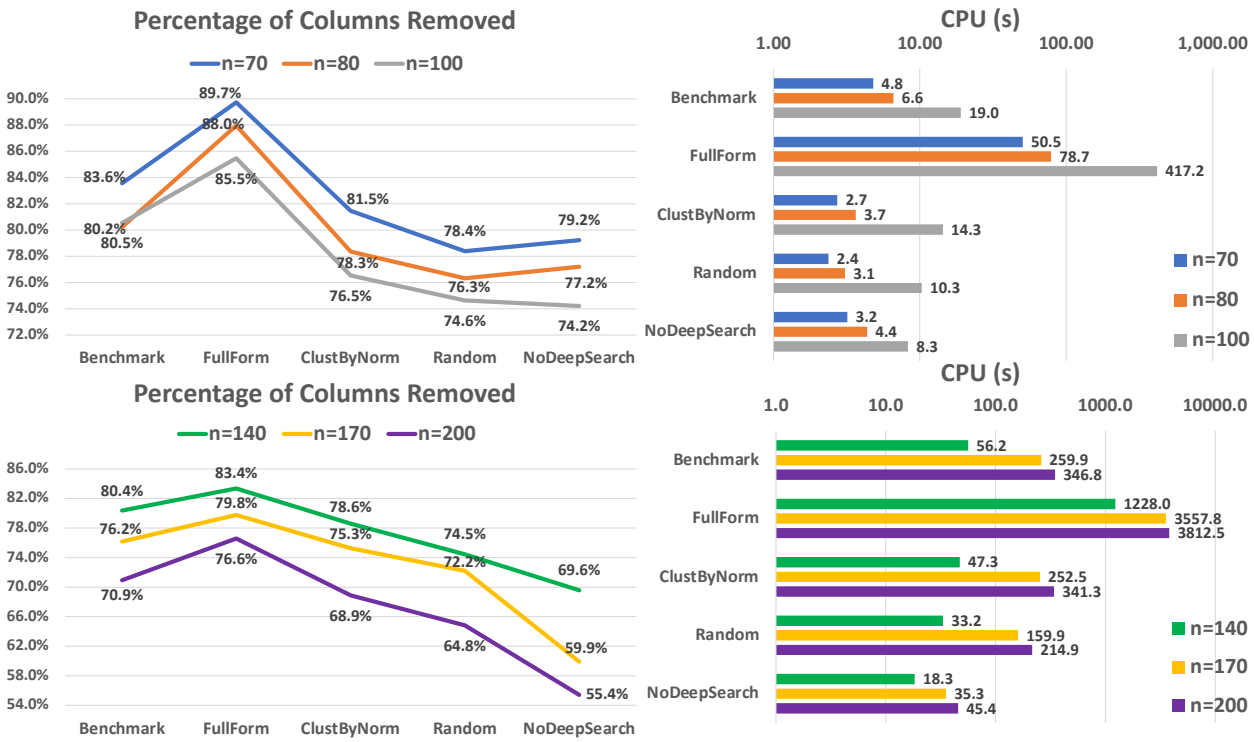

Figure 2 Comparison of the percentage of columns removed and the CPU in seconds (the x-axis is in logarithmic scale) for the CMTVRPTW. Each number is the average value taken over instances of the same size.

Figure 2 illustrates the performance comparison among different variants. Benchmark outperforms all other variants except FullForm, in terms of the percentage of columns removed. This superiority

becomes more pronounced when dealing with larger instances, where the number of customers is higher and the challenges are greater. It is important to note that even a 1% increase in the removal percentage translates to thousands of additional variables being eliminated, considering the average pool size of over 100,000 columns. Such extra reductions in variables drastically impact the overall algorithmic performance. While Benchmark may not remove as many columns as FullForm, it manages to reduce the computational time to less than one-tenth that of FullForm for instances with 140 or fewer customers. Such CPU reductions are crucial for the success of DeLuxing and highlight the significance of the new formulation. It is worth mentioning that FullForm hits the 3-hour time limit for most 170- and 200-customer R and RC instances, which is the reason why the CPU differences between Benchmark and FullForm are less significant.

Comparing ClustByNorm with Random and Benchmark, we conclude that ClustByNorm is effective, exhibiting much better performance than random initial clustering, albeit slightly inferior to Benchmark. Furthermore, the computational overhead associated with ClustByNorm is smaller than Benchmark. The importance of the proposed deep search is evident when comparing Benchmark with NoDeepSearch. Notably, for large instances, i.e., those with 140 or more customers, the proposed deep search substantially increases the percentage of columns removed.

7.1.1. Sensitivity Analysis In this section, we aim to justify our selection of values for the three parameters p, β_1, and β_2 and illustrate their sensitivity. We conduct tests with varying values: $p \in \{10, 20, 50, 100\}$, $\beta_1 \in \{100{,}000;\ 200{,}000;\ 500{,}000;\ 1{,}000{,}000\}$, and $\beta_2 \in \{10, 50, 100\}$, resulting in a total of 48 combinations. For each combination, we run an experiment on the same 30 column pools from the 30 instances with 170 customers. We present the average percentage of columns removed and the average CPU in seconds.

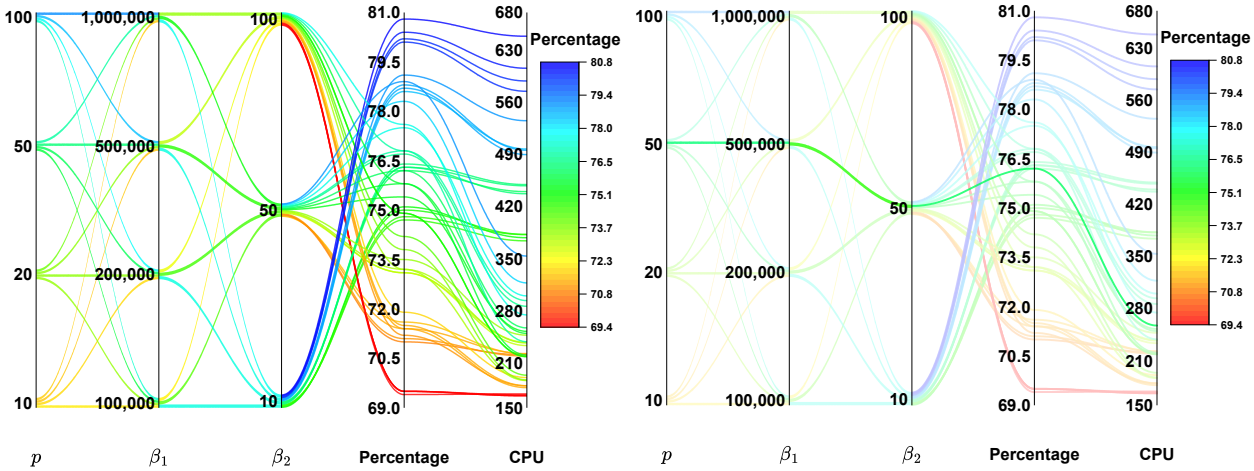

Figure 3 Comparison of the average percentage of columns removed and the average CPU in seconds for the 170-customer instances. The right subfigure highlights the choice of $p = 50$, $\beta_1 = 500{,}000$, and $\beta_2 = 50$.

As shown in Figure 3, our chosen values of $p = 50$, $\beta_1 = 500,000$, and $\beta_2 = 50$ achieve a 76.2% column reduction in 259.9 seconds, striking a suitable balance between the percentage of columns removed and the computational time. Moreover, the parameters are not overly sensitive to variations. In particular, multiple combinations, such as (1) $p = 50$, $\beta_1 = 100,000$, $\beta_2 = 50$; (2) $p = 50$, $\beta_1 = 200,000$, $\beta_2 = 50$; (3) $p = 100$, $\beta_1 = 200,000$, $\beta_2 = 100$; and (4) $p = 100$, $\beta_1 = 500,000$, $\beta_2 = 100$, achieve a comparable reduction of around 76% within the range of 220 to 260 seconds. Detailed results are summarized in Table EC.2 in Section EC.1 of the e-companion.

7.2. Effectiveness of DeLuxing and Three Extensions

We demonstrate the isolated effectiveness of DeLuxing and the three inspired extensions described in Section 6. Our baseline method, denoted by Default, is a modified implementation of the EPCEM from Yang (2023) with DeLuxing and the three extensions incorporated as follows (see Figure 4 for an illustration). The extension in Section 6.3 (our new primal heuristic) is applied in *Step 2* in place of the standard primal heuristic to compute a valid UB. In addition, we replace *Steps 4 and 5* of the EPCEM with *Step 4'* that performs multiple rounds of alternating cut addition and column removal by DeLuxing. In each round, we add cuts using the extension in Section 6.2 (the new way of cut addition), followed by DeLuxing to remove columns. Finally, the extension in Section 6.1 (the variable relaxation) is applied to relax the IP in *Step 7* and calculate the optimal solution by solving the resulting MIP.

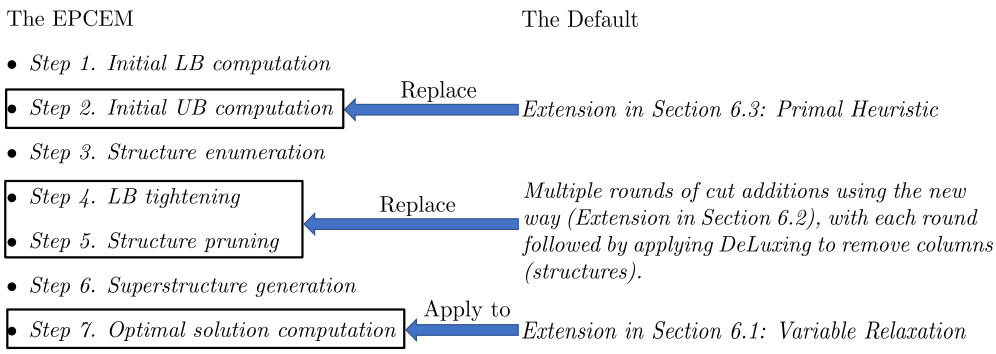

Figure 4 An illustration of the development of Default from the EPCEM in Yang (2023).

We disable each component separately on top of Default each time, and the resulting settings are denoted by NoDeluxing, NoVarRelax, OldCutAdd, and OldPrimalHeu (applying the traditional primal heuristic used in the EPCEM to compute a UB), respectively. We report the number of customers (n), the number of instances of this size (#Inst), the number of instances solved to optimality (Solved), and the average optimality gap $\frac{ub-lb}{ub} \times 100\%$ at termination (Gap%). The Gap is averaged over instances that cannot be solved optimally within the time limit.

According to Table 1 and Figure 5, Default solves significantly more instances than NoDeLuxing and OldPrimalHeu while being 48%, 21%, 161%, 115%, 18%, and 26% faster than NoDeLuxing, and 106%, 206%, 813%, 249%, 38%, and 67% faster than OldPrimalHeu, respectively, for instances of sizes 70 to 200. These results confirm the high effectiveness of DeLuxing in accelerating the algorithm and its essential contribution to solving challenging instances. Furthermore, the adoption of the primal heuristic, in which DeLuxing plays a pivotal role, significantly enhances the algorithm's capability to solve large instances by providing tight upper bounds at an early stage. Although the variable relaxation and the new approach for cutting plane addition may have limited effectiveness for small-sized instances, they prove to be valuable in achieving optimality faster for larger instances. In particular, Default outperforms NoVarRelax by solving two more instances, and is 14% faster for 140-customer instances. While Default and OldCutAdd solve the same total number of instances, Default surpasses OldCutAdd by being 25% faster for instances of size 140.

Table 1 Summary of the results for the CMTVRPTW.

n	#Inst	Default		NoDeLuxing		NoVarRelax		OldCutAdd		OldPrimalHeu	
		Solved	Gap%	Solved	Gap%	Solved	Gap%	Solved	Gap%	Solved	Gap%
70	27	27	0.0	27	0.0	27	0.0	27	0.0	27	0.0
80	27	27	0.0	27	0.0	27	0.0	27	0.0	27	0.0
100	27	27	0.0	27	0.0	27	0.0	27	0.0	23	2.6
140	30	29	0.1	27	0.4	30	0.0	30	0.0	21	2.2
170	30	22	0.5	18	0.4	21	0.5	22	0.5	15	1.6
200	30	22	0.4	17	0.5	20	0.3	21	0.4	12	1.4

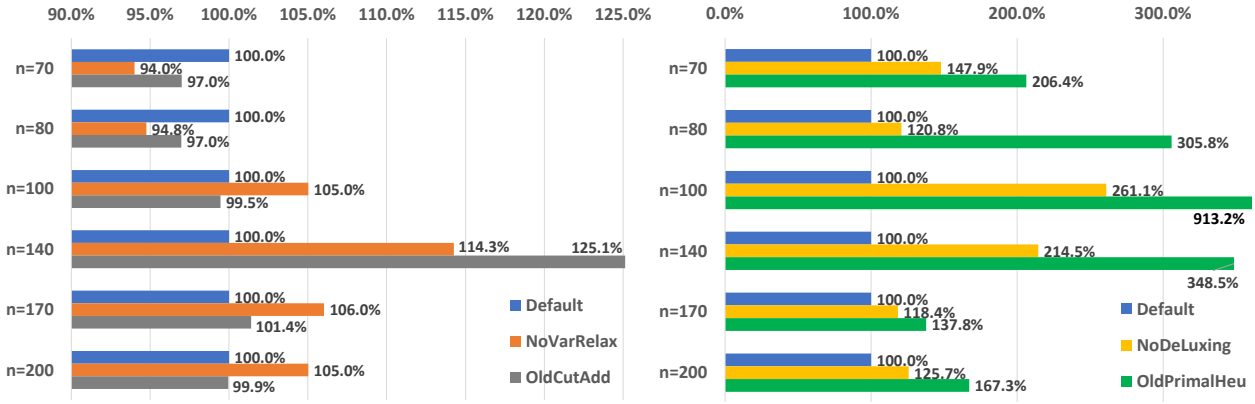

Figure 5 Comparison on CPU. Each number represents the ratio of the average CPU time compared to that of the Default method, where each average value is taken over instances of the same size. The CPU values are recorded as 10,800 for instances that cannot be solved to optimality within the 3-hour time limit.

7.3. Comparison with State-of-the-Art Algorithms

In this section, we compare our Default with three state-of-the-art algorithms: Yang (2023), Roboredo et al. (2023), and Zhang (2022). It is worth mentioning that our hardware is better than others.

To ensure a fair comparison, we scale the computational times of the other three methods based on their CPU frequencies. More precisely, the CPU frequencies reported in Yang (2023), Roboredo et al. (2023), and Zhang (2022) are 3.7GHz, 3.6GHz, and 2.9GHz, respectively, which necessitates dividing their reported times by a factor of 1.05, 1.08, and 1.34.

In Tables 2 to 5, we report the values of n, #Inst, Solved, Gap, and the computational time in seconds (CPU). Detailed results for each instance are reported in Tables EC.3 to EC.5 in Section EC.2 of the e-companion. The CPU values presented in Tables 2 to 5 are averaged over **all** instances of the same size and are scaled values for the three benchmark methods. If an instance cannot be solved to optimality within the 3-hour time limit, its CPU value is recorded as 10,800 even though it may be terminated early due to insufficient memory. Note that Zhang (2022) only reported results for instances of sizes 80 and 100. Moreover, Roboredo et al. (2023) did not experiment with the two variants considered in this paper and did not report the optimality gap at termination. Roboredo et al. (2023) reported results for two settings, i.e., with or without initial ub. For consistency with other methods, we use the setting without ub in Table 2.

7.3.1. Comparison on the CMTVRPTW As shown in Table 2, our method can solve all 81 CMTVRPTW instances optimally while being, on average, more than 10 and 7 times, respectively, as fast as Zhang (2022) for instances of sizes 70 and 100. In contrast, Yang (2023) can only solve 65 out of the 81 instances to optimality and is more than 20 times slower than our method. For a fair comparison with Roboredo et al. (2023), we conducted two additional sets of experiments: one with the optimal value as an initial upper bound and the other without the upper bound. Both experiments were run using a single thread for solving all LPs and IPs, and the results are summarized in the following Table 3. In both cases, our method can still solve all instances optimally while Roboredo et al. (2023) solves 51 and 67 instances, respectively, without/with an upper bound.

Table 2 Comparison on the CMTVRPTW using eight threads.

n	#Inst	This Paper (Default)			Yang (2023)			Zhang (2022)		
		Solved	Gap%	CPU	Solved	Gap%	CPU	Solved	Gap%	CPU
70	27	27	0.0	26.7	27	0.0	1230.3	27	0.0	343.9
80	27	27	0.0	49.6	24	1.0	2197.5	—	—	—
100	27	27	0.0	240.5	14	1.3	7122.5	27	0.0	1686.4

Table 3 Comparison on the CMTVRPTW with and without UB using one thread.

n	#Inst	Default (one thread) without UB			Default (one thread) with UB			Roboredo et al. (2023) without UB			Roboredo et al. (2023) with UB		
		Solved	Gap%	CPU	Solved	Gap%	CPU	Solved	Gap%	CPU	Solved	Gap%	CPU
70	27	27	0.0	39.0	27	0.0	20.1	22	—	3443.6	25	—	2511.3
80	27	27	0.0	83.1	27	0.0	43.5	18	—	4446.9	25	—	2261.7
100	27	27	0.0	570.0	27	0.0	589.2	11	—	6357.9	17	—	5380.1

Table 4 Comparison on the CMTVRPTW-LT.

n	#Inst	This Paper (Default)			Yang (2023)			Zhang (2022)		
		Solved	Gap%	CPU	Solved	Gap%	CPU	Solved	Gap%	CPU
70	27	27	0.0	27.4	27	0.0	1497.0	27	0.0	329.7
80	27	27	0.0	48.4	24	1.1	2558.2	—	—	—
100	27	27	0.0	362.3	13	1.2	7201.4	27	0.0	1868.4

7.3.2. Comparison on the CMTVRPTW-LT According to Table 4, our method consistently outperforms all the benchmark algorithms substantially on the CMTVRPTW-LT. Specifically, it can solve all 81 instances optimally, with computational speeds more than 10 times for 70-customer instances and over 5 times for 100-customer instances as fast as those of Zhang (2022). In contrast, Yang (2023) only solves 64 of the 81 instances optimally, and it once again takes over 20 times more time than our method.

7.3.3. Comparison on the CMTVRPTW-R Table 5 summarizes the results for 243 CMTVRPTW-R instances. Our method can solve all but one instance optimally and achieves an optimality gap of 0.3% for the only unsolved instance. In terms of computational speed, our method is, once again, significantly faster than Zhang (2022) and Yang (2023).

Table 5 Comparison on the CMTVRPTW-R.

n	κ	#Inst	This Paper (Default)			Yang (2023)			Zhang (2022)		
			Solved	Gap%	CPU	Solved	Gap%	CPU	Solved	Gap%	CPU
70	0.25	27	27	0.0	23.2	27	0.0	251.9	27	0.0	190.8
70	0.50	27	27	0.0	13.8	27	0.0	190.2	27	0.0	164.8
70	0.75	27	27	0.0	24.9	26	0.6	490.6	27	0.0	156.5
80	0.25	27	27	0.0	17.9	27	0.0	1055.8	—	—	—
80	0.50	27	27	0.0	26.4	27	0.0	433.7	—	—	—
80	0.75	27	27	0.0	50.0	27	0.0	542.7	—	—	—
100	0.25	27	27	0.0	196.8	23	2.0	3534.8	27	0.0	771.6
100	0.50	27	26	0.3	583.5	23	1.4	3140.9	27	0.0	743.4
100	0.75	27	27	0.0	182.7	21	1.6	3288.0	27	0.0	536.5

7.4. Computational Results for Large Multi-Trip Instances

In this section, we test Default on significantly larger instances with sizes twice as large as the largest ones currently documented in the literature, which are exponentially more difficult to solve. For the CMTVRPTW and CMTVRPTW-LT, the CPU of an instance whose optimality cannot be proved within the 3-hour time limit is again counted as 10,800, and the values of Gap and CPU are averaged over **all** instances. However, for some CMTVRPTW-R instances, no feasible solution can be found at termination. Such instances (3, 2, and 9 instances of sizes 140, 170, and 200, respectively; 15 in total) are excluded from the computation of Gap and CPU values. Table 6 summarizes the results and more details can be found in Tables EC.3 to EC.5 in Section EC.2 of the e-companion.

According to Table 6, all but one CMTVRPTW instance of size 140 can be solved within 3 hours, and the optimality of the only unsolved one can be proved within 5 hours. Among the 60 CMTVRPTW instances of sizes 170 and 200, 44 instances can be solved. The average gaps for the

Table 6 Computational results for large instances.

n	CMTVRPTW				CMTVRPTW-LT				CMTVRPTW-R			
	#Inst	Solved	Gap%	CPU	#Inst	Solved	Gap%	CPU	#Inst	Solved	Gap%	CPU
140	30	29	0.1	1254.0	30	28	0.2	1372.5	90	84	1.4	365.9
170	30	22	0.5	4210.7	30	21	0.5	4048.4	90	61	1.1	722.0
200	30	22	0.4	4306.3	30	21	0.5	4649.6	90	47	1.3	930.0

unsolved instances are approximately 0.5% and 0.4%, respectively. Our method achieves very similar results for the CMTVRPTW-LT: it solves all but two instances of size 140. In addition, 70% of 170- and 200-customer instances can be solved, and the average gaps of the unsolved ones are 0.5%. For the CMTVRPTW-R, around 93%, 68%, and 52% of instances of sizes 140, 170, and 200 can be solved. The average gaps of the unsolved instances are all below 1.5%. The optimality of all solved instances can be proved, on average, in less than 16 minutes. These results clearly demonstrate that the Default brings our capabilities of solving CMTVRPTW, CMTVRPTW-LT, and CMTVRPTW-R to an entirely new level.

7.5. Computational Results for CVRP and VRPTW Instances

In this section, we focus on demonstrating the effectiveness of DeLuxing and the new primal heuristic from Section 6.3 for solving two classic types of VRPs: CVRP and VRPTW. For each problem class, we conduct six sets of experiments: the basic VRP-Solver, basic RouteOpt (RouteOpt0), RouteOpt with DeLuxing (RouteOp1), RouteOpt with the traditional (old) primal heuristic (RouteOpt2), RouteOpt with our new primal heuristic (RouteOpt3), and RouteOpt with both DeLuxing and the new primal heuristic (RouteOpt4). The initial upper bound is set to $ub^0 = (1 + 0.1\%) \cdot v^*$ so that all instances can be solved to optimality while ensuring the enumeration does not often succeed at the root, where v^* is the optimal value.

It is important to note that branching proves ineffective in solving the multi-trip instances in previous sections and, thus, is not performed. In such scenarios, we apply DeLuxing aggressively to make the final IP as small as possible because the primary challenge lies in solving this IP. However, this is not the case anymore when solving the CVRP and VRPTW. Branching is indispensable, leading to the generation of many nodes and, thus, IPs to solve. Applying DeLuxing immediately after a successful enumeration at each BBN becomes inefficient. Instead, branching can initially downsize an enumerated pool rapidly. We further observe that in a later stage, branching does not reduce the pool effectively and ends up creating too many nodes.

The goal of DeLuxing, in this case, shifts to lessening dependence on branching. It closes each node early, thereby reducing the BB tree size. Specifically, DeLuxing is applied moderately at each node once the pool size falls below a set threshold, denoted as $\bar{N}$. The reduced IP can then be solved efficiently by Gurobi, and thus, the node is closed without further branching. Through experimentation, we found that setting $p = 20$, $\beta_1 = 1000$, $\beta_2 = 1000$, and $\bar{N} = 50,000$ typically halves the pool

size within 10 seconds across most instances. Consequently, these parameters are adopted across all experiments in this section.

7.5.1. Effectiveness of DeLuxing Table 7 summarizes the average time in seconds (CPU) and the tree size measured by the number of BBNs (Node). For detailed results, readers are referred to Tables EC.6 and EC.7 in Section EC.3 of the e-companion. Without DeLuxing, RouteOpt is already, on average, about 90% faster than the VRP-Solver for the CVRP instances and 42% faster for the VRPTW instances. DeLuxing further speeds up it by 21% and 31%, respectively, resulting in a nearly 2.3X speed for the CVRP and a 1.9X speed for the VRTPW compared with the VRP-Solver. This acceleration is primarily attributed to DeLuxing consistently reducing the tree size by more than two-thirds. This improvement is even more pronounced for difficult instances: the 76 CVRP and 100 VRPTW instances that take more than an hour to solve using the VRP-Solver. RouteOpt1 achieves over 2.5 times and nearly 2 times the speed of the VRP-Solver, respectively, for these CVRP and VRPTW instances.

Table 7 Computational results for the CVRP and VRPTW instances.

	n	#Inst	Solved	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
				CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
CVRP	200	200	200	4495.5	158.2	2369.9	171.5	1959.8	55.0	2342.5	166.3	1814.0	94.7	1635.5	30.4
VRPTW	300	200	200	5574.4	152.3	3915.1	449.9	2993.9	141.29	3949.5	450.5	2779.1	290.37	2293.8	88.25

7.5.2. Effectiveness of Our Primal Heuristic We observe in our experiments that the standard primal heuristic yields a solution of better value than the given initial upper bound for only 43 out of the 200 CVRP instances and 64 out of the 200 VRPTW instances. According to Table 7, its application in RouteOpt results in little difference: 1% speedup for the CVRP instances and 1% slowdown for the VRPTW instances. By comparing RouteOpt0 and Routeopt3, we conclude that our proposed primal heuristic accelerates RouteOpt by around 31% and 41%, substantially outperforming the standard primal heuristic.

7.5.3. Effectiveness of DeLuxing and Our Primal Heuristic Combined Comparing RouteOpt1 and RouteOpt4 in Table 7, we can conclude that even when DeLuxing is enabled, our primal heuristic can still accelerate RouteOpt by around 20% and 31%, respectively. When DeLuxing and our primal heuristic are both enabled, they reduce the tree size by more than 80%, making RouteOpt 45% and 71% faster. Overall, the speed is about 175% and 143%, respectively, faster than the VRP-Solver for solving the CVRP and VRPTW instances.

7.5.4. Results for Long CVRP Instances The 400 CVRP and VRPTW instances tested have moderate route lengths: the average number of customers per route is 7.1 (with a maximum of 15) for the CVRP and 7.9 (with a maximum of 10) for the VRPTW. For these instances, enumeration typically succeeds relatively early in the solution process, with the first successful enumeration occurring at around 14.3% and 4.7% of the whole process for the CVRP and VRPTW, respectively, and the last success happening at 87.1% and 84.7% of the duration. Despite early enumeration, the branching tree remains large, with average tree sizes of 171.5 and 449.9, respectively, when solved under setting RouteOpt0 (see Table 7).

In instances with long routes, successful enumeration tends to occur much later in the process. Once successful, branching can rapidly close a node, resulting in much smaller BB trees. In such cases, DeLuxing is expected to be less effective. To verify this intuition, we conducted additional experiments on long CVRP instances. To this end, we increased the capacity of the 200 CVRP instances by 80%, on average lengthening the routes to 15.6 (with a maximum of 23). We obtained optimal solutions for 65 of these instances and used them for the following analysis. As expected, the first successful enumeration occurs much later, at around 27.8%, and the last happens at 99.8% of the process. Additionally, the tree size is significantly smaller, with an average of 63.7. In this scenario, our experiments indicate that DeLuxing is no longer effective, as nodes are closed rapidly by branching following successful enumeration. Nonetheless, our primal heuristic still accelerates RouteOpt by more than 25%. Detailed results are included in Table EC.8 in Section EC.3 of the e-companion.

8. Concluding Remarks

We propose a highly effective variable fixing strategy, called DeLuxing, that employs a novel deep search method for identifying promising dual solutions. Based on theoretical results, it solves a novel LP formulation with only a small subset of the enumerated variables in each iteration. DeLuxing can remove more than 75% variables in most cases, achieving a direct acceleration of over 50%. Enhanced by the additional three extensions inspired by DeLuxing, our method can be more than 7 times on average and up to more than 20 times as fast as the best-performing exact method in the literature. In particular, our method can solve all but one CMTVRPTW instance with 140 customers in 3 hours and prove optimality for the remaining one in 5 hours, which doubles the size of previously completely solvable instances. Significant performance improvement is also achieved for the two important variants (the CMTVRPTW-LT and CMTVRPTW-R) and two classic VRPs (the CVRP and VRPTW). It is worth mentioning that the enumeration of all necessary columns early in the solution process is crucial to the success of DeLuxing. When the routes are long, DeLuxing is less effective since enumeration may not succeed until a late stage. Nonetheless, the new primal

inspired by DeLuxing performs consistently well across all tested instances in this paper, even when full enumeration only succeeds late.

Currently, in the subroutine of DeLuxing (Algorithm 2), we employ the dual simplex method or interior point method without crossover to solve the LP formulation $\widehat{\mathrm{F}}(\mathcal{R}_1, \widetilde{\mathcal{J}})$ and obtain each time one optimal dual solution for determining the set of removable columns $\mathcal{D}$. In future research, it would be beneficial to explore the possibility of recording all feasible dual solutions encountered during the solution process and utilizing them to compute LUs for further variable fixing. Another potential research direction is to investigate the similarities among columns in an artificial cluster $\mathcal{D}\backslash\widetilde{\mathcal{J}}$ and develop even more effective approaches to bundle columns for computing qualified dual solutions. In addition, extending the basic principle underpinning DeLuxing to other contexts such as the pricing algorithm and branching variable selection can potentially lead to extra acceleration. Finally, establishing theoretical guarantees, in a probabilistic sense, regarding the performance of DeLuxing under potentially mild assumptions can also be an interesting research direction.

Acknowledgments

This work is partially supported by National Science Foundation [Grant CMMI-2309667] and Alibaba Group US [Grant AGR00022274]. We would like to thank the area editor, associate editor, and three anonymous referees for providing valuable comments and suggestions.

References

Achterberg T (2018) Exploiting degeneracy in MIP. *Talk at Aussois 22nd Combinatorial Optimization Workshop*, URL `http://www.iasi.cnr.it/aussois/web/uploads/2018/slides/achterbergt.pdf`.

Achterberg T, Berthold T, Koch T, Wolter K (2008) Constraint integer programming: A new approach to integrate CP and MIP. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 5th International Conference, CPAIOR 2008 Paris, France, May 20-23, 2008 Proceedings 5*, 6–20 (Springer).

Aloise D, Deshpande A, Hansen P, Popat P (2009) NP-hardness of euclidean sum-of-squares clustering. *Machine Learning* 75:245–248.

Amaldi E, Kann V (1995) The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science* 147(1-2):181–210.

Arthur D, Vassilvitskii S (2007) K-means++ the advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.

Bacchus F, Hyttinen A, Järvisalo M, Saikko P (2017) Reduced cost fixing in maxsat. *Principles and Practice of Constraint Programming: 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28–September 1, 2017, Proceedings 23*, 641–651 (Springer).

Bajgiran OS, Cire AA, Rousseau LM (2017) A first look at picking dual variables for maximizing reduced cost fixing. *Integration of AI and OR Techniques in Constraint Programming: 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings 14*, 221–228 (Springer).

Balas E, Carrera MC (1996) A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research* 44(6):875–890.

Balas E, Saltzman MJ (1991) An algorithm for the three-index assignment problem. *Operations Research* 39(1):150–161.

Baldacci R, Bartolini E, Mingozzi A (2011a) An exact algorithm for the pickup and delivery problem with time windows. *Operations Research* 59(2):414–426.

Baldacci R, Bartolini E, Mingozzi A, Valletta A (2011b) An exact algorithm for the period routing problem. *Operations Research* 59(1):228–241.

Baldacci R, Christofides N, 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.

Baldacci R, Hadjiconstantinou E, Mingozzi A (2004) An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research* 52(5):723–738.

Baldacci R, Mingozzi A, Roberti R (2011c) New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research* 59(5):1269–1283.

Baldacci R, Mingozzi A, Roberti R (2012) New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing* 24(3):356–371.

Baldacci R, Mingozzi A, Roberti R, Calvo RW (2013) An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations Research* 61(2):298–314.

Baldacci R, Mingozzi A, Wolfler Calvo R (2011d) An exact method for the capacitated location-routing problem. *Operations Research* 59(5):1284–1296.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MW, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46(3):316–329.

Bixby ER, Fenelon M, Gu Z, Rothberg E, Wunderling R (2000) Mip: Theory and practiceclosing the gap. *System Modelling and Optimization: Methods, Theory and Applications. 19 th IFIP TC7 Conference on System Modelling and Optimization July 12–16, 1999, Cambridge, UK 19*, 19–49 (Springer).

Bixby RE (2002) Solving real-world linear programs: A decade and more of progress. *Operations Research* 50(1):3–15.

Bradley PS, Fayyad UM (1998) Refining initial points for k-means clustering. *ICML*, volume 98, 91–99 (Citeseer).

Breugem T, Dollevoet T, Huisman D (2022) Is equality always desirable? Analyzing the trade-off between fairness and attractiveness in crew rostering. *Management Science* 68(4):2619–2641.

Cappanera P, Gallo G (2004) A multicommodity flow approach to the crew rostering problem. *Operations Research* 52(4):583–596.

Cattaruzza D, Absi N, Feillet D (2016a) The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science* 50(2):676–693.

Cattaruzza D, Absi N, Feillet D (2016b) Vehicle routing problems with multiple trips. *4OR* 14(3):223–259.

Celebi ME, Kingravi HA, Vela PA (2013) A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications* 40(1):200–210.

Cheng C, Adulyasak Y, Rousseau LM (2020) Drone routing with energy function: Formulation and exact algorithm. *Transportation Research Part B: Methodological* 139:364–387.

Contardo C, Martinelli R (2014) A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization* 12:129–146.

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

Crainic TG, Maggioni F, Perboli G, Rei W (2018) Reduced cost-based variable fixing in two-stage stochastic programming. *Annals of Operations Research* 1–37.

Crowder H, Johnson EL, Padberg M (1983) Solving large-scale zero-one linear programming problems. *Operations Research* 31(5):803–834.

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* 2(4):393–410.

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. *Operations Research* 8(1):101–111.

de Lima VL, Iori M, Miyazawa FK (2023) Exact solution of network flow models with strong relaxations. *Mathematical Programming* 197(2):813–846.

Desaulniers G, Errico F, Irnich S, Schneider M (2016a) Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research* 64(6):1388–1405.

Desaulniers G, Gschwind T, Irnich S (2020) Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models. *Transportation Science* 54(5):1170–1188.

Desaulniers G, Rakke JG, Coelho LC (2016b) A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science* 50(3):1060–1076.

Engineer FG, Furman KC, Nemhauser GL, Savelsbergh MW, Song JH (2012) A branch-price-and-cut algorithm for single-product maritime inventory routing. *Operations Research* 60(1):106–122.

Ford LR, Fulkerson DR (1958) A suggested computation for maximal multi-commodity network flows. *Management Science* 5(1):97–101.

Fukasawa R, Longo H, Lysgaard J, Aragão MPd, Reis M, Uchoa E, Werneck RF (2006) Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming* 106:491–511.

Gurobi Optimization, LLC (2023) Gurobi Optimizer Reference Manual. URL `https://www.gurobi.com/documentation/10.0/refman/index.html`.

Hernandez F, Feillet D, Giroudeau R, Naud O (2014) A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4or* 12(3):235–259.

Hernandez F, Feillet D, Giroudeau R, Naud O (2016) Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research* 249(2):551–559.

Holmberg K, Yuan D (2000) A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research* 48(3):461–481.

Homberger J, Gehring H (2005) A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research* 162(1):220–238.

Hooker JN, Ottosson G, Thorsteinsson ES, Kim HJ (1999) On integrating constraint propagation and linear programming for combinatorial optimization. *AAAI/IAAI*, 136–141.

Irnich S, Desaulniers G, Desrosiers J, Hadjar A (2010) Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing* 22(2):297–313.

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

Johnson EL, Kostreva MM, Suhl UH (1985) Solving 0-1 integer programming problems arising from large scale planning models. *Operations Research* 33(4):803–819.

Kohl N, Desrosiers J, Madsen OB, Solomon MM, Soumis F (1999) 2-path cuts for the vehicle routing problem with time windows. *Transportation Science* 33(1):101–116.

Land AH, Doig AG (2010) *An automatic method for solving discrete programming problems* (Springer).

Laporte G, Nobert Y (1983) A branch and bound algorithm for the capacitated vehicle routing problem. *Operations Research Spektrum* 5:77–85.

Liguori PH, Mahjoub AR, Marques G, Sadykov R, Uchoa E (2023) Nonrobust strong knapsack cuts for capacitated location routing and related problems. *Operations Research* 71(5):1577–1595.

Lima I, Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A, Oliveira D, Queiroga E (2014) Cvrplib: Capacitated vehicle routing problem library. URL `http://vrp.galgos.inf.puc-rio.br/index.php/en/`.

Lloyd S (1982) Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28(2):129–137.

Lysgaard J, Letchford AN, Eglese RW (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming* 100:423–445.

Marques G, Sadykov R, Deschamps JC, Dupas R (2020) An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Computers & Operations Research* 114:104833.

Mingozzi A, Roberti R, Toth P (2013) An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing* 25(2):193–207.

Paradiso R, Roberti R, Laganá D, Dullaert W (2020) An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research* 68(1):180–198.

Pecin D, Contardo C, Desaulniers G, Uchoa E (2017a) New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing* 29(3):489–502.

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

Pessoa A, Sadykov R, Uchoa E, Vanderbeck F (2020) A generic exact solver for vehicle routing and related problems. *Mathematical Programming* 183(1):483–523.

Pessoa A, Uchoa E, De Aragão MP, Rodrigues R (2010) Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation* 2:259–290.

Pessoa A, Uchoa E, Poggi de Aragão M (2009) A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks: An International Journal* 54(4):167–177.

Posta M, Ferland JA, Michelon P (2012) An exact method with variable fixing for solving the generalized assignment problem. *Computational Optimization and Applications* 52:629–644.

Quesnel F, Desaulniers G, Soumis F (2020) Improving air crew rostering by considering crew preferences in the crew pairing problem. *Transportation Science* 54(1):97–114.

Roberti R, Ruthmair M (2021) Exact methods for the traveling salesman problem with drone. *Transportation Science* 55(2):315–335.

Roboredo M, Sadykov R, Uchoa E (2023) Solving vehicle routing problems with intermediate stops using vrpsolver models. *Networks* 81(3):399–416.

Sadykov R, Uchoa E, Pessoa A (2021) A bucket graph–based labeling algorithm with application to vehicle routing. *Transportation Science* 55(1):4–28.

Sellmann M (2004) Theoretical foundations of CP-based Lagrangian relaxation. *Principles and Practice of Constraint Programming–CP 2004: 10th International Conference, CP 2004, Toronto, Canada, September 27-October 1, 2004. Proceedings 10*, 634–647 (Springer).

Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A (2017) New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research* 257(3):845–858.

Wolsey LA, Nemhauser GL (1999) *Integer and combinatorial optimization*, volume 55 (John Wiley & Sons).

Yang Y (2023) An exact price-cut-and-enumerate method for the capacitated multitrip vehicle routing problem with time windows. *Transportation Science* 57(1):230–251.

You Z, Yang Y, Wang X, Yin W (2023) Two-stage learning to branch in branch-price-and-cut algorithms for solving vehicle routing problems exactly. Available at SSRN: `https://ssrn.com/abstract=4630549`. Under major revision at *Operations Research* .

Yunes T, Aron ID, Hooker JN (2010) An integrated solver for optimization problems. *Operations Research* 58(2):342–356.

Zhang S (2022) *Solving the capacitated multi-trip vehicle routing problem with time windows*. MPhil Thesis, Hong Kong Polytechnic University.

Online Supplement

EC.1. Detailed Results for the First Set of Experiments

Table EC.1 presents the detailed results for each individual instance of the first set of experiments in Section 7.1. The following information is included: the instance name (Name), the number of customers (n), the size of the enumerated column pool ($|\underline{\mathcal{R}}|$), the number of columns with reduced costs not exceeding half of the gap ($|\underline{\mathcal{R}}_2^{\leqslant}|$), the number of columns with reduced costs more than half the gap ($|\underline{\mathcal{R}}_2^{>}|$), the percentage of columns removed by each method ($\mathcal{D}\%$), and the computational time of each instance in seconds (CPU).

Table EC.1: Detailed results for the first set of experiments.

Name	n	$\lvert\underline{\mathcal{R}}\rvert$	$\lvert\underline{\mathcal{R}}_2^{\leqslant}\rvert$	$\lvert\underline{\mathcal{R}}_2^{>}\rvert$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
					$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU
C201	70	69,562	10,625	58,937	84.7	2.7	88.3	16.9	79.7	1.6	76.8	1.3	77.8	2.1
C202	70	167,231	14,858	152,373	91.1	7.9	91.7	104.3	88.6	4.5	85.9	3.9	86.9	5.3
C203	70	2,148	781	1,367	63.6	0.6	91.2	0.8	61.2	0.3	61.6	0.3	59.3	0.6
C204	70	2,384	1,215	1,169	49.0	0.7	89.1	0.9	71.4	0.4	63.4	0.3	51.3	0.7
C205	70	241,821	13,520	228,301	94.4	10.9	95.0	188.0	93.1	5.7	91.8	4.8	90.8	8.9
C206	70	198,754	17,205	181,549	91.3	12.1	92.7	131.3	86.4	5.9	87.3	5.3	87.6	7.1
C207	70	232,952	32,147	200,805	86.2	16.6	93.0	216.0	84.0	7.2	88.8	8.6	84.7	9.9
C208	70	2,590	807	1,783	68.8	0.5	92.0	0.9	61.7	0.3	62.8	0.2	65.4	0.5
R201	70	11,097	1,724	9,373	84.5	0.8	92.9	2.6	82.6	0.5	78.7	0.4	83.0	0.6
R202	70	61,021	8,917	52,104	85.4	4.0	88.3	27.4	81.1	2.6	78.6	1.9	80.2	2.6
R203	70	54,387	5,080	49,307	90.7	3.5	91.5	30.4	86.5	2.0	83.6	1.9	86.7	2.6
R204	70	108,401	11,160	97,241	89.7	9.0	90.9	113.7	86.5	5.6	83.9	4.3	84.2	5.4
R205	70	86,765	14,008	72,757	83.9	8.9	83.9	82.2	78.3	5.1	73.0	3.9	76.1	5.0
R206	70	111,271	12,828	98,443	88.5	9.5	89.6	126.7	84.4	5.3	82.8	4.8	82.0	5.8
R207	70	65,745	7,951	57,794	87.9	6.1	89.7	43.0	83.5	4.0	80.8	3.0	82.4	3.7
R208	70	113,628	12,608	101,020	88.9	7.5	90.6	100.2	87.6	4.9	84.5	3.8	84.8	4.6
R209	70	37,046	4,220	32,826	88.6	2.9	90.6	18.4	84.8	1.8	81.2	1.5	84.5	2.1
R210	70	52,710	5,888	46,822	88.8	5.2	90.3	47.3	84.0	3.2	81.6	3.0	82.4	3.8
R211	70	69,533	8,444	61,089	87.9	4.9	89.3	39.1	85.1	2.8	81.9	2.4	83.3	3.0
RC201	70	5,806	1,343	4,463	76.9	0.6	88.8	1.1	81.8	0.4	76.9	0.3	76.3	0.5
RC202	70	6,423	1,555	4,868	75.8	0.6	82.5	1.4	70.5	0.4	66.5	0.3	72.4	0.5
RC203	70	19,780	4,251	15,529	78.5	1.8	86.8	7.4	74.8	1.1	69.2	1.0	73.8	1.3
RC204	70	41,386	4,973	36,413	88.0	3.4	91.5	14.9	87.5	2.1	82.3	1.8	81.8	2.6
RC205	70	9,659	1,955	7,704	79.8	1.5	81.8	4.6	74.1	1.1	65.2	0.9	74.6	1.3
RC206	70	12,448	1,526	10,922	87.7	1.2	89.6	5.6	87.7	0.7	82.2	0.9	83.6	1.2
RC207	70	28,008	3,394	24,614	87.9	2.4	89.6	13.1	86.4	1.6	82.4	1.3	82.9	1.8
RC208	70	48,135	5,936	42,199	87.7	3.9	91.3	24.1	86.4	2.8	82.7	2.4	80.6	3.1
C201	80	295,358	32,044	263,314	89.2	15.7	94.7	216.1	88.5	7.2	91.2	6.3	87.0	11.3
C202	80	356	153	203	57.0	0.1	80.9	0.2	68.8	0.1	62.4	0.1	57.0	0.1
C203	80	484	280	204	42.2	0.2	77.3	0.3	41.3	0.1	38.4	0.1	42.2	0.2
C204	80	656,380	37,455	618,925	94.3	17.8	95.6	184.4	93.7	9.1	92.7	8.0	91.3	11.8
C205	80	264	20	244	92.4	0.0	92.4	0.0	92.4	0.0	92.4	0.0	92.4	0.0
C206	80	380,825	25,544	355,281	93.3	8.9	96.7	79.8	92.9	4.9	93.5	4.2	92.2	6.4
C207	80	389,157	46,998	342,159	87.9	17.8	94.2	195.9	87.3	8.2	91.0	7.0	85.5	11.7
C208	80	449,968	33,450	416,518	92.6	9.8	95.2	112.0	93.3	5.4	92.9	4.9	92.6	6.7
R201	80	68,398	4,887	63,511	92.9	4.2	93.3	42.8	88.5	2.5	87.4	2.2	88.9	3.2
R202	80	70,136	8,662	61,474	87.7	5.1	88.9	40.9	84.0	2.7	80.4	2.2	81.9	3.4
R203	80	121,315	9,427	111,888	92.2	8.1	92.4	129.8	89.6	5.4	88.8	4.8	87.8	5.5
R204	80	10,718	4,375	6,343	59.2	5.1	78.2	12.3	56.6	2.9	46.6	2.3	60.9	3.6
R205	80	118,485	16,211	102,274	86.3	11.9	86.7	124.5	81.2	6.9	77.4	5.0	78.9	6.5
R206	80	136,015	16,004	120,011	88.2	15.9	89.2	237.2	84.0	8.9	82.4	7.4	81.8	8.8

Continued on next page

Table EC.1 – *Continued from previous page*

Name	n	$\lvert\underline{\mathcal{R}}\rvert$	$\lvert\underline{\mathcal{R}}_2^{\leqslant}\rvert$	$\lvert\underline{\mathcal{R}}_2^{>}\rvert$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
					$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU
R207	80	193,472	15,664	177,808	91.9	17.9	92.1	392.6	88.9	10.7	87.3	7.9	86.2	11.0
R208	80	1,041	648	393	37.8	0.8	82.2	1.1	38.6	0.5	50.5	0.4	37.8	0.8
R209	80	113,636	12,240	101,396	89.2	12.3	90.0	155.4	83.9	6.4	82.8	5.7	82.8	7.3
R210	80	98,047	11,839	86,208	87.9	10.5	88.6	139.5	83.8	6.8	78.9	5.5	81.2	6.1
R211	80	12,271	4,371	7,900	64.4	4.4	74.5	13.1	54.8	2.5	50.7	2.1	58.1	3.3
RC201	80	7,950	1,065	6,885	86.6	0.7	90.7	2.4	83.0	0.4	81.9	0.4	82.5	0.7
RC202	80	13,712	2,324	11,388	83.1	1.3	87.1	5.1	79.7	0.8	76.2	0.8	79.7	1.1
RC203	80	24,426	3,487	20,939	85.7	1.3	87.4	5.9	83.3	0.8	78.3	0.7	81.2	1.0
RC204	80	39,195	4,798	34,397	87.8	1.7	93.1	6.8	88.1	1.2	87.9	1.0	87.3	1.3
RC205	80	21,824	4,542	17,282	79.2	2.6	83.4	11.6	76.5	1.6	70.2	1.6	76.3	1.9
RC206	80	20,353	2,746	17,607	86.5	2.1	89.0	10.1	84.7	1.3	80.2	1.3	82.9	2.0
RC207	80	3,261	1,740	1,521	46.6	0.6	69.2	1.0	43.0	0.4	34.5	0.3	46.0	0.6
RC208	80	36,018	5,704	30,314	84.2	1.7	91.9	5.4	85.0	1.1	83.9	1.1	82.5	1.2
C201	100	313,754	14,701	299,053	95.3	16.2	95.7	336.9	95.2	9.7	93.9	7.3	92.9	13.1
C202	100	596	256	340	57.1	0.2	89.1	0.3	58.2	0.2	57.6	0.2	57.1	0.2
C203	100	735,149	38,193	696,956	94.8	17.6	95.2	506.4	94.8	17.7	94.0	14.6	90.3	7.3
C204	100	827	158	669	80.9	0.1	93.1	0.2	73.0	0.1	84.8	0.1	77.4	0.1
C205	100	243	51	192	79.0	0.1	76.1	0.2	86.0	0.1	81.9	0.1	79.0	0.1
C206	100	334,947	22,312	312,635	93.3	15.3	94.4	158.9	92.2	7.8	91.5	6.5	89.4	11.0
C207	100	318	38	280	88.1	0.0	89.9	0.1	64.2	0.1	89.3	0.0	88.1	0.0
C208	100	290	61	229	79.0	0.1	83.8	0.1	85.5	0.0	83.8	0.0	79.0	0.1
R201	100	362,623	39,147	323,476	89.2	32.0	89.8	937.2	87.8	21.4	86.2	15.7	80.7	16.7
R202	100	892,622	128,224	764,398	85.6	121.0	86.2	4230.1	85.6	124.6	82.6	78.6	61.7	21.3
R203	100	43,969	20,005	23,964	54.5	20.6	59.7	75.8	44.9	12.0	36.7	8.5	45.8	11.0
R204	100	18,178	6,414	11,764	64.7	11.5	85.5	32.2	60.3	5.9	57.4	5.2	63.7	7.4
R205	100	490,622	59,700	430,922	87.8	52.4	88.3	1653.6	85.4	29.3	82.3	21.5	78.7	23.0
R206	100	541,592	103,294	438,298	80.9	53.3	80.8	1591.4	80.9	53.4	77.4	37.2	56.5	9.6
R207	100	32,326	8,648	23,678	73.3	18.8	79.6	76.4	64.8	12.1	55.0	9.8	65.3	10.8
R208	100	10,876	3,003	7,873	72.4	5.8	83.4	17.0	65.7	3.7	57.6	3.3	66.9	4.5
R209	100	15,708	5,850	9,858	62.8	7.0	71.8	20.9	48.6	4.3	39.4	3.4	55.9	4.9
R210	100	27,332	9,770	17,562	64.3	16.3	76.1	58.5	54.3	9.5	50.4	8.1	57.1	10.1
R211	100	73,958	15,480	58,478	79.1	29.2	84.9	165.7	73.7	18.8	72.4	14.2	71.5	13.8
RC201	100	9,551	2,368	7,183	75.2	1.2	79.7	3.2	70.8	0.7	65.5	0.5	71.3	0.9
RC202	100	100,395	14,447	85,948	85.6	9.1	86.3	95.2	81.6	5.6	76.1	3.8	77.9	5.5
RC203	100	108,682	8,109	100,573	92.5	8.7	93.2	113.8	89.6	5.5	88.3	3.8	87.6	6.4
RC204	100	235,679	14,422	221,257	93.9	14.3	95.0	317.6	93.1	9.0	91.9	7.4	90.7	10.5
RC205	100	188,739	29,647	159,092	84.3	27.3	84.6	444.9	79.3	15.6	76.0	12.2	74.5	13.7
RC206	100	109,935	16,990	92,945	84.6	10.1	84.9	119.4	79.7	5.4	76.7	4.2	78.0	6.5
RC207	100	60,775	8,995	51,780	85.2	7.0	87.9	54.4	82.4	4.6	78.8	3.5	81.4	4.3
RC208	100	208,715	18,462	190,253	91.2	17.3	92.9	255.4	89.0	10.8	88.0	8.8	85.6	11.6
C2_2_01	140	30,507	6,467	24,040	78.8	4.5	86.3	15.8	76.5	2.8	72.8	2.4	75.0	3.5
C2_2_02	140	79,763	16,182	63,581	79.7	13.9	81.0	72.3	73.4	7.4	70.1	6.8	71.5	8.3
C2_2_03	140	103,992	12,320	91,672	88.2	11.4	91.2	116.6	88.2	7.5	85.5	6.5	84.7	8.1
C2_2_04	140	1,470	837	633	43.1	1.0	76.1	1.6	43.3	0.7	33.3	0.7	43.1	1.0
C2_2_05	140	39,657	7,239	32,418	81.8	5.3	82.9	23.1	78.6	3.6	72.6	2.8	76.3	3.9
C2_2_06	140	114,300	22,663	91,637	80.2	15.9	80.8	196.2	77.3	8.6	72.2	6.7	71.7	10.0
C2_2_07	140	86,989	14,821	72,168	83.0	9.4	83.9	60.1	79.7	6.2	78.4	4.3	76.6	5.9
C2_2_08	140	214,605	38,124	176,481	82.2	27.0	82.6	455.7	79.3	15.8	76.3	11.8	74.3	14.2
C2_2_09	140	398,209	57,192	341,017	85.6	55.6	86.0	1066.7	84.5	34.9	80.7	22.3	76.8	29.4
C2_2_10	140	227,213	43,288	183,925	81.0	29.6	81.6	366.9	79.8	19.4	75.1	12.7	71.7	17.0
R2_2_01	140	193,870	42,937	150,933	77.9	20.2	78.4	336.4	74.4	10.8	70.7	7.6	67.3	11.3
R2_2_02	140	84,668	19,590	65,078	76.9	8.8	77.6	53.4	73.8	5.1	68.5	3.7	68.7	5.0
R2_2_03	140	166,339	46,093	120,246	72.3	61.3	73.2	859.6	67.7	39.5	62.4	27.0	61.4	28.1
R2_2_04	140	56,430	10,471	45,959	81.4	4.9	86.6	50.1	80.7	3.6	78.9	2.3	80.8	3.2
R2_2_05	140	278,203	46,275	231,928	83.4	32.4	83.5	648.9	81.2	17.7	78.3	12.9	75.0	17.8
R2_2_06	140	552,662	93,918	458,744	83.0	35.4	83.6	1132.3	83.0	35.4	79.4	24.7	67.9	7.1
R2_2_07	140	882,247	126,877	755,370	85.6	62.6	86.0	2748.0	85.6	62.5	81.5	37.4	67.8	12.4
R2_2_08	140	37,185	8,463	28,722	77.2	3.6	85.6	14.9	78.4	2.4	72.9	1.8	74.2	2.5
R2_2_09	140	379,979	76,850	303,129	79.8	50.1	80.9	987.4	78.8	32.9	74.5	19.1	69.4	24.5
R2_2_10	140	974,093	189,266	784,827	80.6	59.0	81.2	2137.2	80.6	59.0	78.4	41.0	56.6	10.2
RC2_2_01	140	329,296	39,548	289,748	88.0	44.1	88.7	987.4	86.6	28.1	83.1	18.5	80.1	22.2
RC2_2_02	140	737,639	178,492	559,147	75.8	271.3	77.1	4722.2	75.8	285.1	74.8	237.0	33.7	31.8

Continued on next page

Table EC.1 – *Continued from previous page*

Name	n	$\lvert\underline{\mathcal{R}}\rvert$	$\lvert\underline{\mathcal{R}}_2^{\leqslant}\rvert$	$\lvert\underline{\mathcal{R}}_2^{>}\rvert$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
					$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU
RC2_2_03	140	234,689	33,056	201,633	85.9	97.3	87.5	1655.6	84.7	65.7	77.9	45.1	77.9	50.5
RC2_2_04	140	16,701	5,352	11,349	68.0	12.1	83.9	30.6	62.5	8.1	59.1	6.0	62.2	8.2
RC2_2_05	140	643,875	99,072	544,803	84.6	70.7	85.4	2130.0	84.6	70.8	80.8	43.0	65.1	16.1
RC2_2_06	140	1,073,216	151,743	921,473	85.9	326.9	85.8	8411.0	85.9	337.0	81.6	215.9	56.1	43.0
RC2_2_07	140	229,745	33,752	195,993	85.3	37.2	85.8	557.5	82.4	24.2	78.0	18.1	76.9	18.7
RC2_2_08	140	349,177	49,715	299,462	85.8	137.9	86.1	3290.1	81.1	81.3	77.6	66.8	75.9	67.4
RC2_2_09	140	221,394	30,244	191,150	86.3	84.9	86.3	1567.5	85.0	51.8	78.1	36.7	79.1	49.4
RC2_2_10	140	756,533	118,577	637,956	84.3	91.9	85.4	2145.1	84.3	89.7	80.2	56.0	69.7	19.5
C2_2_01	170	162,493	33,815	128,678	79.2	33.7	82.9	306.3	77.0	20.7	73.7	12.4	71.5	18.1
C2_2_02	170	115,713	19,386	96,327	83.3	24.9	89.9	161.1	82.5	12.1	82.2	10.2	79.7	15.7
C2_2_03	170	11,130	4,135	6,995	62.9	5.3	74.4	10.4	55.8	3.4	46.4	2.8	53.1	3.7
C2_2_04	170	10,131	3,163	6,968	68.8	4.8	81.9	11.6	68.8	3.8	59.4	3.2	62.4	3.9
C2_2_05	170	289,587	41,474	248,113	85.7	48.0	87.4	702.6	84.1	28.0	82.2	20.0	78.1	24.9
C2_2_06	170	77,616	16,300	61,316	79.0	10.4	84.5	51.8	77.9	6.1	77.6	5.4	74.4	7.6
C2_2_07	170	91,257	16,338	74,919	82.1	11.4	85.4	64.6	81.3	7.1	78.0	5.8	77.9	7.5
C2_2_08	170	104,173	12,405	91,768	88.1	16.2	90.8	82.5	84.9	6.6	83.4	6.8	82.9	10.8
C2_2_09	170	1,427	878	549	38.5	1.1	78.7	1.3	47.2	0.7	38.4	0.7	38.5	1.2
C2_2_10	170	547	338	209	38.2	0.4	75.7	0.6	38.9	0.4	75.5	0.3	38.2	0.4
R2_2_01	170	870,862	236,740	634,122	72.8	219.7	73.6	7473.0	72.8	217.9	69.6	158.8	40.0	20.2
R2_2_02	170	1,075,281	259,338	815,943	75.9	258.8	75.9	9132.7	75.9	266.1	70.0	164.2	38.2	25.3
R2_2_03	170	119,149	19,171	99,978	83.9	21.6	87.8	169.4	81.4	13.7	79.8	10.2	78.5	12.4
R2_2_04	170	440,199	50,925	389,274	88.4	65.3	89.2	1265.1	86.6	39.6	83.3	22.5	81.9	36.4
R2_2_05	170	1,108,306	309,317	798,989	72.1	244.0	71.5	7546.1	72.1	253.1	66.8	161.5	31.2	20.0
R2_2_06	170	1,458,809	323,333	1,135,476	77.8	396.8	71.7	10800.4	77.8	422.7	75.3	323.8	43.2	35.3
R2_2_07	170	264,562	34,200	230,362	87.1	36.3	88.1	585.9	86.4	20.9	82.5	14.7	80.7	24.0
R2_2_08	170	432,248	52,062	380,186	88.0	53.9	88.5	1056.9	85.8	28.7	83.1	17.3	80.4	33.0
R2_2_09	170	922,050	241,016	681,034	73.9	167.7	74.3	5554.0	73.9	174.5	71.4	142.9	33.6	20.4
R2_2_10	170	929,278	189,258	740,020	79.6	78.0	80.3	3292.4	79.6	77.9	77.7	57.1	55.6	11.0
RC2_2_01	170	2,510,141	622,751	1,887,390	75.2	1345.2	68.9	10810.1	75.2	1333.1	71.5	829.1	38.3	62.6
RC2_2_02	170	140,353	30,900	109,453	78.0	67.5	78.2	612.1	71.7	39.8	66.4	29.9	69.0	34.4
RC2_2_03	170	2,363,139	561,610	1,801,529	76.2	1321.1	68.5	10827.3	76.2	1326.1	72.5	766.5	44.7	105.1
RC2_2_04	170	262,193	55,361	206,832	78.9	114.5	79.7	2000.0	76.6	88.9	68.8	59.6	66.8	56.5
RC2_2_05	170	414,124	76,137	337,987	81.6	272.6	82.4	5375.8	80.0	212.0	73.9	113.7	71.2	115.4
RC2_2_06	170	221,690	46,266	175,424	79.1	130.7	79.8	2124.0	76.8	91.3	69.1	61.5	69.7	63.8
RC2_2_07	170	513,743	95,088	418,655	81.5	209.3	82.0	4823.2	81.5	214.5	78.6	147.3	57.0	54.3
RC2_2_08	170	2,680,336	603,111	2,077,225	77.5	2127.7	61.5	10898.0	77.5	2160.4	74.0	1413.5	43.3	149.0
RC2_2_09	170	113,256	28,185	85,071	75.1	30.8	81.0	176.2	74.0	21.5	64.9	11.0	68.4	17.8
RC2_2_10	170	868,556	192,761	675,795	77.8	478.8	78.7	10818.8	77.8	484.0	70.8	223.8	48.8	67.5
C2_2_01	200	14,760	5,496	9,264	62.8	5.6	73.6	15.7	63.3	4.4	53.1	3.6	60.3	4.6
C2_2_02	200	24,626	12,246	12,380	50.3	18.9	56.3	46.1	40.2	10.9	26.8	7.8	39.5	11.6
C2_2_03	200	7,098	3,777	3,321	46.8	6.3	74.9	10.2	44.4	3.8	39.2	3.7	46.0	5.5
C2_2_04	200	52,665	15,463	37,202	70.6	32.8	76.3	123.4	69.4	27.3	59.6	20.1	62.4	19.7
C2_2_05	200	8,922	4,420	4,502	50.5	4.6	72.0	8.7	48.5	2.7	42.5	2.5	48.8	3.7
C2_2_06	200	15,128	7,789	7,339	48.5	8.0	58.9	18.1	41.9	5.2	28.0	3.9	43.1	5.9
C2_2_07	200	1,215	796	419	34.5	1.4	74.3	1.9	32.4	0.9	31.9	0.9	34.5	1.5
C2_2_08	200	652	471	181	27.8	0.7	71.0	0.9	21.5	0.5	40.2	0.4	27.8	0.7
C2_2_09	200	191,241	26,298	164,943	86.3	32.3	86.6	416.0	84.7	21.1	80.1	15.8	78.6	20.0
C2_2_10	200	1,152	710	442	38.4	1.3	73.5	1.9	30.8	0.9	38.5	0.8	38.4	1.3
R2_2_01	200	2,186,524	522,063	1,664,461	76.1	1179.5	65.5	10819.9	76.1	1185.7	72.2	746.3	35.5	58.8
R2_2_02	200	317,183	72,181	245,002	77.2	144.1	77.9	3365.9	74.1	83.0	69.8	60.2	66.0	69.2
R2_2_03	200	99,222	17,168	82,054	82.7	18.8	84.7	151.9	80.1	11.5	74.7	8.3	75.5	12.6
R2_2_04	200	169,281	32,663	136,618	80.7	81.0	81.2	1026.9	78.1	59.2	76.0	45.7	70.0	46.8
R2_2_05	200	1,265,511	276,063	989,448	78.2	427.4	75.8	10814.5	78.2	430.9	74.4	276.4	47.1	38.4
R2_2_06	200	749,779	179,829	569,950	76.0	254.5	76.5	7896.9	76.0	255.8	73.0	184.4	45.3	29.5
R2_2_07	200	579,843	86,053	493,790	85.2	54.1	85.5	1899.1	85.2	54.0	82.7	38.6	67.6	11.8
R2_2_08	200	173,780	34,160	139,620	80.3	87.7	81.1	1222.2	76.8	65.0	71.4	48.2	68.9	50.5
R2_2_09	200	4,020,994	909,531	3,111,463	77.4	3015.4	62.5	10814.3	77.4	3054.6	73.9	1924.0	37.7	108.4
R2_2_10	200	267,272	64,256	203,016	76.0	103.9	76.9	1953.5	73.5	66.7	69.4	50.5	62.6	47.6
RC2_2_01	200	776,563	92,601	683,962	88.1	93.1	88.1	3603.9	88.1	93.1	86.8	72.2	70.6	26.3
RC2_2_02	200	240,386	56,893	183,493	76.3	194.7	77.7	2636.2	73.6	124.6	67.3	76.9	65.7	81.3
RC2_2_03	200	437,521	79,433	358,088	81.8	133.5	82.0	2246.3	79.3	100.0	74.7	65.0	70.0	57.6
RC2_2_04	200	769,750	112,560	657,190	85.4	109.3	87.3	3112.8	85.4	111.4	81.8	82.7	63.4	25.0

Continued on next page

Table EC.1 – *Continued from previous page*

Name	n	$\lvert\underline{\mathcal{R}}\rvert$	$\lvert\underline{\mathcal{R}}_2^{\leqslant}\rvert$	$\lvert\underline{\mathcal{R}}_2^{>}\rvert$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
					$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU
RC2_2_05	200	1,016,354	202,905	813,449	80.0	459.1	80.3	10800.3	80.0	461.3	78.6	350.6	44.0	68.8
RC2_2_06	200	1,926,998	371,656	1,555,342	80.7	1182.1	75.5	10818.7	80.7	1177.3	77.4	806.5	42.9	148.2
RC2_2_07	200	1,654,753	318,170	1,336,583	80.8	1500.4	74.2	10807.4	80.8	1538.0	74.5	794.1	57.3	172.4
RC2_2_08	200	698,876	103,638	595,238	85.2	317.7	84.8	8399.9	85.2	320.8	79.3	183.5	65.7	65.9
RC2_2_09	200	1,839,184	310,862	1,528,322	83.1	866.3	80.5	10847.6	83.1	900.3	78.8	517.7	58.3	120.8
RC2_2_10	200	98,714	19,487	79,227	80.3	70.1	82.4	493.6	77.6	67.4	68.1	57.0	69.1	48.4

Table EC.2: Detailed results for the sensitivity analysis.

p	β_1	β_2	$\mathcal{D}\%$	CPU	p	β_1	β_2	$\mathcal{D}\%$	CPU
10	100,000	10	75.1	377.9	50	100,000	10	78.6	496.9
10	100,000	50	71.6	222	50	100,000	50	75.4	252.5
10	100,000	100	69.5	166	50	100,000	100	73.2	186.4
10	200,000	10	75	382.3	50	200,000	10	78.7	489.4
10	200,000	50	71.4	222.6	50	200,000	50	75.8	249.7
10	200,000	100	69.5	167.3	50	200,000	100	73.8	187.3
10	500,000	10	74.8	382.2	50	500,000	10	78.8	495.2
10	500,000	50	71	220.9	50	500,000	50	76.2	259.9
10	500,000	100	69.4	168.6	50	500,000	100	74.2	193.7
10	1,000,000	10	74.7	373.8	50	1,000,000	10	79.1	534.9
10	1,000,000	50	71.1	219.3	50	1,000,000	50	76.7	286
10	1,000,000	100	69.5	166	50	1,000,000	100	74.9	218.9
20	100,000	10	76.4	449.2	100	100,000	10	80.1	574.2
20	100,000	50	73.1	237.8	100	100,000	50	76.8	293.7
20	100,000	100	71.2	179.4	100	100,000	100	74.9	218.4
20	200,000	10	76.3	436.9	100	200,000	10	80.2	588.1
20	200,000	50	73.1	233.4	100	200,000	50	77.5	300
20	200,000	100	71.4	177	100	200,000	100	75.8	220.7
20	500,000	10	76.2	439.8	100	500,000	10	80.4	605.1
20	500,000	50	73.2	236.9	100	500,000	50	78.3	317.3
20	500,000	100	71.5	179.6	100	500,000	100	76.8	238.5
20	1,000,000	10	76.3	447.6	100	1,000,000	10	80.8	648
20	1,000,000	50	73.5	247.4	100	1,000,000	50	78.9	353.4
20	1,000,000	100	71.9	190.5	100	1,000,000	100	77.6	274.5

EC.2. Detailed Results for the Experiments in Sections 7.3 and 7.4

Tables EC.3 to EC.5 present the detailed results for each individual instance of the two sets of experiments in Sections 7.3 and 7.4. We report the instance name (Name), the number of customers (n), the upper bound (ub), and the optimality gap $\frac{ub-lb}{ub} \times 100$ at termination (Gap) for the CMTVRPTW-LT and CMTVRPTW-R. For the CMTVRPTW, we additionally report the total number of CG iterations before the finishing phase (CG), the number of branch-and-bound nodes in the finishing phase (Node), the size of the enumerated pool (Pool), the time spent on the column and cut generation (t_{CCG}), the time spent on column enumeration (t_{ENU}), the time spent on DeLuxing (t_{DX}), the time spent on solving the IP in the primal heuristic (t_{IP1}), the time spent on solving the final IP (t_{IP2}), and the total computation time (CPU). All times are in seconds. It is worth mentioning the algorithm may terminate before reaching the 3-hour time limit due to insufficient memory. In this case, the reported CPU corresponds to the elapsed time. When the information about an entry in the table is not available at termination, it is reported as "—".

Table EC.3: Detailed results for the CMTVRPTW.

Name	n	ub	Gap%	CG	Node	Pool	t_{CCG}	t_{ENU}	t_{DX}	t_{IP1}	t_{IP2}	CPU
C201	70	1052.2	0.00	143	0	69,562	0.9	0.1	3.4	0.3	0.0	4.7
C202	70	1047.7	0.00	144	0	167,231	2.0	0.3	10.9	5.9	0.0	19.4
C203	70	1040.4	0.00	190	24	286,653	3.6	0.9	19.4	3.5	0.4	28.4
C204	70	1036.8	0.00	226	1	549,100	4.8	2.9	31.7	3.8	0.8	45.3
C205	70	1047.9	0.00	161	0	241,821	3.8	0.9	14.9	2.7	0.0	22.9
C206	70	1042.0	0.00	189	0	198,754	2.6	0.6	14.8	4.2	0.0	22.6
C207	70	1040.3	0.00	191	0	232,952	3.0	0.7	15.7	15.2	0.0	35.1
C208	70	1040.3	0.00	177	21	185,324	3.0	0.6	14.1	10.5	0.5	29.0
R201	70	1118.4	0.00	111	0	11,097	2.8	0.9	0.8	0.5	0.0	5.1
R202	70	1041.1	0.00	140	3,068	61,021	8.1	7.8	8.8	3.2	3.6	32.1
R203	70	958.0	0.00	157	0	54,387	10.0	9.4	4.2	2.3	0.0	26.7
R204	70	921.8	0.00	216	0	108,401	9.0	11.1	10.2	6.1	0.0	37.1
R205	70	1033.4	0.00	132	7,334	86,765	10.9	10.9	22.3	10.6	33.9	89.7
R206	70	985.9	0.00	156	3,910	111,271	15.2	22.7	24.8	9.3	20.3	94.7
R207	70	942.0	0.00	190	0	65,745	7.6	9.0	6.2	5.8	0.0	29.2
R208	70	917.5	0.00	204	0	113,628	8.4	10.5	8.9	5.7	0.0	34.1
R209	70	955.3	0.00	170	0	37,046	9.7	7.7	3.4	2.8	0.0	24.4
R210	70	980.4	0.00	170	0	52,710	8.9	6.7	6.6	4.2	0.0	27.0
R211	70	914.8	0.00	210	0	69,533	5.7	5.0	6.2	2.8	0.0	19.9
RC201	70	1364.5	0.00	88	0	5,806	1.8	0.6	0.6	0.5	0.0	3.4
RC202	70	1284.6	0.00	124	0	6,423	1.9	1.6	0.7	0.5	0.0	4.6
RC203	70	1230.5	0.00	157	0	19,780	3.3	3.3	2.1	0.7	0.0	9.5
RC204	70	1206.6	0.00	220	0	41,386	4.4	5.4	3.3	0.9	0.0	14.1
RC205	70	1335.3	0.00	109	166	9,659	7.5	3.1	4.1	1.0	0.5	16.4
RC206	70	1285.5	0.00	118	0	12,448	2.9	1.4	1.3	0.9	0.0	6.6
RC207	70	1236.5	0.00	168	0	28,008	3.4	2.3	2.5	0.5	0.0	8.8
RC208	70	1208.2	0.00	205	0	48,135	5.3	3.9	4.4	17.2	0.0	30.9
C201	80	1182.5	0.00	189	0	295,358	2.7	0.5	21.7	1.7	0.0	27.1
C202	80	1178.4	0.00	233	0	251,585	4.9	0.5	15.8	4.4	0.0	26.0
C203	80	1172.1	0.00	263	1	432,608	4.2	1.0	25.6	3.5	0.1	35.4
C204	80	1163.1	0.00	325	0	656,380	4.3	1.9	27.2	1.2	0.0	36.0
C205	80	1170.6	0.00	237	0	259,584	3.0	0.5	14.7	1.8	0.0	20.5
C206	80	1168.9	0.00	289	0	380,825	2.1	0.8	14.9	1.2	0.0	19.8
C207	80	1167.2	0.00	262	0	389,157	3.0	1.0	17.0	1.8	0.0	23.8
C208	80	1167.2	0.00	322	0	449,968	2.9	1.1	20.5	1.0	0.0	26.6
R201	80	1201.5	0.00	164	1,401	68,398	8.9	3.3	10.7	2.2	4.1	29.5
R202	80	1121.2	0.00	210	1,864	70,136	11.7	11.1	16.7	37.4	5.0	82.9
R203	80	1034.6	0.00	251	0	121,315	11.6	11.6	10.7	8.4	0.0	43.4
R204	80	1002.1	0.00	377	1,087	332,728	33.3	62.7	42.5	24.0	2.6	171.3
R205	80	1103.6	0.00	224	5,146	118,485	13.2	6.2	28.0	14.0	30.9	92.7
R206	80	1055.4	0.00	303	6,676	136,015	30.4	33.5	44.5	39.5	33.6	184.8
R207	80	1011.3	0.00	300	0	193,472	23.7	24.2	21.3	6.1	0.0	77.7
R208	80	993.5	0.00	335	1	365,127	18.2	23.4	40.4	4.6	0.1	88.8
R209	80	1034.9	0.00	240	0	113,636	12.0	7.8	14.1	19.8	0.0	54.4
R210	80	1052.8	0.00	251	0	98,047	11.4	6.7	12.2	16.6	0.0	47.4
R211	80	999.0	0.00	319	1,688	288,485	26.8	22.3	36.0	24.5	4.4	116.4
RC201	80	1545.8	0.00	131	146	7,950	2.8	1.6	1.7	0.6	0.3	7.1
RC202	80	1458.3	0.00	169	1,780	13,712	3.6	4.5	3.2	1.1	0.9	13.3
RC203	80	1392.3	0.00	217	0	24,426	2.6	4.2	1.7	0.1	0.0	8.7
RC204	80	1366.5	0.00	270	0	39,195	3.8	6.8	2.2	0.2	0.0	13.2
RC205	80	1516.8	0.00	176	1,120	21,824	6.4	3.7	6.2	1.4	2.5	20.4
RC206	80	1455.6	0.00	172	0	20,353	4.7	2.0	2.2	1.3	0.0	10.4
RC207	80	1402.9	0.00	192	7,567	23,901	3.0	2.9	3.0	36.5	3.6	49.0
RC208	80	1364.1	0.00	286	0	36,018	4.7	4.5	2.3	1.2	0.0	12.8
C201	100	1473.3	0.00	235	0	313,754	3.3	0.5	22.9	0.9	0.0	28.3
C202	100	1464.1	0.00	358	1	717,519	10.1	2.0	22.4	3.0	0.1	39.0
C203	100	1456.3	0.00	410	0	735,149	5.5	1.7	26.5	2.8	0.0	38.7
C204	100	1448.7	0.00	568	0	957,982	8.2	2.8	41.9	2.8	0.0	59.4
C205	100	1460.2	0.00	300	0	387,733	4.5	0.9	27.9	1.9	0.0	36.2
C206	100	1455.1	0.00	321	0	334,947	3.2	0.7	17.1	1.1	0.0	22.8
C207	100	1454.5	0.00	336	1	433,450	4.6	0.8	26.4	6.0	0.0	39.2
C208	100	1451.9	0.00	323	0	659,624	5.0	1.4	20.6	2.7	0.0	31.7

Continued on next page

Table EC.3 – *Continued from previous page*

Name	n	ub	Gap%	CG	Node	Pool	t_{CCG}	t_{ENU}	t_{DX}	t_{IP1}	t_{IP2}	CPU
R201	100	1399.6	0.00	233	9,394	362,623	16.3	8.3	58.0	13.1	94.9	191.9
R202	100	1304.7	0.00	300	9,751	892,622	314.8	43.1	256.3	299.5	477.1	1396.4
R203	100	1204.8	0.00	383	9,112	463,065	43.8	25.0	77.7	51.1	138.0	338.5
R204	100	1162.2	0.00	452	424	1,241,771	63.0	41.6	10.7	34.1	5.8	162.3
R205	100	1267.3	0.00	296	11,670	490,622	24.3	13.8	105.6	107.0	184.0	436.5
R206	100	1220.9	0.00	360	85,003	541,592	34.2	29.2	169.6	275.2	1138.6	1650.4
R207	100	1182.5	0.00	401	1,410	850,676	45.3	25.0	108.6	39.2	11.7	233.2
R208	100	1157.5	0.00	537	199	1,528,311	72.7	51.4	6.6	42.6	3.4	185.2
R209	100	1205.4	0.00	337	934	276,293	33.6	16.1	44.3	29.7	6.6	132.0
R210	100	1211.8	0.00	332	1,843	526,470	44.7	25.8	81.9	159.3	18.9	333.6
R211	100	1160.6	0.00	500	1,698	1,380,290	66.9	36.9	30.2	51.9	20.7	214.0
RC201	100	1806.8	0.00	197	0	9,551	5.2	1.4	1.3	0.9	0.0	8.7
RC202	100	1680.2	0.00	266	4,738	100,395	10.6	9.5	18.0	4.5	13.8	56.8
RC203	100	1601.0	0.00	323	0	108,682	37.2	60.3	13.2	8.4	0.0	125.2
RC204	100	1574.6	0.00	450	0	235,679	24.4	13.3	15.8	1.8	0.0	55.9
RC205	100	1732.6	0.00	250	15,018	188,739	22.4	12.3	49.8	26.8	139.9	252.3
RC206	100	1698.1	0.00	258	17,173	109,935	9.4	7.3	20.1	20.1	49.9	107.2
RC207	100	1640.7	0.00	297	0	60,775	27.3	6.6	8.5	218.3	0.0	261.0
RC208	100	1570.7	0.00	456	0	208,715	24.5	9.4	19.0	2.9	0.0	56.6
C2_2_01	140	3436.2	0.00	216	762	30,507	14.5	0.2	11.9	4.4	3.7	34.9
C2_2_02	140	3380.3	0.00	354	2,550	79,763	18.0	1.0	29.6	18.0	17.6	84.4
C2_2_03	140	3304.6	0.00	547	0	103,992	15.6	0.9	14.1	15.9	0.0	46.8
C2_2_04	140	3289.5	0.00	1,001	1	160,467	29.5	1.4	21.5	3.5	0.6	57.0
C2_2_05	140	3382.9	0.00	277	2,116	39,657	10.6	0.6	12.2	7.2	5.6	36.3
C2_2_06	140	3367.4	0.00	295	5,737	114,300	14.2	0.8	28.7	12.6	66.9	123.5
C2_2_07	140	3362.5	0.00	352	1,081	86,989	19.1	0.8	19.6	17.6	6.1	63.4
C2_2_08	140	3354.6	0.00	313	8,338	214,605	19.9	1.7	48.6	39.3	153.6	263.6
C2_2_09	140	3345.1	0.00	374	8,186	398,209	22.7	3.8	93.6	60.1	268.4	449.7
C2_2_10	140	3337.4	0.00	371	11,883	227,213	17.6	2.0	54.9	54.1	142.0	271.1
R2_2_01	140	3998.9	0.00	286	51,033	193,870	17.8	1.4	32.1	19.4	321.6	392.6
R2_2_02	140	3734.7	0.00	348	3,573	84,668	32.7	3.8	17.5	4.2	17.7	76.4
R2_2_03	140	3601.9	0.00	472	105,181	1,032,610	592.7	12.8	67.4	109.1	2624.8	3409.5
R2_2_04	140	3473.3	0.00	529	1,692	56,430	46.4	4.5	13.6	3.5	4.3	72.8
R2_2_05	140	3859.3	0.00	327	16,349	278,203	29.7	2.3	48.8	7.0	162.7	251.1
R2_2_06	140	3671.9	0.00	379	29,862	552,662	46.6	7.7	81.8	39.6	486.5	663.8
R2_2_07	140	3558.3	0.00	466	167,554	882,247	579.2	11.4	159.5	227.8	2577.9	3558.4
R2_2_08	140	3468.4	0.00	603	0	37,185	40.5	2.3	3.9	5.2	0.0	52.1
R2_2_09	140	3779.6	0.00	333	35,678	379,979	30.4	3.6	74.0	25.4	624.8	759.2
R2_2_10	140	3693.5	0.00	365	57,031	974,093	619.5	7.3	157.7	44.3	896.4	1727.4
RC2_2_01	140	3718.2	0.00	400	6,546	329,296	36.0	2.4	70.0	8.4	135.8	253.3
RC2_2_02	140	3573.6	0.11	472	127,834	3,240,326	1677.1	32.8	1007.9	517.4	7556.6	10800.1
RC2_2_03	140	3487.5	0.00	681	2,513	1,319,745	782.8	20.4	125.9	151.0	207.2	1291.5
RC2_2_04	140	3449.3	0.00	913	165	203,825	166.1	6.6	58.2	18.7	4.3	254.7
RC2_2_05	140	3598.7	0.00	594	3,638	643,875	430.6	4.7	165.4	25.1	102.8	730.2
RC2_2_06	140	3622.4	0.00	549	27,445	3,127,825	1684.3	16.1	1001.9	200.2	3260.4	6169.8
RC2_2_07	140	3565.6	0.00	641	2,757	229,745	56.3	4.4	79.2	90.0	39.8	270.6
RC2_2_08	140	3539.1	0.00	739	46,385	2,285,608	1007.1	16.8	168.1	1153.8	961.5	3312.3
RC2_2_09	140	3532.4	0.00	866	5,078	1,392,640	852.3	14.8	106.1	141.9	206.5	1325.1
RC2_2_10	140	3511.4	0.00	848	1,634	756,533	518.2	9.0	165.4	83.3	41.8	819.8
C2_2_01	170	4048.3	0.00	312	6,814	162,493	31.2	1.0	64.8	52.9	125.3	275.5
C2_2_02	170	3976.9	0.00	526	0	115,713	24.5	1.2	25.2	3.3	0.0	54.5
C2_2_03	170	3932.0	0.00	772	233	233,244	41.9	2.1	37.2	16.3	4.1	102.1
C2_2_04	170	3914.0	0.00	1,231	279	441,040	53.5	4.8	69.5	26.0	2.9	157.9
C2_2_05	170	3987.2	0.00	408	6,187	289,587	36.3	3.9	91.3	42.2	132.7	307.3
C2_2_06	170	3964.3	0.00	477	0	77,616	15.0	0.7	13.7	26.4	0.0	56.0
C2_2_07	170	3958.6	0.00	493	0	91,257	14.1	1.1	14.1	15.0	0.0	44.5
C2_2_08	170	3937.0	0.00	522	0	104,173	13.1	1.0	15.4	3.5	0.0	33.2
C2_2_09	170	3931.6	0.00	570	1	149,680	36.5	1.6	20.3	4.7	0.2	63.7
C2_2_10	170	3930.7	0.00	585	0	188,221	18.5	1.9	31.0	3.4	0.0	55.4
R2_2_01	170	4631.2	0.41	529	788,455	4,847,649	1577.4	19.4	738.9	389.1	8067.3	10800.1
R2_2_02	170	4379.8	0.00	616	233,234	3,208,480	1733.4	38.1	1038.1	391.7	5897.9	9107.6
R2_2_03	170	4194.6	0.00	893	4,321	119,149	138.0	22.5	40.1	13.3	18.1	234.3
R2_2_04	170	4099.7	0.00	1,064	7,403	440,199	225.6	26.7	90.4	65.4	71.9	483.4

Continued on next page

Table EC.3 – *Continued from previous page*

Name	n	ub	Gap%	CG	Node	Pool	t_{CCG}	t_{ENU}	t_{DX}	t_{IP1}	t_{IP2}	CPU
R2_2_05	170	4476.5	0.43	603	507,632	3,936,236	1634.9	20.0	928.5	278.0	7931.7	10800.3
R2_2_06	170	4296.6	0.41	618	655,166	4,256,119	1690.9	43.1	1297.9	668.1	7089.8	10800.4
R2_2_07	170	4170.2	0.00	851	19,389	264,562	192.4	28.5	66.4	51.8	159.8	501.9
R2_2_08	170	4098.5	0.00	1,048	10,352	432,248	180.2	27.5	89.9	68.0	95.4	464.8
R2_2_09	170	4378.1	0.27	589	523,315	3,958,346	1520.3	20.5	647.2	546.5	8058.0	10800.1
R2_2_10	170	4306.9	0.00	575	33,304	929,278	633.4	9.1	155.1	55.5	853.1	1708.5
RC2_2_01	170	4404.3	0.61	564	200,302	11,590,760	3486.8	49.2	2144.8	1418.4	3677.4	10800.2
RC2_2_02	170	4231.7	0.00	607	15,079	1,315,197	840.8	11.4	80.2	221.6	561.3	1717.9
RC2_2_03	170	4160.2	0.31	768	1,802	12,984,600	3810.1	103.4	5019.5	1524.4	302.4	10800.0
RC2_2_04	170	4126.3	0.00	1,118	5,232	3,931,459	1195.1	48.6	217.6	239.4	498.6	2212.8
RC2_2_05	170	4302.0	0.00	702	8,211	4,450,275	1628.2	31.7	280.6	514.4	923.5	3388.3
RC2_2_06	170	4297.2	0.00	691	12,036	2,015,810	1123.9	13.8	158.7	230.0	665.9	2196.3
RC2_2_07	170	4242.6	0.00	716	121,402	3,939,113	1563.7	27.6	252.0	1506.0	3881.9	7240.6
RC2_2_08	170	4231.2	0.49	925	0	16,086,535	3841.4	112.7	4671.4	2140.2	0.0	10800.2
RC2_2_09	170	4236.6	1.21	985	0	113,256	56.6	—	30.7	2142.8	0.0	2232.1
RC2_2_10	170	4197.6	0.00	1,016	18,957	5,616,301	2984.1	59.5	1444.3	523.9	4487.2	9514.9
C2_2_01	200	4623.9	0.00	477	535	104,350	41.6	0.5	29.6	22.5	4.3	98.7
C2_2_02	200	4562.2	0.00	789	2,373	233,389	81.5	1.7	78.8	79.8	24.6	267.1
C2_2_03	200	4517.1	0.00	1,208	196	400,386	115.1	3.8	117.2	21.3	3.3	261.9
C2_2_04	200	4505.3	0.00	1,875	3,946	685,366	123.4	6.8	165.8	71.3	59.5	429.1
C2_2_05	200	4558.3	0.00	717	401	108,552	67.5	0.8	22.9	24.8	4.7	120.9
C2_2_06	200	4543.6	0.00	659	603	140,157	65.4	1.2	36.6	45.6	6.9	156.0
C2_2_07	200	4528.9	0.00	748	59	150,819	65.0	1.1	33.7	10.1	0.3	110.6
C2_2_08	200	4517.3	0.00	778	0	195,794	55.1	1.7	36.6	3.6	0.0	97.5
C2_2_09	200	4512.8	0.00	781	0	191,241	77.5	1.7	42.5	3.9	0.0	126.1
C2_2_10	200	4511.1	0.00	821	0	212,946	64.5	2.2	50.4	4.9	0.2	122.7
R2_2_01	200	5295.8	0.47	601	164,080	13,963,381	2944.9	57.8	2480.8	1371.5	3920.3	10800.0
R2_2_02	200	5021.4	0.00	817	32,240	4,117,501	1157.1	73.3	149.7	194.8	1196.4	2784.7
R2_2_03	200	4860.0	0.00	1,019	6,999	99,222	218.4	16.2	54.0	148.6	26.5	465.3
R2_2_04	200	4761.1	0.00	1,277	6,456	1,183,334	951.9	39.0	109.8	139.0	287.1	1532.8
R2_2_05	200	5119.5	0.33	642	439,129	8,750,436	2343.0	46.0	1392.8	1283.9	5715.8	10800.1
R2_2_06	200	4950.1	0.22	845	176,313	5,273,862	2261.0	68.8	798.2	1597.0	6061.1	10800.1
R2_2_07	200	4847.6	0.00	1,137	28,706	579,843	287.3	27.3	132.1	200.1	708.5	1358.7
R2_2_08	200	4760.4	0.00	1,319	6,894	1,381,585	1180.4	34.0	102.5	156.6	287.4	1766.7
R2_2_09	200	5034.4	0.48	700	1,530	16,597,634	3502.0	95.2	4825.6	2147.6	193.0	10800.0
R2_2_10	200	4942.0	0.00	719	60,921	1,462,597	837.4	16.9	111.6	211.5	2285.6	3466.3
RC2_2_01	200	4902.0	0.00	857	22,264	776,563	147.9	5.6	239.8	191.7	1590.1	2177.3
RC2_2_02	200	4795.5	0.00	1,004	6,421	2,704,604	1451.3	28.6	224.4	299.8	475.5	2486.4
RC2_2_03	200	4733.5	0.55	1,399	0	437,521	246.0	—	152.7	2079.0	0.0	2768.1
RC2_2_04	200	4688.9	0.51	1,629	0	769,750	281.0	—	120.9	837.4	0.0	1597.6
RC2_2_05	200	4841.8	0.00	1,167	16,341	5,145,831	2724.6	38.6	1193.2	858.8	3918.4	8746.5
RC2_2_06	200	4844.7	0.29	1,084	2,200	12,088,394	4895.5	78.1	3224.0	2131.5	438.1	10800.0
RC2_2_07	200	4790.3	0.30	1,519	0	11,550,782	5822.1	85.6	2748.3	2114.5	0.0	10800.1
RC2_2_08	200	4776.0	0.00	1,621	4,505	6,334,097	2521.6	60.2	791.1	548.7	436.9	4374.7
RC2_2_09	200	4752.0	0.00	1,707	18,849	9,692,588	4147.4	90.1	2586.1	2103.0	1263.9	10218.8
RC2_2_10	200	4739.4	0.00	1,907	4,442	1,268,032	1189.1	22.1	142.0	116.1	146.0	1619.4

Table EC.4: Detailed results for the CMTVRPTW-LT.

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
C201	70	1063.2	0.00	18.6	80	1185.7	0.00	34.1	100	1480.6	0.00	48.8
C202	70	1053.4	0.00	38.6	80	1180.2	0.00	28.1	100	1465.5	0.00	29.9
C203	70	1045.2	0.00	37.2	80	1172.9	0.00	40.0	100	1459.6	0.00	42.3
C204	70	1038.4	0.00	42.0	80	1163.1	0.00	22.4	100	1448.7	0.00	28.2
C205	70	1048.2	0.00	17.4	80	1172.8	0.00	42.2	100	1461.9	0.00	33.6
C206	70	1044.1	0.00	23.7	80	1171.1	0.00	124.4	100	1456.9	0.00	25.6
C207	70	1040.3	0.00	27.3	80	1167.2	0.00	19.1	100	1454.8	0.00	32.3
C208	70	1040.3	0.00	23.5	80	1167.2	0.00	13.3	100	1451.9	0.00	25.1
R201	70	1118.4	0.00	5.9	80	1205.6	0.00	27.8	100	1403.1	0.00	122.8
R202	70	1041.1	0.00	31.7	80	1121.2	0.00	37.2	100	1305.8	0.00	625.9
R203	70	959.5	0.00	40.4	80	1035.4	0.00	44.0	100	1206.4	0.00	477.7

Continued on next page

Table EC.4 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
R204	70	921.8	0.00	35.4	80	1002.1	0.00	117.0	100	1162.2	0.00	209.3
R205	70	1033.4	0.00	36.4	80	1105.7	0.00	115.7	100	1267.7	0.00	158.5
R206	70	985.9	0.00	52.6	80	1055.7	0.00	82.7	100	1222.9	0.00	3601.4
R207	70	942.0	0.00	31.9	80	1011.4	0.00	61.8	100	1182.5	0.00	251.6
R208	70	917.5	0.00	37.9	80	993.5	0.00	85.9	100	1157.5	0.00	178.0
R209	70	955.9	0.00	20.1	80	1038.4	0.00	130.6	100	1207.8	0.00	167.5
R210	70	983.4	0.00	57.7	80	1053.7	0.00	68.0	100	1215.8	0.00	383.1
R211	70	914.8	0.00	18.1	80	999.0	0.00	99.9	100	1164.0	0.00	901.4
RC201	70	1367.5	0.00	3.8	80	1554.1	0.00	11.3	100	1809.5	0.00	8.7
RC202	70	1284.6	0.00	4.6	80	1459.9	0.00	16.9	100	1689.2	0.00	253.2
RC203	70	1230.5	0.00	9.8	80	1392.3	0.00	7.6	100	1601.0	0.00	52.7
RC204	70	1206.6	0.00	13.8	80	1366.5	0.00	14.1	100	1574.6	0.00	58.8
RC205	70	1340.4	0.00	20.1	80	1519.8	0.00	22.6	100	1737.7	0.00	230.5
RC206	70	1290.2	0.00	7.2	80	1457.5	0.00	14.8	100	1702.5	0.00	1632.7
RC207	70	1241.1	0.00	12.5	80	1402.9	0.00	13.2	100	1641.7	0.00	50.0
RC208	70	1209.4	0.00	71.7	80	1365.6	0.00	12.4	100	1572.7	0.00	151.5
C2_2_01	140	3461.7	0.00	51.8	170	4059.8	0.00	176.6	200	4641.6	0.00	666.1
C2_2_02	140	3392.2	0.00	183.7	170	3981.5	0.00	42.6	200	4566.7	0.00	343.5
C2_2_03	140	3306.3	0.00	32.8	170	3932.0	0.00	115.0	200	4517.1	0.00	242.8
C2_2_04	140	3289.5	0.00	72.6	170	3914.3	0.00	115.9	200	4505.3	0.00	392.6
C2_2_05	140	3396.6	0.00	109.3	170	3995.6	0.00	575.0	200	4559.0	0.00	70.6
C2_2_06	140	3369.6	0.00	35.4	170	3967.2	0.00	125.8	200	4544.2	0.00	145.2
C2_2_07	140	3367.7	0.00	50.0	170	3964.1	0.00	126.4	200	4531.4	0.00	101.5
C2_2_08	140	3358.7	0.00	309.9	170	3938.9	0.00	32.9	200	4519.6	0.00	136.7
C2_2_09	140	3348.6	0.00	318.5	170	3931.6	0.00	38.1	200	4513.4	0.00	77.1
C2_2_10	140	3339.4	0.00	283.5	170	3930.9	0.00	65.4	200	4511.1	0.00	135.4
R2_2_01	140	4004.3	0.00	369.5	170	4631.6	0.00	3373.8	200	5298.1	0.45	10800.0
R2_2_02	140	3735.4	0.00	71.3	170	4387.7	0.45	10800.1	200	5021.4	0.00	2024.9
R2_2_03	140	3607.6	0.00	2693.1	170	4194.6	0.00	257.5	200	4860.0	0.00	410.6
R2_2_04	140	3473.3	0.00	78.7	170	4099.7	0.00	413.8	200	4761.1	0.00	1559.7
R2_2_05	140	3859.3	0.00	171.8	170	4476.5	0.24	10800.1	200	5122.8	0.40	10800.2
R2_2_06	140	3672.6	0.00	317.2	170	4297.4	0.22	10800.1	200	4950.1	0.23	10800.0
R2_2_07	140	3562.6	0.28	10815.6	170	4170.2	0.00	326.8	200	4847.6	0.00	1232.4
R2_2_08	140	3468.4	0.00	56.2	170	4098.5	0.00	591.8	200	4760.4	0.00	1818.0
R2_2_09	140	3780.9	0.00	406.9	170	4378.1	0.11	10800.1	200	5037.9	0.93	2364.1
R2_2_10	140	3693.5	0.00	1334.1	170	4306.9	0.00	1481.2	200	4946.0	0.00	9282.1
RC2_2_01	140	3722.8	0.00	332.2	170	4406.6	0.57	10800.1	200	4904.9	0.00	989.1
RC2_2_02	140	3575.4	0.08	10800.1	170	4231.7	0.00	1429.9	200	4796.2	0.00	2452.9
RC2_2_03	140	3487.9	0.00	1325.8	170	4158.5	0.36	10800.3	200	4733.6	0.55	2749.6
RC2_2_04	140	3449.3	0.00	192.1	170	4126.3	0.00	2250.1	200	4688.9	0.50	1264.7
RC2_2_05	140	3600.1	0.00	224.0	170	4302.9	0.00	4252.0	200	4842.1	0.00	6369.6
RC2_2_06	140	3623.4	0.00	5391.4	170	4299.2	0.00	2622.1	200	4848.6	0.67	2492.1
RC2_2_07	140	3566.2	0.00	206.3	170	4242.6	0.00	5838.0	200	4784.0	0.00	8265.5
RC2_2_08	140	3539.1	0.00	3394.3	170	4236.2	0.89	2457.4	200	4787.8	0.64	2677.6
RC2_2_09	140	3532.4	0.00	1023.0	170	4226.2	0.97	2477.1	200	4754.9	0.17	10800.0
RC2_2_10	140	3511.4	0.00	539.7	170	4200.2	0.25	10800.4	200	4739.4	0.00	5573.0

Table EC.5: Detailed results for the CMTVRPTW-R.

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
C201R0.25	70	1068.7	0.00	2.1	80	1213.4	0.00	3.6	100	1500.6	0.00	15.3
C201R0.5	70	1072.0	0.00	1.2	80	1216.1	0.00	2.1	100	1500.6	0.00	6.0
C201R0.75	70	1080.9	0.00	0.7	80	1226.8	0.00	1.3	100	1504.0	0.00	2.7
C202R0.25	70	1121.0	0.00	1.7	80	1249.5	0.00	2.8	100	1545.4	0.00	11.4
C202R0.5	70	1121.0	0.00	2.0	80	1249.5	0.00	3.0	100	1547.3	0.00	12.6
C202R0.75	70	1121.0	0.00	1.0	80	1251.7	0.00	3.6	100	1552.9	0.00	83.0
C203R0.25	70	1156.3	0.00	9.1	80	1283.0	0.00	17.6	100	1577.7	0.00	40.4
C203R0.5	70	1156.3	0.00	15.9	80	1283.0	0.00	20.0	100	1578.7	0.00	64.5
C203R0.75	70	1156.3	0.00	29.1	80	1287.1	0.00	33.7	100	1579.6	0.00	90.2
C204R0.25	70	1145.6	0.00	22.0	80	1269.0	0.00	57.7	100	1560.5	0.00	113.2
C204R0.5	70	1145.6	0.00	23.7	80	1269.0	0.00	35.2	100	1560.9	0.00	253.3

Continued on next page

Table EC.5 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
C204R0.75	70	1145.6	0.00	26.2	80	1274.4	0.00	94.3	100	1569.1	0.00	427.7
C205R0.25	70	1063.2	0.00	1.9	80	1202.3	0.00	2.7	100	1488.2	0.00	44.0
C205R0.5	70	1066.6	0.00	1.7	80	1210.1	0.00	1.8	100	1490.0	0.00	7.8
C205R0.75	70	1075.9	0.00	1.1	80	1213.6	0.00	1.8	100	1491.7	0.00	6.0
C206R0.25	70	1053.4	0.00	2.8	80	1195.6	0.00	3.2	100	1476.0	0.00	12.5
C206R0.5	70	1062.3	0.00	2.6	80	1201.3	0.00	4.1	100	1481.7	0.00	13.3
C206R0.75	70	1072.5	0.00	2.8	80	1206.6	0.00	4.5	100	1490.5	0.00	8.3
C207R0.25	70	1047.2	0.00	3.5	80	1192.3	0.00	3.6	100	1472.8	0.00	7.8
C207R0.5	70	1051.9	0.00	3.2	80	1193.9	0.00	3.7	100	1474.4	0.00	6.3
C207R0.75	70	1060.6	0.00	2.8	80	1199.9	0.00	4.1	100	1480.4	0.00	10.3
C208R0.25	70	1050.6	0.00	2.7	80	1192.7	0.00	3.6	100	1471.2	0.00	12.9
C208R0.5	70	1055.9	0.00	3.3	80	1198.3	0.00	3.9	100	1477.4	0.00	12.2
C208R0.75	70	1058.5	0.00	2.5	80	1198.3	0.00	2.7	100	1481.2	0.00	8.5
R201R0.25	70	1159.1	0.00	2.9	80	1244.7	0.00	5.9	100	1435.6	0.00	22.5
R201R0.5	70	1173.9	0.00	3.1	80	1261.8	0.00	6.5	100	1442.6	0.00	15.8
R201R0.75	70	1214.4	0.00	3.2	80	1284.3	0.00	3.1	100	1483.6	0.00	14.6
R202R0.25	70	1115.4	0.00	2.4	80	1185.2	0.00	3.5	100	1401.4	0.00	44.6
R202R0.5	70	1125.5	0.00	2.4	80	1203.4	0.00	8.7	100	1413.8	0.00	60.0
R202R0.75	70	1125.5	0.00	2.9	80	1212.6	0.00	6.3	100	1429.0	0.00	55.4
R203R0.25	70	1113.0	0.00	5.5	80	1196.1	0.00	16.4	100	1370.9	0.00	115.8
R203R0.5	70	1123.8	0.00	7.5	80	1205.1	0.00	34.1	100	1372.8	0.00	216.3
R203R0.75	70	1148.1	0.00	264.0	80	1227.5	0.00	657.8	100	1394.7	0.00	385.2
R204R0.25	70	1057.7	0.00	154.1	80	1152.7	0.00	105.7	100	1324.6	0.00	894.7
R204R0.5	70	1057.7	0.00	101.7	80	1152.7	0.00	215.7	100	1324.6	0.00	1031.4
R204R0.75	70	1079.8	0.00	114.0	80	1162.3	0.00	155.4	100	1334.6	0.00	301.4
R205R0.25	70	1073.5	0.00	6.0	80	1147.0	0.00	5.8	100	1314.4	0.00	44.3
R205R0.5	70	1083.0	0.00	4.4	80	1159.7	0.00	6.9	100	1332.3	0.00	51.6
R205R0.75	70	1084.6	0.00	2.6	80	1185.5	0.00	14.8	100	1361.8	0.00	46.5
R206R0.25	70	1039.6	0.00	3.7	80	1111.5	0.00	13.9	100	1274.8	0.00	41.6
R206R0.5	70	1059.3	0.00	16.8	80	1122.4	0.00	19.1	100	1298.1	0.00	545.4
R206R0.75	70	1070.6	0.00	9.1	80	1149.1	0.00	49.3	100	1323.5	0.00	103.0
R207R0.25	70	1049.3	0.00	8.5	80	1113.7	0.00	14.0	100	1286.7	0.00	133.4
R207R0.5	70	1056.5	0.00	14.8	80	1128.7	0.00	30.8	100	1297.3	0.00	119.4
R207R0.75	70	1056.5	0.00	11.7	80	1128.7	0.00	78.2	100	1304.7	0.00	115.9
R208R0.25	70	997.4	0.00	137.7	80	1083.2	0.00	19.4	100	1253.1	0.00	707.9
R208R0.5	70	997.4	0.00	45.0	80	1083.2	0.00	108.6	100	1253.1	0.00	785.3
R208R0.75	70	997.4	0.00	30.3	80	1086.2	0.00	79.1	100	1253.1	0.00	1790.9
R209R0.25	70	995.4	0.00	19.7	80	1079.1	0.00	33.2	100	1255.8	0.00	528.8
R209R0.5	70	997.4	0.00	14.5	80	1083.5	0.00	33.9	100	1258.8	0.00	887.6
R209R0.75	70	1033.8	0.00	8.6	80	1109.9	0.00	5.4	100	1291.6	0.00	60.6
R210R0.25	70	1026.5	0.00	7.5	80	1098.9	0.00	14.4	100	1277.3	0.00	732.1
R210R0.5	70	1032.7	0.00	7.0	80	1111.1	0.00	22.9	100	1283.7	0.00	147.0
R210R0.75	70	1094.5	0.00	5.4	80	1165.0	0.00	8.9	100	1341.5	0.00	73.1
R211R0.25	70	930.4	0.00	23.9	80	1012.3	0.00	53.1	100	1171.4	0.00	114.8
R211R0.5	70	930.4	0.00	19.4	80	1013.1	0.00	48.6	100	1175.0	0.00	138.2
R211R0.75	70	959.1	0.00	36.3	80	1039.0	0.00	56.2	100	1199.3	0.00	476.3
RC201R0.25	70	1367.5	0.00	1.3	80	1573.3	0.00	3.9	100	1839.1	0.00	19.4
RC201R0.5	70	1397.6	0.00	2.8	80	1596.6	0.00	4.0	100	1849.6	0.00	7.1
RC201R0.75	70	1434.6	0.00	3.2	80	1625.1	0.00	3.3	100	1871.2	0.00	3.2
RC202R0.25	70	1409.8	0.00	5.3	80	1558.6	0.00	3.4	100	1790.8	0.00	15.5
RC202R0.5	70	1413.9	0.00	5.1	80	1565.2	0.00	3.0	100	1813.4	0.00	15.4
RC202R0.75	70	1438.3	0.00	1.8	80	1609.3	0.00	2.7	100	1841.7	0.00	29.7
RC203R0.25	70	1397.9	0.00	4.9	80	1579.8	0.00	19.6	100	1808.2	0.00	485.7
RC203R0.5	70	1407.7	0.00	9.4	80	1606.7	0.00	16.6	100	1831.1	0.00	184.1
RC203R0.75	70	1483.9	0.00	4.7	80	1665.2	0.00	38.6	100	1880.7	0.00	637.4
RC204R0.25	70	1354.0	0.00	164.3	80	1540.4	0.00	44.2	100	1749.4	0.00	241.8
RC204R0.5	70	1354.0	0.00	36.1	80	1540.4	0.00	40.8	100	1749.4	0.00	241.1
RC204R0.75	70	1409.5	0.00	79.1	80	1567.1	0.00	16.9	100	1780.4	0.00	80.8
RC205R0.25	70	1361.5	0.00	2.4	80	1537.3	0.00	3.6	100	1760.4	0.00	20.9
RC205R0.5	70	1433.0	0.00	7.9	80	1610.2	0.00	7.1	100	1819.0	0.00	27.6
RC205R0.75	70	1474.6	0.00	3.0	80	1661.1	0.00	2.4	100	1877.8	0.00	16.5
RC206R0.25	70	1309.1	0.00	2.3	80	1500.8	0.00	3.5	100	1734.1	0.00	9.9
RC206R0.5	70	1309.9	0.00	2.0	80	1502.0	0.00	3.4	100	1746.9	0.00	16.7
RC206R0.75	70	1347.7	0.00	2.1	80	1539.0	0.00	5.8	100	1793.6	0.00	11.5
RC207R0.25	70	1281.8	0.00	4.5	80	1462.5	0.00	7.7	100	1694.4	0.00	72.3

Continued on next page

Table EC.5 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
RC207R0.5	70	1281.8	0.00	4.6	80	1462.5	0.00	7.1	100	1694.4	0.00	73.0
RC207R0.75	70	1382.5	0.00	14.3	80	1554.0	0.00	11.4	100	1780.4	0.00	23.2
RC208R0.25	70	1216.4	0.00	22.7	80	1382.9	0.00	15.1	100	1595.5	0.00	809.3
RC208R0.5	70	1216.4	0.00	15.5	80	1386.4	0.00	22.5	100	1602.8	0.34	10815.8
RC208R0.75	70	1235.3	0.00	9.9	80	1419.9	0.00	8.6	100	1620.1	0.00	71.1
C2_2_01R0.25	140	3503.9	0.00	22.2	170	4107.2	0.00	49.7	200	4687.6	0.00	102.0
C2_2_01R0.5	140	3515.3	0.00	17.2	170	4126.7	0.00	48.3	200	4702.5	0.00	126.8
C2_2_01R0.75	140	3530.4	0.00	6.0	170	4135.3	0.00	22.2	200	4738.7	0.00	240.2
C2_2_02R0.25	140	3569.5	0.00	101.4	170	4182.5	0.00	687.0	200	4787.7	0.00	1965.6
C2_2_02R0.5	140	3578.7	0.00	64.9	170	4194.6	0.00	182.6	200	4798.5	0.00	1652.3
C2_2_02R0.75	140	3604.5	0.00	57.7	170	4204.7	0.00	110.7	200	4819.0	0.00	1755.0
C2_2_03R0.25	140	3514.7	0.00	62.3	170	4147.3	0.00	293.5	200	4750.3	0.00	1716.7
C2_2_03R0.5	140	3539.0	0.00	110.8	170	4170.7	0.00	952.7	200	4773.4	0.77	535.1
C2_2_03R0.75	140	3546.4	0.00	104.7	170	4180.8	0.00	775.7	200	—	—	—
C2_2_04R0.25	140	3514.3	0.00	384.2	170	4121.7	0.00	530.5	200	4706.0	0.00	717.6
C2_2_04R0.5	140	3521.3	0.00	505.1	170	4121.7	0.41	884.0	200	—	—	—
C2_2_04R0.75	140	3547.3	0.00	572.7	170	—	—	—	200	—	—	—
C2_2_05R0.25	140	3446.8	0.00	9.7	170	4045.3	0.00	45.8	200	4631.9	0.00	172.5
C2_2_05R0.5	140	3466.7	0.00	25.8	170	4076.6	0.00	60.3	200	4645.3	0.00	76.5
C2_2_05R0.75	140	3479.3	0.00	4.0	170	4084.0	0.00	65.5	200	4669.4	0.00	278.1
C2_2_06R0.25	140	3441.5	0.00	61.8	170	4027.6	0.00	68.5	200	4610.7	0.00	135.0
C2_2_06R0.5	140	3457.8	0.00	42.0	170	4050.3	0.00	67.0	200	4631.0	0.00	136.7
C2_2_06R0.75	140	3462.8	0.00	34.2	170	4062.0	0.00	84.2	200	4653.4	0.00	1009.6
C2_2_07R0.25	140	3418.7	0.00	9.7	170	4020.5	0.00	61.1	200	4597.3	0.00	103.0
C2_2_07R0.5	140	3444.8	0.00	33.7	170	4041.0	0.00	71.1	200	4605.2	0.00	81.5
C2_2_07R0.75	140	3460.0	0.00	17.7	170	4063.6	0.00	113.4	200	4654.5	0.00	1727.6
C2_2_08R0.25	140	3413.6	0.00	35.9	170	4006.4	0.00	33.8	200	4585.3	0.00	129.3
C2_2_08R0.5	140	3431.2	0.00	33.7	170	4033.5	0.00	57.3	200	4598.8	0.00	103.8
C2_2_08R0.75	140	3448.2	0.00	9.9	170	4062.4	0.00	88.5	200	4659.9	1.13	834.7
C2_2_09R0.25	140	3381.9	0.00	12.5	170	3999.4	0.00	89.4	200	4581.0	0.00	97.0
C2_2_09R0.5	140	3387.0	0.00	10.7	170	4008.7	0.00	79.4	200	4586.6	0.00	41.0
C2_2_09R0.75	140	3416.2	0.00	49.7	170	4043.7	0.00	238.3	200	4620.8	0.00	131.7
C2_2_10R0.25	140	3381.7	0.00	14.3	170	3989.7	0.00	53.6	200	4580.2	0.00	163.8
C2_2_10R0.5	140	3384.1	0.00	12.4	170	3989.7	0.00	24.7	200	4588.7	0.00	163.2
C2_2_10R0.75	140	3411.6	0.00	9.6	170	4018.8	0.00	61.4	200	4601.8	0.00	100.6
R2_2_01R0.25	140	4066.5	0.00	117.4	170	4695.3	0.00	106.4	200	5398.6	0.00	2696.0
R2_2_01R0.5	140	4104.8	0.00	32.8	170	4771.3	0.00	835.6	200	5456.6	0.00	651.1
R2_2_01R0.75	140	4160.4	0.00	25.9	170	4811.3	0.00	43.5	200	5527.1	0.00	765.8
R2_2_02R0.25	140	4066.2	0.00	59.9	170	4674.0	0.00	63.0	200	5388.7	1.28	193.0
R2_2_02R0.5	140	4096.5	0.00	40.2	170	4729.4	0.00	702.5	200	5418.7	0.00	1734.7
R2_2_02R0.75	140	4206.7	0.00	56.4	170	4834.6	0.00	731.1	200	5514.8	0.00	980.6
R2_2_03R0.25	140	4005.5	0.00	57.2	170	4680.6	1.25	220.0	200	5346.5	1.04	426.9
R2_2_03R0.5	140	4018.8	0.00	59.7	170	4709.9	1.23	396.9	200	5358.9	0.00	759.8
R2_2_03R0.75	140	4189.8	0.00	339.2	170	4865.4	0.98	286.1	200	5544.5	1.21	505.9
R2_2_04R0.25	140	3902.4	0.00	149.2	170	4555.9	0.90	485.8	200	5171.4	0.00	1026.3
R2_2_04R0.5	140	3902.4	0.00	169.4	170	4555.9	0.79	537.7	200	5219.3	0.69	707.8
R2_2_04R0.75	140	3902.4	0.00	92.0	170	4569.5	0.67	764.8	200	—	—	—
R2_2_05R0.25	140	3919.3	0.00	18.4	170	4546.2	0.00	1058.3	200	5219.6	0.00	1469.0
R2_2_05R0.5	140	4001.4	0.00	19.4	170	4637.4	0.00	53.1	200	5302.0	0.00	989.8
R2_2_05R0.75	140	4032.8	0.00	34.8	170	4686.3	0.00	48.1	200	5344.6	0.00	702.9
R2_2_06R0.25	140	3963.8	0.00	44.9	170	4585.1	0.00	669.9	200	5266.3	0.00	1238.3
R2_2_06R0.5	140	4022.9	0.00	43.7	170	4640.4	0.00	712.7	200	5319.5	1.13	193.4
R2_2_06R0.75	140	4070.4	0.00	72.7	170	4702.1	0.00	1036.4	200	5359.4	1.13	155.8
R2_2_07R0.25	140	3932.8	0.00	61.5	170	4577.5	1.27	251.1	200	5264.0	0.92	314.5
R2_2_07R0.5	140	3951.2	0.00	46.6	170	4601.8	0.00	1554.9	200	5320.2	1.25	318.2
R2_2_07R0.75	140	4027.5	0.00	142.1	170	4666.5	0.00	1234.4	200	5388.4	1.19	390.1
R2_2_08R0.25	140	3843.9	0.00	145.0	170	4462.0	0.00	1233.5	200	5119.3	0.84	534.7
R2_2_08R0.5	140	3850.4	0.00	153.7	170	4474.2	0.00	737.8	200	—	—	—
R2_2_08R0.75	140	3953.1	0.87	532.2	170	—	—	—	200	—	—	—
R2_2_09R0.25	140	3854.4	0.00	60.1	170	4465.3	0.00	4496.8	200	5135.7	0.00	1658.9
R2_2_09R0.5	140	3920.3	0.00	25.4	170	4553.6	0.00	1091.6	200	5213.8	0.00	1043.1
R2_2_09R0.75	140	3997.3	0.00	29.4	170	4609.6	0.00	719.5	200	5277.3	0.00	970.7
R2_2_10R0.25	140	3769.1	0.00	19.4	170	4404.1	0.00	729.4	200	5066.0	0.00	1112.3
R2_2_10R0.5	140	3866.0	0.00	40.4	170	4497.5	0.00	929.2	200	5148.6	0.00	1317.0

Continued on next page

Table EC.5 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
R2_2_10R0.75	140	3916.1	0.00	56.5	170	4556.4	0.00	644.2	200	5185.7	0.00	1033.4
RC2_2_01R0.25	140	3813.1	0.00	1639.1	170	4505.0	0.00	3835.5	200	5024.8	0.00	1118.6
RC2_2_01R0.5	140	3859.7	0.00	723.0	170	4588.1	0.00	2252.5	200	5157.9	1.60	626.3
RC2_2_01R0.75	140	3994.8	0.00	708.0	170	4656.3	0.00	235.0	200	5236.7	0.00	2058.2
RC2_2_02R0.25	140	3837.3	0.00	83.2	170	4577.5	0.00	1065.5	200	5153.5	0.00	3065.1
RC2_2_02R0.5	140	3853.9	0.00	99.2	170	4626.5	0.00	1811.4	200	5187.6	0.00	3887.4
RC2_2_02R0.75	140	3867.6	0.00	85.8	170	4631.3	0.00	1027.7	200	5201.3	0.00	505.4
RC2_2_03R0.25	140	3897.9	0.00	158.9	170	4654.1	0.00	867.1	200	5209.9	1.23	766.1
RC2_2_03R0.5	140	3910.0	0.00	140.1	170	4661.9	0.00	1032.0	200	5238.2	1.13	1106.2
RC2_2_03R0.75	140	3924.4	0.00	151.0	170	4692.5	0.00	1261.9	200	5248.3	1.08	1121.0
RC2_2_04R0.25	140	—	—	—	170	4631.7	1.09	2689.7	200	—	—	—
RC2_2_04R0.5	140	—	—	—	170	4634.2	0.89	1774.4	200	—	—	—
RC2_2_04R0.75	140	—	—	—	170	4635.3	0.73	1442.7	200	—	—	—
RC2_2_05R0.25	140	3693.4	0.00	1339.8	170	4415.1	0.00	3142.2	200	4981.0	1.47	383.5
RC2_2_05R0.5	140	3739.0	0.00	1252.5	170	4499.9	1.47	227.1	200	5098.5	1.95	896.5
RC2_2_05R0.75	140	3834.7	0.00	1977.5	170	4562.9	1.30	222.3	200	5139.8	1.10	1275.4
RC2_2_06R0.25	140	3692.7	0.00	733.4	170	4411.4	0.15	10800.1	200	4965.1	1.01	688.9
RC2_2_06R0.5	140	3756.5	0.00	1098.2	170	4473.9	1.31	269.2	200	5039.6	0.96	255.9
RC2_2_06R0.75	140	3836.4	0.00	1594.8	170	4525.0	0.92	107.1	200	5149.2	1.51	431.2
RC2_2_07R0.25	140	3666.5	0.00	1223.3	170	4376.9	1.29	683.3	200	4933.7	1.14	1175.7
RC2_2_07R0.5	140	3703.4	0.00	881.1	170	4425.4	1.33	401.3	200	4987.7	1.21	299.7
RC2_2_07R0.75	140	3832.6	1.65	247.7	170	4527.6	1.65	159.8	200	5099.8	1.74	219.3
RC2_2_08R0.25	140	3609.9	0.00	1505.2	170	4323.8	1.25	1612.1	200	4940.1	2.18	2281.5
RC2_2_08R0.5	140	3646.6	0.00	1280.5	170	4376.6	1.27	1687.5	200	4934.0	0.97	334.1
RC2_2_08R0.75	140	3731.3	0.00	2373.5	170	4434.5	1.39	278.1	200	5068.1	2.34	2348.1
RC2_2_09R0.25	140	3612.3	0.00	360.6	170	4313.9	1.18	373.8	200	4857.2	0.86	572.3
RC2_2_09R0.5	140	3658.7	0.00	1235.7	170	4364.5	1.12	1339.8	200	4923.1	1.01	2368.1
RC2_2_09R0.75	140	3737.3	0.00	153.2	170	4449.2	1.56	148.7	200	5078.6	2.61	2337.6
RC2_2_10R0.25	140	3588.6	0.00	4498.3	170	4279.0	0.00	3065.3	200	4843.4	0.92	483.4
RC2_2_10R0.5	140	3588.7	0.00	789.6	170	4316.8	1.45	1441.7	200	4888.0	1.48	799.8
RC2_2_10R0.75	140	3722.7	1.77	193.9	170	4390.7	1.14	212.4	200	4967.6	0.96	300.1

EC.3. Detailed Results for the Experiments in Section 7.5

Tables EC.6 to EC.8 present the detailed results for each individual instance of the experiments in Section 7.5. Recall that we use the following notation.

- RouteOpt0: basic RouteOpt
- RouteOpt1: RouteOpt with DeLuxing
- RouteOpt2: RouteOpt with the standard (old) primal heuristic
- RouteOpt3: RouteOpt with our new primal heuristic from Section 6.3
- RouteOpt4: RouteOpt with DeLuxing and our new primal heuristic from Section 6.3

Table EC.6: Detailed results for the CVRP instances.

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
CVRP_200_001	200	569.0	27	643.3	71	271.5	15	648.0	71	643.6	71	271.8	15
CVRP_200_002	200	2111.4	71	1066.2	90	989.0	37	990.7	80	730.0	31	766.8	11
CVRP_200_003	200	9718.7	443	3596.6	410	1901.1	95	3611.9	410	2211.5	137	1955.3	35
CVRP_200_004	200	13426.1	383	7395.5	491	5478.7	89	7463.1	491	8429.8	491	6513.1	89
CVRP_200_005	200	3499.8	119	1791.7	117	1405.8	41	1812.2	117	1954.9	99	1817.2	43
CVRP_200_006	200	12145.8	335	8211.6	384	4292.7	85	8251.6	384	9257.7	384	5338.8	85
CVRP_200_007	200	2644.8	57	1165.8	20	1068.9	13	845.5	13	1481.6	13	1576.1	5
CVRP_200_008	200	4570.5	207	1883.7	202	1020.1	71	1886.9	202	1883.3	202	1031.4	69
CVRP_200_009	200	957.5	31	832.8	75	667.7	29	439.2	29	504.0	25	451.7	7
CVRP_200_010	200	5606.9	187	3214.3	237	2397.9	63	3237.8	237	1791.6	91	1577.5	29

Continued on next page

Table EC.6 – *Continued from previous page*

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
CVRP_200_011	200	479.4	27	656.3	67	398.5	13	336.3	39	656.8	67	398.8	13
CVRP_200_012	200	1275.3	59	693.3	69	630.9	25	710.2	69	540.5	41	478.0	11
CVRP_200_013	200	550.7	25	327.4	40	290.0	17	329.1	40	328.2	40	290.2	17
CVRP_200_014	200	8242.5	293	7852.0	797	3667.5	117	3252.5	209	8872.4	797	4688.1	117
CVRP_200_015	200	4718.9	103	2017.0	60	1968.8	33	1692.8	70	1977.2	39	1967.4	13
CVRP_200_016	200	1732.1	73	926.7	81	699.3	21	940.0	81	520.4	31	452.2	9
CVRP_200_017	200	3213.6	109	2290.8	142	1707.0	35	2456.6	146	1552.9	77	1451.7	27
CVRP_200_018	200	2629.3	69	1735.8	123	1124.7	29	1039.3	73	1076.7	53	902.2	17
CVRP_200_019	200	3756.9	129	1857.2	147	1337.1	47	1880.0	147	1111.4	51	1088.1	17
CVRP_200_020	200	1316.7	37	784.4	19	878.7	11	805.4	19	643.8	9	668.3	5
CVRP_200_021	200	1367.5	35	1379.9	74	1210.5	37	1417.7	74	943.4	29	823.3	7
CVRP_200_022	200	2774.2	73	1364.5	29	1441.6	19	1379.7	29	785.4	17	824.1	7
CVRP_200_023	200	11930.6	445	8299.3	472	6193.5	169	8362.4	472	5862.5	289	4735.6	105
CVRP_200_024	200	2158.6	73	2224.4	154	1589.3	45	2238.6	150	3444.6	175	2348.3	35
CVRP_200_025	200	18083.1	423	6393.5	212	6098.9	83	6422.0	212	4245.0	107	4240.9	41
CVRP_200_026	200	2332.7	73	1666.1	131	918.4	25	1689.6	129	977.3	43	737.0	13
CVRP_200_027	200	2784.5	77	1927.3	71	2081.6	39	1947.3	71	2333.3	71	2151.7	35
CVRP_200_028	200	8606.8	245	3569.7	110	3427.9	51	3594.1	110	2340.4	61	2630.1	29
CVRP_200_029	200	1683.8	49	678.9	21	616.2	9	707.0	21	755.5	17	710.1	7
CVRP_200_030	200	7266.0	333	3453.7	393	2935.7	127	3469.8	393	2217.4	193	1858.0	59
CVRP_200_031	200	1885.8	31	1689.2	31	1717.7	15	1717.7	31	770.2	13	728.5	7
CVRP_200_032	200	1000.3	41	974.5	57	820.0	21	996.5	57	594.3	23	613.9	7
CVRP_200_033	200	671.6	37	445.9	75	299.8	23	448.5	75	446.5	75	300.4	23
CVRP_200_034	200	5081.6	221	1788.3	151	1290.9	37	1805.3	151	1766.0	115	1618.9	31
CVRP_200_035	200	3266.4	117	1756.8	119	1590.5	39	1781.2	119	1133.1	49	815.5	15
CVRP_200_036	200	5178.6	177	2108.9	181	1903.4	51	2125.6	181	1287.1	81	1224.8	29
CVRP_200_037	200	1325.1	39	801.9	30	534.7	7	823.2	30	906.7	19	763.5	5
CVRP_200_038	200	1946.4	49	2391.9	127	2426.4	55	858.0	33	1110.5	37	984.2	13
CVRP_200_039	200	4502.0	139	1852.8	111	1592.6	49	1883.5	111	1338.7	39	1296.0	13
CVRP_200_040	200	14321.7	787	2124.7	417	1476.8	75	2133.6	417	1697.0	205	1534.2	47
CVRP_200_041	200	3286.2	121	1237.1	90	1097.6	35	1232.4	86	1299.4	61	1054.5	19
CVRP_200_042	200	6064.6	181	5040.8	389	3392.2	119	5056.0	389	2722.8	89	3015.7	47
CVRP_200_043	200	1691.5	49	1519.9	72	1620.1	37	1544.7	72	722.0	23	661.5	9
CVRP_200_044	200	793.8	21	932.2	46	925.3	21	960.4	46	661.5	19	613.9	5
CVRP_200_045	200	13609.7	399	7281.1	378	4185.0	89	7302.9	378	4034.4	135	4417.7	57
CVRP_200_046	200	7851.4	375	4659.8	710	2405.9	103	4718.4	710	1645.8	159	1298.2	39
CVRP_200_047	200	696.6	21	742.7	24	453.1	9	747.1	24	743.5	24	454.2	9
CVRP_200_048	200	6019.4	165	3180.3	141	2562.3	53	3205.3	141	2230.2	47	1969.7	17
CVRP_200_049	200	1353.8	53	1218.1	130	828.5	33	1237.3	130	683.7	43	550.0	13
CVRP_200_050	200	1698.0	53	1128.9	71	967.8	25	1093.3	63	793.1	33	861.2	15
CVRP_200_051	200	7797.8	199	5153.7	108	5167.3	59	5184.7	108	2357.1	37	2437.4	15
CVRP_200_052	200	1020.8	33	603.2	37	443.3	11	620.8	37	448.9	13	408.5	5
CVRP_200_053	200	8342.6	399	2305.7	277	1427.1	77	2332.8	277	1608.3	149	1257.1	37
CVRP_200_054	200	1578.1	43	1315.9	43	1298.4	15	1342.3	43	882.1	19	1045.6	5
CVRP_200_055	200	990.3	37	761.8	28	1021.8	17	1057.3	35	675.3	13	661.7	7
CVRP_200_056	200	1970.8	33	1706.1	23	1574.3	15	1744.4	23	2612.1	23	2488.4	15
CVRP_200_057	200	1540.4	51	1525.4	124	966.9	33	1546.5	124	1339.4	53	1098.1	11
CVRP_200_058	200	1384.5	41	595.5	32	546.4	19	611.6	32	437.5	11	427.5	7
CVRP_200_059	200	614.1	15	310.4	8	322.5	7	312.5	8	311.8	8	324.2	7
CVRP_200_060	200	4520.4	115	2916.1	115	3061.1	43	2943.0	115	3164.2	85	2870.7	29
CVRP_200_061	200	1654.6	61	1538.5	128	1280.8	51	1539.6	128	1537.8	128	1281.8	51
CVRP_200_062	200	2942.4	81	1173.6	42	1234.2	19	1199.1	42	1402.9	35	1399.2	13
CVRP_200_063	200	3060.8	141	1741.7	210	1685.5	79	1839.0	195	2205.7	103	2353.4	31
CVRP_200_064	200	3423.8	133	683.1	60	551.9	23	700.8	60	680.6	37	649.1	13
CVRP_200_065	200	687.0	29	401.7	66	269.8	21	403.3	66	402.6	66	270.5	21
CVRP_200_066	200	1134.2	15	964.8	19	966.9	9	996.7	19	792.7	9	759.2	3
CVRP_200_067	200	1893.1	53	964.8	36	783.4	17	984.0	36	662.9	15	604.1	5
CVRP_200_068	200	4066.9	159	5117.5	615	3674.2	165	5131.6	615	1528.8	125	1206.3	31
CVRP_200_069	200	8085.7	251	5507.2	229	4748.4	81	5530.0	229	4972.4	153	5290.5	73
CVRP_200_070	200	1776.4	69	499.2	52	369.3	21	519.9	52	573.7	21	602.9	5
CVRP_200_071	200	5011.6	243	1360.3	154	945.4	45	1373.9	154	1098.4	89	932.6	25
CVRP_200_072	200	939.3	35	817.9	41	702.2	11	680.1	33	623.1	19	548.5	3
CVRP_200_073	200	952.2	15	431.6	5	427.7	5	338.6	3	470.3	3	468.3	3
CVRP_200_074	200	242.2	11	155.1	14	153.8	5	156.4	14	155.5	14	154.1	5

Continued on next page

Table EC.6 – *Continued from previous page*

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
CVRP_200_075	200	4385.2	163	983.9	76	1275.8	53	970.7	77	985.3	76	1277.4	53
CVRP_200_076	200	2154.1	67	623.4	29	636.4	15	648.2	29	881.1	25	876.0	7
CVRP_200_077	200	696.2	21	411.6	32	450.1	19	414.3	32	412.4	32	450.5	19
CVRP_200_078	200	6030.6	265	5844.3	645	4372.4	143	5883.8	645	2623.5	161	2441.3	53
CVRP_200_079	200	2155.8	77	1584.9	142	2193.8	63	1853.2	178	1587.5	142	2196.5	63
CVRP_200_080	200	11525.2	313	5507.9	199	4493.7	73	5559.1	199	4520.1	131	4271.3	49
CVRP_200_081	200	5445.2	133	3740.8	144	3283.1	55	3782.4	144	3268.6	87	2842.0	31
CVRP_200_082	200	6350.4	137	6378.6	140	6003.5	67	6402.6	140	7501.9	140	7126.6	67
CVRP_200_083	200	13333.9	415	4954.4	261	3865.4	91	4973.4	261	5988.2	261	4899.2	91
CVRP_200_084	200	3149.1	115	3446.4	411	2869.3	121	3419.2	367	1600.0	107	1482.0	33
CVRP_200_085	200	3312.1	131	1130.0	112	831.7	35	1150.2	112	943.1	71	682.2	15
CVRP_200_086	200	2753.7	93	1082.3	90	1096.5	45	1102.0	90	1314.5	61	1242.4	27
CVRP_200_087	200	1928.4	55	668.7	30	753.1	15	693.9	30	920.4	21	917.8	9
CVRP_200_088	200	1889.3	29	3416.5	37	3147.5	29	3468.4	37	1031.0	13	998.4	3
CVRP_200_089	200	2676.6	91	1805.9	216	1654.9	117	1881.7	252	1805.7	216	1653.1	117
CVRP_200_090	200	1669.7	45	1313.4	47	1259.5	19	1337.4	47	938.1	19	890.4	9
CVRP_200_091	200	1110.3	31	895.5	56	613.9	15	903.8	56	902.1	56	614.2	15
CVRP_200_092	200	3624.3	225	1127.1	258	1151.3	85	1133.4	258	529.6	81	444.4	19
CVRP_200_093	200	1940.7	75	794.1	58	663.8	25	826.2	58	698.1	35	670.6	15
CVRP_200_094	200	2366.6	75	2027.1	145	1249.4	25	1499.8	93	885.0	33	811.4	9
CVRP_200_095	200	5718.2	167	4175.3	191	3413.3	69	4192.3	191	1594.2	51	1604.2	21
CVRP_200_096	200	28239.7	1145	5104.9	560	4309.7	197	5114.9	560	6259.0	606	5055.2	183
CVRP_200_097	200	694.0	33	304.1	28	238.3	11	305.3	28	304.4	28	239.2	11
CVRP_200_098	200	4244.0	177	2193.4	296	2779.6	85	2203.6	296	1839.8	137	1673.9	29
CVRP_200_099	200	3361.3	131	1997.8	196	1173.7	53	1419.2	115	1533.5	105	1011.2	23
CVRP_200_100	200	5926.2	275	4800.0	626	4152.6	211	4834.7	626	2029.5	141	1849.1	55
CVRP_200_101	200	410.0	23	221.2	27	206.8	17	222.1	27	221.4	27	207.3	17
CVRP_200_102	200	1528.3	57	911.8	74	663.9	27	318.5	21	367.6	21	338.0	9
CVRP_200_103	200	2119.3	49	526.5	11	500.9	7	569.1	11	854.2	9	849.4	5
CVRP_200_104	200	807.8	45	660.2	108	553.4	23	662.8	108	661.4	108	553.7	23
CVRP_200_105	200	23930.5	867	8564.9	611	4010.1	107	8612.6	611	3465.5	177	3608.3	59
CVRP_200_106	200	2604.8	77	801.6	46	766.4	19	827.2	46	757.7	21	715.7	9
CVRP_200_107	200	6516.3	215	5453.8	229	5985.0	133	5468.9	229	2826.9	101	2881.9	61
CVRP_200_108	200	3000.2	117	2560.4	241	2557.4	93	2556.7	237	1567.2	71	1467.7	23
CVRP_200_109	200	1465.1	47	1345.4	75	972.9	25	1375.4	75	1062.1	37	1064.7	11
CVRP_200_110	200	3855.5	113	1584.1	67	1504.6	27	1607.8	67	3340.5	111	2981.5	43
CVRP_200_111	200	3737.7	167	1348.3	153	970.8	43	1359.1	153	906.4	55	888.1	15
CVRP_200_112	200	9382.8	387	3643.5	319	3523.0	135	3661.8	319	1422.5	79	1324.2	33
CVRP_200_113	200	2083.5	85	1572.3	152	747.4	41	1594.5	152	846.1	43	697.2	13
CVRP_200_114	200	3034.9	125	1908.1	200	1570.1	61	2296.2	219	1135.4	47	883.4	13
CVRP_200_115	200	6839.5	223	6876.7	518	4592.4	149	7000.7	518	3721.0	179	3013.5	53
CVRP_200_116	200	1086.8	21	451.5	14	366.3	9	453.0	14	452.7	14	368.0	9
CVRP_200_117	200	4341.8	197	1700.1	217	1421.4	65	1748.2	217	2190.2	241	1895.4	81
CVRP_200_118	200	7329.8	319	4420.8	517	4130.5	157	4433.2	517	3113.0	283	2736.2	73
CVRP_200_119	200	786.0	37	617.3	80	390.1	19	620.0	80	617.5	80	391.2	19
CVRP_200_120	200	1502.2	67	502.8	68	358.6	19	505.1	68	503.5	68	358.8	19
CVRP_200_121	200	4512.1	127	2269.7	113	2211.3	47	2283.2	113	1472.5	49	1412.7	23
CVRP_200_122	200	1972.4	61	2123.0	166	1671.1	51	2046.5	144	1237.3	69	1008.2	23
CVRP_200_123	200	5628.8	193	3132.4	173	2226.4	45	3147.3	173	2107.1	97	2135.0	47
CVRP_200_124	200	1003.8	31	1697.0	108	1335.9	49	1695.8	108	712.4	27	600.6	5
CVRP_200_125	200	1913.9	39	829.8	22	804.9	15	746.3	17	543.3	13	488.5	3
CVRP_200_126	200	1484.7	33	1168.5	28	1413.0	21	1196.2	28	1221.5	9	1225.4	5
CVRP_200_127	200	24451.8	1095	5511.3	535	4039.7	199	5526.2	535	4699.0	443	3783.8	137
CVRP_200_128	200	914.4	29	1420.4	34	1201.7	15	1436.9	29	558.8	9	534.3	3
CVRP_200_129	200	714.1	23	726.8	24	513.8	9	753.1	24	557.8	9	548.6	5
CVRP_200_130	200	3847.8	159	1766.7	169	1220.3	43	1778.5	169	1114.7	73	912.6	19
CVRP_200_131	200	932.1	33	580.9	27	558.4	13	395.0	11	469.8	11	424.1	3
CVRP_200_132	200	585.3	25	414.5	56	376.9	23	413.8	56	415.0	56	378.4	23
CVRP_200_133	200	7922.2	375	6041.6	743	6104.4	237	6060.1	743	1535.7	169	1643.8	57
CVRP_200_134	200	2294.4	105	1560.1	126	890.6	31	1575.4	126	751.4	47	649.0	15
CVRP_200_135	200	40711.3	1459	10279.3	750	8965.5	297	10307.3	750	7192.4	445	6935.7	203
CVRP_200_136	200	1943.2	63	2159.1	79	2107.7	41	2195.0	79	1212.0	31	1155.5	15
CVRP_200_137	200	1822.8	65	2910.9	157	2392.0	55	2942.6	153	1044.8	37	919.7	13
CVRP_200_138	200	7149.7	285	1541.3	124	1497.0	57	1593.8	124	2912.8	153	2760.6	61

Continued on next page

Table EC.6 – *Continued from previous page*

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
CVRP_200_139	200	1866.8	91	1768.1	292	952.0	61	1765.8	278	837.8	95	719.1	31
CVRP_200_140	200	918.5	33	667.5	47	541.5	15	689.3	47	545.7	15	582.8	7
CVRP_200_141	200	1115.6	33	971.5	88	512.9	27	666.0	63	1037.2	55	866.5	15
CVRP_200_142	200	2704.5	111	1859.9	222	1026.9	53	1896.2	196	1110.6	63	888.2	15
CVRP_200_143	200	3027.0	103	1894.6	85	2001.0	41	2099.5	103	1586.6	65	1273.0	15
CVRP_200_144	200	1260.8	55	568.5	50	391.8	9	594.4	50	497.6	23	512.5	7
CVRP_200_145	200	1519.2	45	3035.6	121	2715.8	43	3075.0	121	1188.3	29	1140.4	11
CVRP_200_146	200	5518.7	179	5091.0	329	5076.3	129	5114.2	329	5270.5	342	4605.8	93
CVRP_200_147	200	3747.3	115	3288.4	145	2964.2	57	3322.6	145	2466.5	55	2275.1	23
CVRP_200_148	200	3187.4	167	1906.7	339	1636.7	89	1902.1	333	1023.2	115	938.9	23
CVRP_200_149	200	1335.4	59	646.7	78	541.5	21	648.4	78	647.5	78	541.6	21
CVRP_200_150	200	5353.0	183	1846.4	118	1891.2	43	1876.4	118	1861.1	81	2011.2	31
CVRP_200_151	200	12318.1	437	9318.9	650	9229.8	185	9335.1	650	10348.6	650	10259.7	185
CVRP_200_152	200	6127.9	263	2693.3	343	2458.1	121	2737.9	343	3456.8	289	2705.4	77
CVRP_200_153	200	1133.4	47	1469.3	131	1153.1	41	1487.6	131	1863.6	115	1582.8	37
CVRP_200_154	200	7951.0	423	4287.9	753	3268.5	175	4308.3	753	2387.5	241	2393.4	75
CVRP_200_155	200	17015.1	585	9787.6	864	6754.3	211	9816.9	864	3574.1	237	2782.2	61
CVRP_200_156	200	38462.7	1357	10694.1	1227	7978.0	349	10720.1	1227	5591.9	499	4906.3	157
CVRP_200_157	200	1519.3	37	1785.1	84	1767.7	33	1820.8	84	891.2	21	799.1	7
CVRP_200_158	200	5427.9	83	16083.8	185	14910.4	125	16117.7	185	7699.0	55	9696.7	47
CVRP_200_159	200	1411.3	37	922.6	54	1067.9	27	919.0	52	940.9	54	1067.4	27
CVRP_200_160	200	2272.9	77	1782.2	119	1517.1	43	1805.5	119	1072.8	35	939.2	11
CVRP_200_161	200	2025.8	73	1033.5	74	888.6	33	1052.5	74	1199.7	47	1147.8	19
CVRP_200_162	200	10118.9	329	3059.2	197	2187.1	65	3078.7	197	2750.4	109	2425.1	45
CVRP_200_163	200	990.0	41	1478.8	142	1154.0	55	1454.1	140	785.7	49	670.9	17
CVRP_200_164	200	740.6	23	755.1	22	744.0	11	783.1	22	638.5	13	628.0	5
CVRP_200_165	200	4038.1	101	2559.9	53	2511.7	31	2584.2	53	3309.4	53	3255.4	31
CVRP_200_166	200	2209.0	79	609.7	40	567.5	21	634.3	40	757.7	25	711.3	7
CVRP_200_167	200	3652.9	77	2303.3	54	2535.9	29	2341.7	54	2361.1	31	2457.6	13
CVRP_200_168	200	2641.5	81	2672.6	127	2482.7	69	2625.0	123	2914.0	109	2528.6	37
CVRP_200_169	200	1923.9	95	692.4	87	638.4	27	710.0	89	783.0	57	784.5	17
CVRP_200_170	200	4028.9	117	1929.1	136	2104.6	69	1950.9	136	1458.4	71	1210.3	19
CVRP_200_171	200	15557.7	517	5212.5	307	5042.1	133	5231.5	307	3654.4	185	3394.3	75
CVRP_200_172	200	1897.0	49	860.1	31	551.6	7	883.3	31	723.9	15	605.5	5
CVRP_200_173	200	1692.2	45	568.5	39	589.4	19	592.1	39	798.9	27	766.4	9
CVRP_200_174	200	473.5	11	431.9	17	344.8	5	433.9	17	432.0	17	345.7	5
CVRP_200_175	200	1161.4	25	558.7	14	611.7	9	591.1	14	825.5	9	821.8	7
CVRP_200_176	200	13714.9	459	4314.4	340	2992.7	93	4358.0	340	2078.1	89	1903.2	27
CVRP_200_177	200	1858.5	87	811.6	85	454.2	19	838.0	85	595.1	39	410.3	13
CVRP_200_178	200	979.7	27	982.5	34	874.1	11	1003.3	34	523.6	9	516.4	3
CVRP_200_179	200	726.8	33	357.1	58	336.6	13	358.1	58	357.1	58	336.5	13
CVRP_200_180	200	14313.4	375	4936.2	173	4459.9	77	4968.3	173	6011.2	173	5534.9	77
CVRP_200_181	200	668.3	39	743.2	110	714.1	29	745.7	110	745.1	110	716.0	29
CVRP_200_182	200	1092.4	53	784.0	83	576.7	21	801.2	83	773.9	69	582.3	17
CVRP_200_183	200	4350.5	129	3289.3	240	3098.2	91	3313.0	240	1878.1	101	1681.4	29
CVRP_200_184	200	3044.0	59	2221.4	57	2110.1	27	2289.3	57	1431.3	25	1393.0	11
CVRP_200_185	200	3644.9	125	1792.0	188	1365.1	57	1611.5	182	1285.3	79	1142.4	27
CVRP_200_186	200	2129.0	73	1505.0	41	1208.1	21	1488.4	43	1059.4	23	998.8	9
CVRP_200_187	200	871.1	31	1041.1	65	1365.4	33	1064.6	65	1474.6	41	1460.4	17
CVRP_200_188	200	3220.4	119	1444.8	104	1345.4	37	1534.8	108	1690.9	73	1595.2	31
CVRP_200_189	200	770.5	39	445.7	82	599.9	31	773.0	138	446.0	82	596.2	31
CVRP_200_190	200	3500.6	63	1912.6	21	1894.9	17	1945.0	21	1245.1	13	1205.1	5
CVRP_200_191	200	2991.1	181	1069.0	191	1133.8	55	1082.7	191	914.6	117	908.8	25
CVRP_200_192	200	1863.2	55	1056.8	65	1167.4	33	1076.9	65	1642.1	63	1500.5	21
CVRP_200_193	200	5931.1	191	823.5	62	785.5	19	843.7	62	1254.8	37	1217.8	11
CVRP_200_194	200	881.5	29	794.6	40	689.1	21	831.5	40	799.4	13	745.9	5
CVRP_200_195	200	4973.1	125	2854.8	76	2774.2	35	2878.5	76	3087.8	77	3940.6	31
CVRP_200_196	200	7240.3	175	1942.6	44	2093.3	27	1975.1	44	1921.0	37	1936.0	15
CVRP_200_197	200	14930.4	693	7089.0	852	4965.5	263	7132.8	852	7864.0	795	5416.7	207
CVRP_200_198	200	8746.4	281	3127.7	205	2135.2	59	3148.2	205	1694.2	77	1847.1	27
CVRP_200_199	200	554.4	23	423.8	42	281.7	7	443.6	42	348.8	19	334.2	5
CVRP_200_200	200	4891.3	125	1687.7	59	1560.2	23	1743.8	59	2103.1	63	1919.1	19
Average		4495.5	158.2	2369.9	171.5	1959.8	55.0	2342.5	166.3	1814.0	94.7	1635.5	30.4

Table EC.7: Detailed results for the VRPTW instances.

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
VRPTW_300_001	300	186.9	3	139.0	12	132.1	7	139.8	12	140.1	12	133.2	7
VRPTW_300_002	300	8436.9	213	11436.3	1002	7481.3	289	11467.9	1002	3254.6	273	2879.8	77
VRPTW_300_003	300	7237.5	245	3549.3	536	2775.9	157	3575.3	536	3004.7	335	2729.1	105
VRPTW_300_004	300	1148.0	41	303.6	34	296.1	21	304.6	34	305.3	34	297.2	21
VRPTW_300_005	300	2735.0	69	4239.3	422	3054.0	133	4317.0	422	1323.2	109	1186.9	23
VRPTW_300_006	300	407.6	13	316.8	28	198.9	9	317.3	28	318.5	28	199.6	9
VRPTW_300_007	300	10746.5	269	6770.2	566	6753.8	267	6809.8	566	3034.4	229	3240.2	83
VRPTW_300_008	300	504.4	17	345.3	32	270.2	15	346.1	32	346.4	32	271.7	15
VRPTW_300_009	300	26976.7	863	15914.8	2112	15112.9	727	15930.3	2112	14300.7	1903	10138.0	485
VRPTW_300_010	300	6237.9	207	3345.2	592	2228.2	123	3374.7	592	991.1	119	1154.9	43
VRPTW_300_011	300	1264.5	33	367.1	24	490.2	23	368.5	24	369.2	24	492.0	23
VRPTW_300_012	300	4506.8	97	2867.2	254	1774.1	87	2911.4	254	795.4	55	777.5	19
VRPTW_300_013	300	12048.0	403	6846.4	998	6957.6	401	6868.5	998	3670.4	475	3842.0	149
VRPTW_300_014	300	17855.6	569	7776.8	1186	7568.0	497	7799.1	1186	9907.5	1405	8539.0	415
VRPTW_300_015	300	10651.0	311	5796.0	592	6667.8	297	5819.6	592	5217.1	623	3901.3	131
VRPTW_300_016	300	2180.2	73	2785.7	390	1976.4	113	2800.6	390	925.6	91	844.1	25
VRPTW_300_017	300	9872.8	259	7972.7	704	4951.4	187	7993.2	704	1673.8	105	1633.2	43
VRPTW_300_018	300	2716.4	97	3576.9	471	2313.7	135	3599.7	471	963.2	107	876.8	29
VRPTW_300_019	300	2559.8	45	1680.9	90	1746.2	58	1710.5	90	1896.8	100	1838.2	63
VRPTW_300_020	300	284.1	7	147.2	11	149.3	11	148.2	11	148.3	11	150.6	11
VRPTW_300_021	300	841.8	29	958.8	80	740.4	43	961.4	80	959.7	80	743.0	43
VRPTW_300_022	300	4217.0	157	4463.5	803	3808.0	223	4476.8	803	3172.2	451	2451.5	93
VRPTW_300_023	300	961.9	39	1027.7	120	845.3	63	958.3	106	959.5	106	849.2	63
VRPTW_300_024	300	7092.4	233	2450.3	390	1854.3	95	2472.7	390	4654.6	598	3622.3	177
VRPTW_300_025	300	8112.3	209	7076.0	784	5578.8	199	7092.6	784	2006.3	179	1687.2	43
VRPTW_300_026	300	14100.8	469	9764.8	1991	6717.6	431	9795.0	1991	13691.7	2855	9929.5	523
VRPTW_300_027	300	2778.4	101	945.1	160	739.9	51	959.6	160	560.9	61	508.9	21
VRPTW_300_028	300	1861.8	51	1042.5	161	825.1	49	1055.9	161	1412.0	161	1193.5	49
VRPTW_300_029	300	6866.1	223	5914.7	776	3372.7	151	5939.5	776	4032.2	497	2668.9	93
VRPTW_300_030	300	12503.8	359	6367.3	644	3647.5	196	6412.2	644	7219.8	799	5097.8	225
VRPTW_300_031	300	184.5	3	88.2	6	99.4	3	88.6	6	88.9	6	100.3	3
VRPTW_300_032	300	12404.0	273	16555.0	1848	9455.0	333	16580.2	1848	6610.7	497	6174.8	153
VRPTW_300_033	300	3698.0	89	2405.4	164	2064.1	92	2433.9	164	1800.9	130	1697.2	57
VRPTW_300_034	300	668.3	21	858.4	70	629.3	31	857.8	72	860.1	70	632.2	31
VRPTW_300_035	300	587.9	17	942.1	112	591.3	43	944.1	112	944.1	112	594.3	43
VRPTW_300_036	300	1919.2	51	1311.1	110	1123.0	65	1334.7	110	798.5	47	829.2	15
VRPTW_300_037	300	5584.7	125	1380.8	80	1305.9	37	1403.6	80	1293.1	65	1585.8	31
VRPTW_300_038	300	403.5	5	262.9	6	293.3	6	264.8	6	264.7	6	294.8	6
VRPTW_300_039	300	19137.7	595	6226.2	930	6083.4	343	6283.4	930	5106.2	609	3951.0	119
VRPTW_300_040	300	1488.5	49	1122.7	106	1302.1	81	1126.0	106	1126.4	106	1305.0	81
VRPTW_300_041	300	6633.2	139	9232.4	1118	4327.0	202	9262.8	1118	6189.8	668	3604.6	113
VRPTW_300_042	300	592.1	13	270.7	18	457.6	13	272.6	18	272.1	18	459.3	13
VRPTW_300_043	300	3932.8	103	2092.6	150	1775.9	79	2116.9	150	2614.7	150	2299.3	79
VRPTW_300_044	300	6522.9	113	3732.4	175	2780.1	77	3789.1	175	2885.0	100	2610.1	45
VRPTW_300_045	300	881.7	19	375.1	19	375.4	17	375.8	19	376.8	19	376.4	17
VRPTW_300_046	300	1268.3	43	520.4	58	732.6	61	521.6	58	522.8	58	602.9	53
VRPTW_300_047	300	1129.7	33	917.0	154	755.3	87	931.8	154	953.5	157	759.0	29
VRPTW_300_048	300	3717.5	79	3588.2	250	3378.2	91	3622.2	250	4275.0	250	4065.7	91
VRPTW_300_049	300	2083.4	41	1293.2	98	1168.8	51	1318.3	98	946.6	39	951.3	17
VRPTW_300_050	300	4020.4	139	1710.7	204	1282.5	91	1750.6	204	2476.8	276	2108.8	111
VRPTW_300_051	300	8118.2	215	3764.2	413	3269.0	157	3829.5	413	7329.9	872	4758.8	199
VRPTW_300_052	300	5446.7	157	7576.0	1024	6161.0	339	7595.3	1024	3229.4	355	3271.8	121
VRPTW_300_053	300	8486.3	319	6113.9	820	4177.1	217	6146.9	820	6358.0	820	4421.3	217
VRPTW_300_054	300	411.6	9	192.1	16	240.0	12	193.2	16	193.1	16	241.3	12
VRPTW_300_055	300	2197.2	61	1652.1	154	1434.9	71	1156.8	119	1602.8	190	1503.4	59
VRPTW_300_056	300	671.2	21	690.0	82	804.0	73	690.6	82	693.1	82	804.3	73
VRPTW_300_057	300	3395.0	77	839.3	48	985.0	42	886.6	48	952.8	47	899.8	29
VRPTW_300_058	300	2640.7	59	1022.7	59	1009.4	41	1026.0	59	1025.1	59	1013.0	41
VRPTW_300_059	300	8255.2	179	3497.7	306	2730.2	89	3544.3	306	2743.9	151	2599.0	51
VRPTW_300_060	300	7313.8	225	6124.2	903	6132.6	335	6144.4	903	2868.3	355	2319.8	77
VRPTW_300_061	300	2145.0	43	1368.8	116	1296.3	45	1395.4	116	666.4	31	721.7	15
VRPTW_300_062	300	2263.8	59	876.7	64	849.2	54	879.8	64	879.1	64	853.8	54
VRPTW_300_063	300	1904.8	49	663.8	59	802.1	39	674.1	59	546.5	37	553.3	13

Continued on next page

Table EC.7 – *Continued from previous page*

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
VRPTW_300_064	300	200.5	7	131.3	10	139.0	7	132.0	10	132.2	10	139.7	7
VRPTW_300_065	300	3485.1	77	4392.1	340	3998.1	125	4413.3	340	4622.2	340	4228.1	125
VRPTW_300_066	300	22991.4	657	14738.8	1798	11410.6	501	14764.1	1798	3731.1	327	3696.9	137
VRPTW_300_067	300	8069.0	263	5744.2	612	4835.9	227	5756.9	612	2381.8	203	2649.6	69
VRPTW_300_068	300	21948.5	477	14738.1	1400	8551.8	289	14789.6	1400	7372.1	687	5169.2	149
VRPTW_300_069	300	2801.5	93	1693.8	172	1015.8	55	1075.0	120	617.7	45	583.4	17
VRPTW_300_070	300	535.2	25	435.7	84	308.3	39	436.6	84	437.2	84	303.6	37
VRPTW_300_071	300	3112.4	83	1453.4	166	1458.3	69	1472.4	166	789.3	61	735.7	27
VRPTW_300_072	300	11094.1	291	5971.9	758	3953.7	165	5992.5	758	1531.3	143	1509.8	45
VRPTW_300_073	300	30408.2	847	5833.1	548	4789.2	254	5894.5	548	6854.2	548	5810.4	254
VRPTW_300_074	300	1198.1	45	531.3	72	497.4	38	531.7	72	532.7	72	498.4	38
VRPTW_300_075	300	17700.9	443	5635.6	600	4872.5	166	5673.9	600	4428.2	399	3794.7	111
VRPTW_300_076	300	582.9	13	714.1	44	871.4	39	716.0	44	716.4	44	873.9	39
VRPTW_300_077	300	12559.3	337	7061.4	904	4508.3	175	7106.0	904	6099.7	711	3694.6	135
VRPTW_300_078	300	1657.5	39	821.5	66	672.6	31	846.7	66	407.1	21	356.9	5
VRPTW_300_079	300	5286.1	135	2927.2	305	2372.6	97	3013.2	305	2165.9	145	2042.7	53
VRPTW_300_080	300	1289.4	29	519.4	34	622.6	27	540.7	34	344.6	15	339.0	7
VRPTW_300_081	300	3869.3	129	1112.5	226	1367.2	115	1141.6	226	992.5	157	1059.9	61
VRPTW_300_082	300	2448.0	61	3002.6	392	2825.6	171	3018.7	392	1590.3	215	1325.0	49
VRPTW_300_083	300	4725.6	139	4927.5	710	5050.2	261	4944.3	710	3289.8	433	3217.4	135
VRPTW_300_084	300	1733.9	43	848.4	52	718.2	29	848.8	52	850.5	52	719.9	29
VRPTW_300_085	300	376.6	5	190.3	13	193.2	13	191.1	13	192.0	13	194.4	13
VRPTW_300_086	300	2971.9	87	841.7	62	854.1	60	842.8	62	843.9	62	857.4	60
VRPTW_300_087	300	3399.4	81	7849.6	582	6224.6	172	7870.1	582	1766.7	83	1941.7	37
VRPTW_300_088	300	5043.0	151	3404.3	426	3072.2	197	3420.7	426	1583.8	209	1313.3	53
VRPTW_300_089	300	5322.0	161	3502.5	520	6316.3	347	3524.1	520	3648.1	483	3803.3	199
VRPTW_300_090	300	6556.4	211	3178.9	378	2090.8	137	3194.6	378	2076.0	230	1902.5	71
VRPTW_300_091	300	2972.5	85	2150.0	240	2154.8	119	2169.9	240	2275.2	240	2280.1	119
VRPTW_300_092	300	8873.3	209	4147.4	328	3616.9	122	4196.0	328	4748.1	314	3172.1	93
VRPTW_300_093	300	13366.6	391	5678.4	470	4715.2	173	5726.4	470	5668.5	475	4156.1	129
VRPTW_300_094	300	5935.5	197	4692.7	783	3350.8	195	4743.2	783	1617.2	249	1341.2	57
VRPTW_300_095	300	5843.4	177	1551.2	216	1308.9	97	1565.1	216	1879.0	158	2082.4	85
VRPTW_300_096	300	787.6	23	379.7	32	432.0	29	380.6	32	382.3	32	434.1	29
VRPTW_300_097	300	3869.6	91	2384.4	234	2340.8	105	2489.2	234	2623.8	217	2279.8	75
VRPTW_300_098	300	9619.3	249	5147.8	498	3694.8	203	5207.3	498	5459.7	498	4006.3	203
VRPTW_300_099	300	2243.7	53	5228.3	380	3781.0	154	5248.9	380	5869.4	380	4424.7	154
VRPTW_300_100	300	4096.2	117	5061.3	919	3566.3	253	5080.2	919	4048.8	671	2626.7	157
VRPTW_300_101	300	862.1	33	649.0	82	577.8	62	650.6	82	650.5	82	580.0	62
VRPTW_300_102	300	4955.2	107	4337.8	318	2600.2	110	4366.7	318	2659.5	121	2533.8	45
VRPTW_300_103	300	509.9	19	343.1	32	450.1	23	344.4	32	344.1	32	452.3	23
VRPTW_300_104	300	14653.3	347	10431.3	1056	8164.5	305	10494.2	1056	10451.4	1092	7436.7	223
VRPTW_300_105	300	1462.6	31	1803.7	136	2045.2	102	1825.5	136	1133.7	83	1223.9	35
VRPTW_300_106	300	5939.6	131	12161.3	1468	5444.9	201	12188.3	1468	4899.8	397	4893.1	147
VRPTW_300_107	300	7972.7	179	7517.2	522	4390.1	121	7575.2	522	2936.3	181	2999.2	57
VRPTW_300_108	300	446.2	7	221.4	11	223.3	11	222.4	11	223.1	11	225.2	11
VRPTW_300_109	300	1548.6	79	814.0	166	792.4	91	929.9	190	816.7	166	653.5	83
VRPTW_300_110	300	996.5	35	474.1	60	448.8	35	475.1	60	476.5	60	450.4	35
VRPTW_300_111	300	683.9	15	482.0	34	571.6	32	483.9	34	486.2	34	576.0	32
VRPTW_300_112	300	2577.5	59	1635.5	100	1357.3	61	1654.2	100	785.0	45	855.5	17
VRPTW_300_113	300	37049.8	1207	21927.7	3552	8960.5	485	21955.1	3552	6381.2	975	3262.6	129
VRPTW_300_114	300	15154.9	339	16767.3	1602	10297.0	349	16801.4	1602	17799.3	1602	11329.1	349
VRPTW_300_115	300	3305.4	121	2215.6	338	2127.2	141	2223.7	338	782.0	133	784.4	39
VRPTW_300_116	300	11930.5	289	12665.8	1314	10037.9	379	12919.2	1314	5517.2	569	4401.8	151
VRPTW_300_117	300	9522.9	309	3754.9	552	2178.2	135	3781.9	552	2019.4	239	1592.5	67
VRPTW_300_118	300	4764.0	105	6985.9	724	4562.9	167	7025.7	724	5393.8	446	3928.6	117
VRPTW_300_119	300	3402.1	99	12784.4	1578	8726.9	417	12811.4	1578	2529.7	249	2167.0	71
VRPTW_300_120	300	2100.3	39	897.0	54	954.6	37	921.2	54	503.9	27	504.4	9
VRPTW_300_121	300	4846.8	135	5210.2	652	3904.1	207	5229.7	652	2622.4	228	2600.4	83
VRPTW_300_122	300	299.3	9	108.2	7	109.0	7	109.0	7	109.2	7	109.8	7
VRPTW_300_123	300	10348.0	291	3548.9	638	4464.8	242	3576.3	638	3586.7	527	3342.4	147
VRPTW_300_124	300	1068.7	47	813.8	122	571.1	61	821.9	122	815.9	122	573.4	61
VRPTW_300_125	300	1283.8	45	282.4	36	354.4	31	284.2	36	284.2	36	356.5	31
VRPTW_300_126	300	1515.6	47	1327.0	238	505.5	31	1339.8	238	469.5	57	477.7	19
VRPTW_300_127	300	3000.3	67	1654.6	148	1364.0	53	1675.0	148	594.6	35	582.8	17

Continued on next page

Table EC.7 – *Continued from previous page*

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
VRPTW_300_128	300	8671.5	219	7307.3	1028	6486.5	299	7371.3	1028	4880.5	581	3592.0	121
VRPTW_300_129	300	9382.0	263	4959.7	612	3468.5	175	4974.1	612	2154.0	241	2145.2	87
VRPTW_300_130	300	9409.7	355	11776.6	1596	7802.2	419	11795.0	1596	6454.9	790	4786.1	229
VRPTW_300_131	300	452.3	15	262.9	34	381.5	23	264.0	34	264.4	34	383.0	23
VRPTW_300_132	300	1238.0	33	890.4	74	1043.4	66	891.4	74	893.6	74	1045.8	66
VRPTW_300_133	300	1948.0	51	3140.8	328	1376.2	67	3164.6	328	1151.9	77	1050.2	23
VRPTW_300_134	300	10447.9	263	3914.0	524	2952.8	127	3932.7	524	1884.5	139	1730.8	45
VRPTW_300_135	300	510.9	13	307.7	24	338.3	15	309.5	24	309.5	24	340.2	15
VRPTW_300_136	300	989.3	21	645.4	42	801.7	40	646.8	42	647.3	42	805.1	40
VRPTW_300_137	300	1696.5	61	1166.4	170	1036.5	101	1178.8	174	1169.1	170	1039.7	101
VRPTW_300_138	300	9094.3	231	12254.9	1510	8896.2	371	12272.4	1510	6451.8	697	4555.7	147
VRPTW_300_139	300	6348.8	121	5633.3	526	4221.6	165	5672.3	526	2989.2	228	2735.6	97
VRPTW_300_140	300	5668.0	113	7258.5	705	3553.4	115	7286.6	705	2557.3	150	2513.0	81
VRPTW_300_141	300	840.1	35	432.9	58	316.6	31	436.7	58	434.6	58	317.6	31
VRPTW_300_142	300	536.4	15	352.6	20	405.0	20	354.2	20	355.0	20	404.8	20
VRPTW_300_143	300	7291.4	159	6266.9	712	10029.5	469	6292.8	712	7177.4	929	5854.6	251
VRPTW_300_144	300	10590.5	227	4166.6	364	3427.6	121	4254.1	364	3381.4	229	2961.2	75
VRPTW_300_145	300	7063.6	203	5852.7	948	5652.8	385	5873.7	948	6869.9	948	6670.0	385
VRPTW_300_146	300	782.1	17	444.3	22	434.3	19	446.3	22	445.9	22	438.1	19
VRPTW_300_147	300	10162.2	237	4222.5	422	4180.9	234	4234.8	422	2742.6	295	2541.6	97
VRPTW_300_148	300	4797.9	111	1934.0	142	1687.9	53	1966.8	142	906.6	39	923.1	15
VRPTW_300_149	300	8441.6	275	9265.7	1214	5618.5	327	9286.6	1214	5959.3	855	3291.2	115
VRPTW_300_150	300	4571.6	163	2099.9	238	1468.5	95	2117.5	238	887.0	69	826.6	17
VRPTW_300_151	300	2153.6	53	1562.1	124	1464.6	45	1599.3	124	490.2	21	517.6	7
VRPTW_300_152	300	1794.6	29	1419.1	44	1089.0	31	1422.5	44	1422.7	44	1092.0	31
VRPTW_300_153	300	31578.5	875	9001.6	914	8676.2	351	9077.2	914	10027.5	914	9702.1	351
VRPTW_300_154	300	6406.6	133	7174.0	682	5144.2	179	7216.3	682	7290.3	682	5260.5	179
VRPTW_300_155	300	1600.5	49	2710.2	246	2721.0	170	2729.3	246	2732.9	246	2727.8	170
VRPTW_300_156	300	4269.6	111	2148.3	172	2392.3	118	2173.6	172	2177.8	179	1875.4	61
VRPTW_300_157	300	16241.8	429	18118.7	2454	11778.1	527	18140.6	2454	19133.9	2454	12793.3	527
VRPTW_300_158	300	1747.5	41	2525.3	209	2245.1	81	2544.8	209	1054.7	59	1060.2	21
VRPTW_300_159	300	1581.6	39	903.3	46	1143.8	45	904.7	46	907.6	46	1149.6	45
VRPTW_300_160	300	10482.0	271	13769.1	1629	6388.5	303	13794.4	1629	5678.9	603	4166.9	163
VRPTW_300_161	300	14392.5	259	18041.4	1774	11132.0	459	18089.6	1774	15465.1	1464	10624.8	369
VRPTW_300_162	300	2050.6	41	455.0	20	510.0	19	455.0	20	456.7	20	512.8	19
VRPTW_300_163	300	6264.1	163	3338.8	352	4145.4	197	3364.4	352	3458.5	352	4264.9	197
VRPTW_300_164	300	16090.1	343	18435.4	2620	8365.8	353	18474.2	2620	14607.9	1924	9462.3	383
VRPTW_300_165	300	2768.4	81	802.8	78	916.1	59	811.7	78	454.9	41	446.3	15
VRPTW_300_166	300	1144.4	25	299.9	22	485.6	15	299.1	22	299.7	22	484.7	15
VRPTW_300_167	300	1616.6	47	560.6	50	692.5	39	562.2	50	561.5	50	696.3	39
VRPTW_300_168	300	3628.2	81	1518.4	139	2676.2	132	1536.9	139	1629.5	139	2787.3	132
VRPTW_300_169	300	8575.0	199	8689.4	325	8740.6	325	12121.4	529	1742.1	143	1608.1	59
VRPTW_300_170	300	2384.5	79	1640.0	102	1694.1	86	1907.3	112	1645.7	102	1700.5	86
VRPTW_300_171	300	7620.0	297	4807.7	606	2293.0	127	4827.4	606	3073.7	361	2194.6	83
VRPTW_300_172	300	10350.3	321	10568.4	1558	9436.3	473	10592.6	1558	8989.9	1242	8879.5	401
VRPTW_300_173	300	434.5	11	239.3	22	278.0	21	240.6	22	239.2	22	279.7	21
VRPTW_300_174	300	11751.3	223	7283.9	686	5223.3	233	7337.3	686	10121.0	1013	7433.7	209
VRPTW_300_175	300	9163.9	377	2065.0	408	1750.3	155	2097.9	408	3078.3	408	2763.6	155
VRPTW_300_176	300	21582.5	443	11396.1	929	8599.2	271	11421.1	929	7134.1	503	5998.7	133
VRPTW_300_177	300	5033.5	143	2813.3	242	2826.8	164	2836.9	242	3017.2	220	3858.9	216
VRPTW_300_178	300	3732.6	69	1449.8	68	1512.1	49	1479.3	68	1848.5	68	1908.8	49
VRPTW_300_179	300	1846.7	47	1862.3	100	1912.0	88	1864.6	100	1865.7	100	1919.4	88
VRPTW_300_180	300	1351.7	35	948.0	72	1198.3	64	950.3	72	953.2	72	1203.2	64
VRPTW_300_181	300	618.7	13	375.5	30	401.0	28	376.8	30	378.9	30	404.5	28
VRPTW_300_182	300	5712.0	109	3163.2	292	2383.9	111	3200.5	292	1479.7	93	1452.0	39
VRPTW_300_183	300	1960.7	51	1567.9	151	959.7	47	1591.4	151	2002.2	151	1394.1	47
VRPTW_300_184	300	4173.8	109	1282.6	96	1463.8	66	1308.1	96	1510.7	116	1189.1	43
VRPTW_300_185	300	2803.4	79	2826.3	484	2773.5	222	2837.6	484	1364.8	241	1266.8	71
VRPTW_300_186	300	2622.6	67	4827.6	534	2767.0	131	4862.3	534	1518.0	141	1066.6	37
VRPTW_300_187	300	2311.7	71	1600.9	170	1535.7	85	1635.2	170	1749.6	175	1648.8	61
VRPTW_300_188	300	4286.3	115	1881.9	172	2415.1	133	1915.2	172	1589.4	129	1646.1	43
VRPTW_300_189	300	749.0	19	801.7	42	926.0	40	802.5	42	803.5	42	928.4	40
VRPTW_300_190	300	7057.5	211	1991.2	294	1819.6	117	2002.3	294	1829.9	205	1733.1	87
VRPTW_300_191	300	1910.6	31	1431.1	74	1169.3	39	1458.5	74	1550.3	74	1288.8	39

Continued on next page

Table EC.7 – *Continued from previous page*

Name	n	VRP-Solver		RouteOpt0		RouteOpt1		RouteOpt2		RouteOpt3		RouteOpt4	
		CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node	CPU	Node
VRPTW_300_192	300	3833.3	103	6129.9	764	3925.9	149	6159.5	764	1326.7	97	1476.0	39
VRPTW_300_193	300	522.3	13	319.5	25	323.7	25	319.5	25	320.8	25	324.1	25
VRPTW_300_194	300	337.2	5	201.1	14	280.1	11	202.0	14	202.8	14	281.7	11
VRPTW_300_195	300	5407.7	157	2687.1	304	1950.6	101	2705.6	304	1321.6	115	1650.2	47
VRPTW_300_196	300	16585.4	559	11110.9	1502	7494.3	331	11203.2	1502	9085.4	989	4479.8	153
VRPTW_300_197	300	10617.8	261	1989.0	166	1997.7	74	2014.8	166	3182.0	164	3363.1	77
VRPTW_300_198	300	769.5	23	479.1	48	472.3	37	480.5	48	481.8	48	473.6	37
VRPTW_300_199	300	1098.9	31	402.0	27	415.4	23	416.8	27	422.2	30	500.2	19
VRPTW_300_200	300	1746.8	69	952.8	162	653.9	45	707.5	143	571.9	69	571.2	15
Average		5574.4	152.3	3915.1	449.9	2993.9	141.29	3949.5	450.5	2779.1	290.37	2293.8	88.25

Table EC.8: Detailed results for the long CVRP instances.

Name	n	RouteOpt0		RouteOpt1		RouteOpt3	
		CPU	Node	CPU	Node	CPU	Node
CVRP_200_18_003	200	449.7	5	509.8	3	376.4	3
CVRP_200_18_006	200	16335.6	129	16056.8	129	16414.8	129
CVRP_200_18_008	200	14153.3	111	14656.7	109	14353.0	111
CVRP_200_18_010	200	2170.9	17	2144.8	17	1917.8	17
CVRP_200_18_014	200	6088.9	35	6030.3	35	1287.8	7
CVRP_200_18_015	200	917.8	7	915.7	7	890.6	5
CVRP_200_18_023	200	10062.6	91	9732.5	89	10134.1	91
CVRP_200_18_025	200	1263.1	9	1264.2	9	734.4	3
CVRP_200_18_028	200	13669.2	109	13002.3	103	13799.1	109
CVRP_200_18_030	200	6259.7	77	6194.6	75	5656.3	77
CVRP_200_18_034	200	6475.1	67	6486.5	63	1739.2	21
CVRP_200_18_036	200	1149.4	9	1140.7	7	512.2	3
CVRP_200_18_039	200	3094.4	39	3055.0	35	3340.9	39
CVRP_200_18_040	200	321.8	3	321.2	3	345.2	3
CVRP_200_18_042	200	2379.4	21	2390.9	21	895.2	5
CVRP_200_18_046	200	120.9	1	120.7	1	121.0	1
CVRP_200_18_048	200	2170.5	15	2191.8	15	587.7	3
CVRP_200_18_051	200	6212.0	45	4576.9	37	6265.1	45
CVRP_200_18_053	200	485.2	3	485.8	3	531.6	3
CVRP_200_18_060	200	2029.7	13	2027.4	13	835.5	5
CVRP_200_18_069	200	557.4	3	551.2	3	639.6	3
CVRP_200_18_071	200	706.9	5	702.7	5	526.3	3
CVRP_200_18_078	200	8595.5	133	8804.7	131	8640.5	133
CVRP_200_18_081	200	6313.0	53	6297.4	53	1978.2	13
CVRP_200_18_082	200	13028.5	89	14134.7	87	7373.4	49
CVRP_200_18_083	200	10944.3	95	17992.3	139	11019.7	95
CVRP_200_18_092	200	2082.1	25	2068.6	23	2205.7	25
CVRP_200_18_095	200	682.7	7	680.4	7	375.5	3
CVRP_200_18_096	200	13411.5	111	12676.7	103	7660.3	61
CVRP_200_18_098	200	9071.0	131	9568.0	125	5251.3	79
CVRP_200_18_100	200	7755.5	89	7246.0	91	3199.2	37
CVRP_200_18_105	200	8965.2	105	9063.2	107	6261.1	73
CVRP_200_18_107	200	4332.9	19	4634.1	17	2184.2	7
CVRP_200_18_110	200	2190.2	11	2170.1	11	853.2	3
CVRP_200_18_111	200	2373.7	19	2365.4	19	1155.3	7
CVRP_200_18_112	200	17255.2	201	15591.3	133	17369.9	201
CVRP_200_18_115	200	4193.3	39	4221.9	33	1828.5	13
CVRP_200_18_117	200	2481.4	31	2457.7	31	3130.2	41
CVRP_200_18_118	200	8291.5	89	8464.5	87	4510.0	41
CVRP_200_18_121	200	163.2	1	163.0	1	165.3	1
CVRP_200_18_123	200	3209.1	25	3199.2	25	1589.6	11
CVRP_200_18_127	200	11175.6	103	11066.1	103	11219.7	103
CVRP_200_18_130	200	417.1	3	416.4	3	430.8	3
CVRP_200_18_133	200	12863.1	271	13094.3	227	12896.5	271
CVRP_200_18_135	200	11131.3	105	11299.0	107	8368.0	87

Continued on next page

Table EC.8 – *Continued from previous page*

Name	n	RouteOpt0		RouteOpt1		RouteOpt3	
		CPU	Node	CPU	Node	CPU	Node
CVRP_200_18_138	200	369.7	3	364.8	3	369.2	3
CVRP_200_18_146	200	2768.4	25	3307.5	27	4338.9	33
CVRP_200_18_147	200	2998.4	25	2718.9	23	1266.5	9
CVRP_200_18_150	200	5828.2	71	4639.1	41	3824.6	47
CVRP_200_18_151	200	19519.2	115	24471.3	131	14249.8	85
CVRP_200_18_152	200	9014.7	109	9006.7	107	9054.4	109
CVRP_200_18_155	200	40425.3	451	40458.1	443	27249.4	263
CVRP_200_18_162	200	208.9	1	208.5	1	212.6	1
CVRP_200_18_165	200	5377.6	41	7138.9	47	5493.8	41
CVRP_200_18_170	200	402.5	3	401.9	3	427.5	3
CVRP_200_18_171	200	1509.4	13	1497.5	13	2682.4	21
CVRP_200_18_176	200	10753.7	99	9490.5	87	6511.5	55
CVRP_200_18_180	200	21599.7	179	22084.5	183	21719.1	179
CVRP_200_18_183	200	5642.0	37	5611.3	37	1871.9	11
CVRP_200_18_185	200	23332.1	199	30633.0	261	23732.6	199
CVRP_200_18_193	200	2238.6	19	2229.2	19	1024.1	7
CVRP_200_18_195	200	6533.6	51	6507.8	51	2560.6	17
CVRP_200_18_197	200	5966.1	74	5975.0	73	7415.3	91
CVRP_200_18_198	200	4814.6	47	4881.9	47	4844.7	47
CVRP_200_18_200	200	1036.3	9	1032.9	9	466.8	3
Average		6589.8	63.7	6844.5	62.3	5244.4	50.3