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) and two important variants with loading times or release dates. 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.

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” variables[1] are 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).

[1] Columns and variables are used interchangeably in this paper in view of their correspondence.

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) 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[2] 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

[2] By default, we consider minimization problems in this paper. A maximization problem can be solved by minimizing the negation of the original objective function.

pool size, causing deterioration in overall performance. 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), its effectiveness is generally 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 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 extend the basic principle underpinning the proposed LP formulation beyond the context of variable fixing, leading to further acceleration.* Specifically, based on the principle, we prove that a large proportion of the integer variables can be relaxed to continuous ones in the final IP solved to close a BBN in some cases. Moreover, the iterative process of adding cutting planes is enhanced by performing the computation on a restricted reformulation that again only includes

variables with reduced costs no larger than half of the current gap. This enhancement incurs negligible or no sacrifice in the quality of the obtained lower bound.

- *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.

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 four 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 can be found in the e-companion. The compiled C++ library for reproducing the results and solving new instances of the CMTVRPTW and its two variants is made publicly available at `https://github.com/Yu1423/DeLuxing`.

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.

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 and Desaulniers (2005) propose to use path-reduced cost to remove arcs from the underlying network of routing and scheduling problems without sacrificing optimality. 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 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.

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.

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 (Baldacci et al. 2008, 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.

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, as non-elementary routes are not feasible. 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 $\mathrm{F}(\mathcal{R})$ 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.

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

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).

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

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.

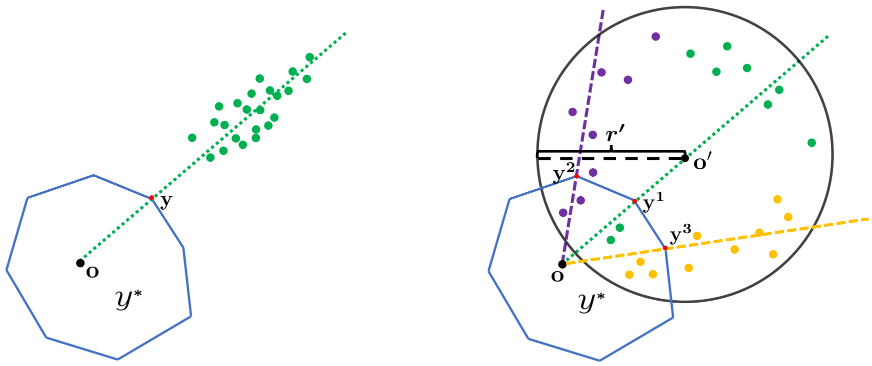

Figure 1 An example illustrating that a direct maximization of the sum of all reduced costs can be problematic, where $\mathbf{o}$ represents the origin, $\mathcal{Y}^*$ represents the feasible region of DF, $\mathbf{y}$, $\mathbf{y}^1$, $\mathbf{y}^2$, and $\mathbf{y}^3$ are extreme points of $\mathcal{Y}^*$, 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, $|\underline{\mathcal{R}}^{\boldsymbol{\pi}}|$ is much smaller than $|\underline{\mathcal{R}}|$, where $\boldsymbol{\pi}$ is a given optimal dual solution to $\overline{\mathrm{F}}(\underline{\mathcal{R}})$, $\underline{\mathcal{R}}^{\boldsymbol{\pi}} := \{r \in \underline{\mathcal{R}} : \bar{c}_r^{\boldsymbol{\pi}} \leqslant \frac{g}{2}\}$, 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. $\boldsymbol{\pi}$. 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 $\underline{\mathcal{R}}^{\boldsymbol{\pi}}$ 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 $\sum_{r\in\underline{\mathcal{R}}_k^{\boldsymbol{\pi}}} x_r \leqslant k-1$ is valid for $\mathrm{F}(\underline{\mathcal{R}})$, where $\underline{\mathcal{R}}_k^{\boldsymbol{\pi}} := \{r\in\underline{\mathcal{R}}:\ c_r^{\boldsymbol{\pi}} > \frac{g}{k}\}$.*

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

Proposition 1. *For any given $\hat{\mathbf{y}}\in\mathcal{Y}^{\boldsymbol{\pi}}$ and $j\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$, 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 $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r\leqslant 1$. Due to Lemma 1, we know $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r\leqslant 1$ is valid for $\mathrm{F}(\underline{\mathcal{R}})$. Therefore, $\mathrm{F}'(\underline{\mathcal{R}})$ can be view 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 $j\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$. 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 $\underline{\mathcal{R}}^{\boldsymbol{\pi}} = \underline{\mathcal{R}}\backslash\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$.

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

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

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

where $\bar{z}^*|_{x_j=1,\ \sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r\leqslant 1}$ is the optimal value of $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ when $x_j=1$ and $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r\leqslant 1$ 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 $j\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ can be removed for a given $\hat{\mathbf{y}}$. Note that vectors $\mathbf{f}^j$ and values $ub - c_j$ for all $j \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ need to be calculated only once and can be reused throughout the computation. Upon obtaining a new feasible dual solution $\hat{\mathbf{y}}$ to $\overline{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}})$, it suffices to compute the inner product $\langle \mathbf{f}^j, \hat{\mathbf{y}} \rangle$ and make the comparison. The following Proposition 2 provides a sufficient condition for removing a variable x_j for $j \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}$.

PROPOSITION 2. *For any given $\hat{\mathbf{y}} \in \mathcal{Y}^{\boldsymbol{\pi}}$ and $j \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}$, 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 $\mathcal{S}^j = \{r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}} : x_r$ 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 $j \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}$ in this case. Additionally, let $\overline{\mathcal{S}^j} = \underline{\mathcal{R}}_2^{\boldsymbol{\pi}} \backslash \mathcal{S}^j$. 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 $\mathcal{X} := \left\{\mathbf{x} \in \mathbb{R}_+^{|\underline{\mathcal{R}}|} : \sum_{r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r \leqslant 1, x_j = 1, x_j + x_{j'} \leqslant 1, \forall j' \in \overline{\mathcal{S}^j}\right\} = \left\{\mathbf{x} \in \mathbb{R}_+^{|\underline{\mathcal{R}}|} : \sum_{r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r \leqslant 1, x_j = 1, x_{j'} = 0, \forall j' \in \overline{\mathcal{S}^j}\right\}$, and $\mathcal{X}' := \mathcal{X} \cap \{\mathbf{x} \in \mathbb{R}_+^{|\underline{\mathcal{R}}|} : x_r = 0, \forall r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}, r \neq j\}$. 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 $z''|_{x_j=1} \leqslant z^*|_{x_j=1}$.

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

where the first inequality is again due to the fact that $\bar{c}_r^{\hat{\mathbf{y}}} \geqslant 0$ for $r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}$. 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 $\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ that can take a positive value simultaneously with x_j. For VRPs, $\mathcal{S}^j$ can be defined as the set $\{r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}} : a_{ir} + a_{ij} \leqslant 1, \forall i \in \mathcal{N}\}$, which essentially identifies routes in $\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ 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 $j' \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ 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 the $\mathrm{F}(\underline{\mathcal{R}})$ as an IP at the end. To accelerate each iteration, it is possible to skip computing $\mathcal{S}^j$ exactly and set $\mathcal{S}^j = \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ 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 $\overline{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}})$ 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 $\sum_{r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}} c_r x_r$ subject to $\sum_{r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}} a_{ir} x_r = f_i^j,\ \forall i \in \mathcal{N}$, $\sum_{r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}} a_{kr} x_r \leqslant f_k^j,\ \forall k \in \mathcal{K}$, and $x_r \geqslant 0,\ \forall r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}$. By the definition of $\mathbf{f}^j$, this means no feasible solution can be constructed using variables from $\underline{\mathcal{R}}^{\boldsymbol{\pi}}$ when $x_j = 1$, which occurs infrequently in our experiments. A plausible explanation is that all enumerated variables, particularly those in $\underline{\mathcal{R}}^{\boldsymbol{\pi}}$, 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 $\underline{\mathcal{R}}^{\boldsymbol{\pi}}$ 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 $a_{ir} \in \{0,1\}, \forall i \in \mathcal{N}, r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}$. Our dual picking thus involves the following LPs.

$$\widehat{\mathrm{DF}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J}):\quad \begin{cases} \max\ \displaystyle\sum_{j\in\mathcal{J}} \left\langle \mathbf{f}^j, \mathbf{y}\right\rangle \\ \ \text{s.t.}\ \displaystyle\sum_{i\in\mathcal{I}} a_{ir} y_i \leqslant c_r,\ \forall r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}, \\ \qquad y_i \geqslant -ub,\ \forall i \in \mathcal{I}, \\ \qquad y_i \leqslant ub,\ \forall i \in \mathcal{N}, \\ \qquad y_i \leqslant 0,\ \forall i \in \mathcal{K}. \end{cases} \quad \xrightarrow{\text{Dual}} \quad \widehat{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J}):\quad \begin{cases} \min\ \displaystyle\sum_{r\in\underline{\mathcal{R}}^{\boldsymbol{\pi}}} c_r x_r + ub\cdot\left(\sum_{i\in\mathcal{I}} w_i + \sum_{i\in\mathcal{N}} v_i\right) \\ \ \text{s.t.}\ \displaystyle\sum_{r\in\underline{\mathcal{R}}^{\boldsymbol{\pi}}} a_{ir}x_r + v_i - w_i = \sum_{j\in\mathcal{J}} f_i^j, & \forall i \in \mathcal{N}, \\ \qquad \displaystyle\sum_{r\in\underline{\mathcal{R}}^{\boldsymbol{\pi}}} a_{kr}x_r - w_i \leqslant \sum_{j\in\mathcal{J}} f_k^j, & \forall k \in \mathcal{K}, \\ \qquad x_r \geqslant 0, & \forall r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}, \\ \qquad w_i \geqslant 0,\quad \forall i \in \mathcal{I},\qquad v_i \geqslant 0,\quad \forall i \in \mathcal{N}. \end{cases}$$

We choose to work with $\widehat{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J})$ instead of $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J})$ for implementation simplicity and computational efficiency. First of all, $\widehat{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J})$ can be modified from $\overline{\mathrm{F}}(\underline{\mathcal{R}})$ more easily than $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J})$ 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 $\widehat{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J})$ 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 $\widehat{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \mathcal{J})$ 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 $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, j)$, and its dual LP, denoted by $\widehat{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, j)$, for a given $j \in \underline{\mathcal{R}}$. Let $\hat{d}^{\{j\}}$ be the optimal value of $\widehat{\mathrm{DF}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, j)$. 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 $\widehat{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}}, \{j\})$. 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 hyperparameters: 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 $\tilde{\mathcal{J}} \leftarrow \underline{\mathcal{R}}^i \backslash \mathcal{H}$.
Do
Call the subroutine with input $\tilde{\mathcal{J}}$, $\mathcal{R}_1$, $\mathcal{R}_2$, and obtain the output $\mathcal{D}$.
Set $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{D}$, $\tilde{\mathcal{J}} \leftarrow \mathcal{D} \backslash \tilde{\mathcal{J}}$, $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \backslash \mathcal{H}$, and $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \backslash \mathcal{H}$.
While $|\mathcal{D}| \geqslant \beta_2$
Set $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{D}$.
End

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

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 $\mathcal{D}$.

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 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 achieves nearly the same or sometimes even better overall performance for Algorithm 1. Here, $\hat{\mathbf{f}}^j := (\mathbf{f}^j_i)_{i\in\mathcal{N}}$ 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 $\tilde{\mathcal{J}}$ (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 $\tilde{\mathcal{J}}$ 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. 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.

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 addition 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 $r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ in $\mathrm{F}(\underline{\mathcal{R}})$ as continuous variables without compromising optimality. Let $\check{\mathrm{F}}(\underline{\mathcal{R}})$ be the formulation obtained by relaxing variables x_r for $r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ in $\mathrm{F}(\underline{\mathcal{R}})$ to be continuous and adding the constraint $\sum_{r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r \leqslant 1$. Let $\mathcal{P} \subset \mathbb{R}_+^{|\underline{\mathcal{R}}|}$ be the polyhedron corresponding to the feasible region of $\check{\mathrm{F}}(\underline{\mathcal{R}})$, $\mathcal{E} \subset \mathcal{P}$ be the set of extreme points of $\mathcal{P}$, and $\mathcal{X}^* \subset \mathcal{P}$ be the set of optimal solutions to $\check{\mathrm{F}}(\underline{\mathcal{R}})$.

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 $r \in \underline{\mathcal{R}}^{\boldsymbol{\pi}}$ 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 $\sum_{r\in\underline{\mathcal{R}}^{\boldsymbol{\pi}}} a_{ir}\bar{x}_r \in \mathbb{Z}\ \forall i \in \mathcal{N}$. Let $u_i := \sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} a_{ir}\bar{x}_r$. Consequently, $u_i = b_i - \sum_{r\in\underline{\mathcal{R}}^{\boldsymbol{\pi}}} a_{ir}\bar{x}_r$ is integral for all $i \in \mathcal{N}$. For $r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$, $\bar{x}_r$ is non-negative, thus $u_i \in \mathbb{Z}_+$. Additionally, the constraint $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r \leqslant 1$ in $\breve{\mathrm{F}}(\underline{\mathcal{R}})$ implies $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} \bar{x}_r \leqslant 1$. Let $\mathcal{Q} := \{r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}} :\ 0 < \bar{x}_r < 1\}$. 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 $\mathcal{N}^{\boldsymbol{r_1}} = \mathcal{N}^{\boldsymbol{r_2}}\ \forall \boldsymbol{r_1}, \boldsymbol{r_2} \in \mathcal{Q}$. The claim is proved by contradiction. First, $\mathcal{Q} \neq \varnothing$ and $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} \bar{x}_r \leqslant 1$ imply there does not exist $r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}}$ such that $\bar{x}_r \geqslant 1$. Thus, we have $x_r = 0\ \forall r \in \underline{\mathcal{R}}_2^{\boldsymbol{\pi}} \backslash \mathcal{Q}$. 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 $0 < x_{r'} \leqslant \sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} a_{kr}\bar{x}_r = \sum_{r\in\mathcal{Q}} a_{kr}\bar{x}_r < \sum_{r\in\mathcal{Q}} \bar{x}_r = \sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} \bar{x}_r \leqslant 1$. 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 $0 < \sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} a_{ir}\bar{x}_r = \bar{x}_r < 1$ 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 $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r \leqslant 1$ 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 (Yang 2023) are initially absent and are added dynamically using a callback function, such a scenario can happen. 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 $r \in \underline{\mathcal{R}}_3^{\boldsymbol{\pi}}$ in the original formulation $\mathrm{F}(\underline{\mathcal{R}})$ to be continuous and adding the constraints $\sum_{r\in\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}} x_r \leqslant 1$ and $\sum_{r\in\underline{\mathcal{R}}_3^{\boldsymbol{\pi}}} x_r \leqslant 2$. 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 $\mathcal{R}^f := \{r \in \underline{\mathcal{R}}_3^{\boldsymbol{\pi}} : \ \bar{x}_r > 0\}$, $\mathcal{R}^t := \{r \in \underline{\mathcal{R}} \backslash \underline{\mathcal{R}}_3^{\boldsymbol{\pi}} : \ \bar{x}_r > 0\}$, $\tilde{c} := \sum_{r\in\mathcal{R}^t} c_r \bar{x}_r$, 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 $r \in \underline{\mathcal{R}}_3^{\boldsymbol{\pi}}$ 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 $r \in \underline{\mathcal{R}}_3^{\boldsymbol{\pi}}$ 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 $\exists r \in \underline{\mathcal{R}}_3^{\boldsymbol{\pi}}$ *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 $r \in \underline{\mathcal{R}}_3^{\boldsymbol{\pi}}$ 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 $r \in \underline{\mathcal{R}}_3^{\boldsymbol{\pi}}$. Since $\sum_{r\in\mathcal{R}^f} x_r \leqslant \sum_{r\in\underline{\mathcal{R}}_3^{\boldsymbol{\pi}}} x_r \leqslant 2$ 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 (infeasible) 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 columns and cuts are added to the LP. Inspired by DeLuxing, we propose to include columns with reduced costs not exceeding half of the gap when the enumeration is successfully performed. In other words, we work with the formulation $\overline{\mathrm{F}}(\underline{\mathcal{R}}^{\boldsymbol{\pi}})$ and iteratively generate cuts to tighten it without adding any more columns from the pool.

This approach offers two advantages. First, since $\underline{\mathcal{R}}^{\boldsymbol{\pi}}$ is relatively small compared to $\underline{\mathcal{R}}$, we can avoid solving large LPs when CG adds an excessive number of columns. Second, we can directly apply Propositions 1 and 2 to remove columns whenever we obtain a new dual solution. We acknowledge that using only columns with indices in $\underline{\mathcal{R}}^{\boldsymbol{\pi}}$ may result in LP values that are not the most accurate, potentially affecting the lower bound and the cuts added. However, our numerical experiments suggest that adding cuts in this manner yields comparable bounds when we add all remaining columns from the pool back into the formulation at termination.

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. 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 does not yield satisfactory results in practice.

We propose to perform a trial enumeration with a small tentative gap and subsequently apply DeLuxing to remove unnecessary columns from the enumerated 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 four sets of experiments with a total computational time exceeding 27 days. 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, the CMTVRPTW *with loading times* (CMTVRPTW-LT; Hernandez et al. 2016) and CMTVRPTW *with release dates* (CMTVRPTW-R; Cattaruzza et al. 2016). 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 last set of experiments seeks to further demonstrate the potential of the proposed approach by solving significantly larger instances, with sizes twice as large as the largest ones currently documented in the literature.

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. (2016) 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.

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). The k-means++ method is run parallelly with all available threads. The time limit for each instance is set to 3 hours. The compiled C++ library and all the test instances are made publicly available at `https://github.com/Yu1423/DeLuxing`.

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 $\beta_1 = 500,000$ and $\beta_2 = 50$. We consider the following four variants, where the parameters 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.

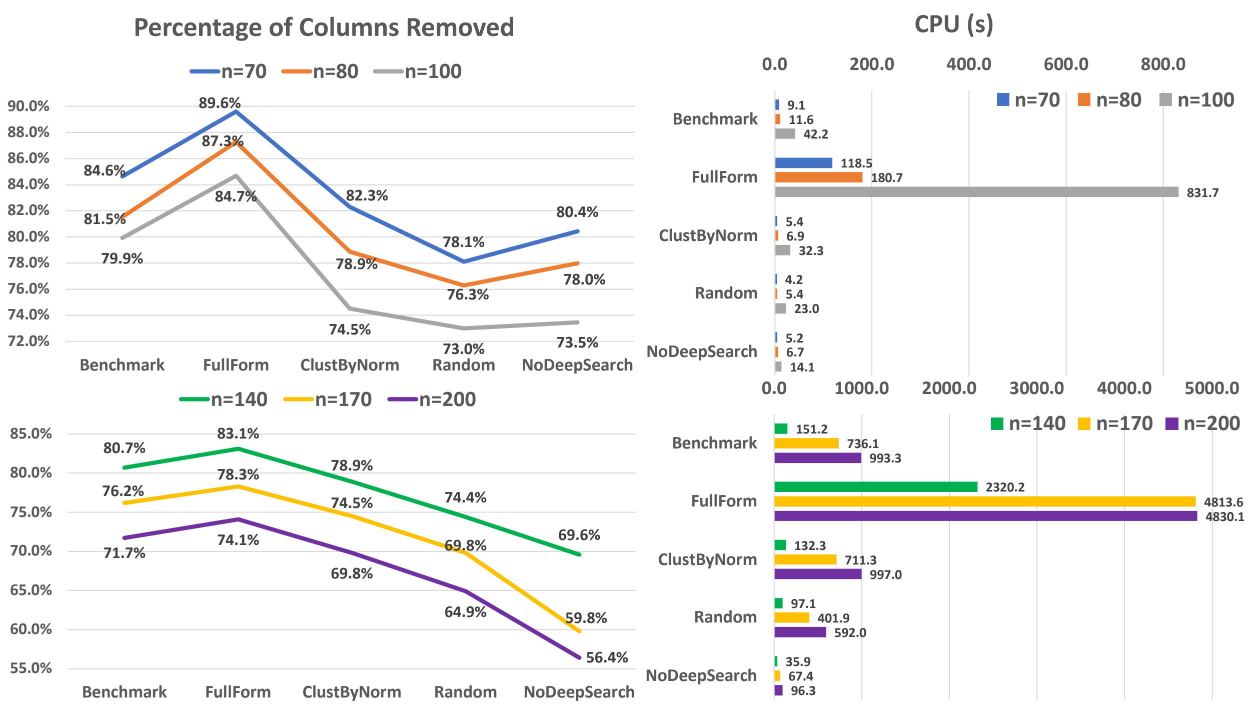

Figure 2 Comparison on the percentage of columns removed and the computing time (in seconds) for the CMTVRPTW. Each number is the average value taken over instances of the same size.

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.

7.2. 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 an implementation of the exact price-cut-and-enumerate method from Yang (2023) with DeLuxing and the three extensions incorporated. We disable each component separately on top of Default each time, and the resulting settings are denoted by NoDeluxing, NoVarRelax, OldCutAdd, and NoPrimalHeu, 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 3, Default solves significantly more instances than NoDeLuxing and NoPrimalHeu while being 32%, 17%, 62%, 53%, 16%, and 20% faster than NoDeLuxing, and 52%, 67%, 89%, 71%, 27%, and 40% faster than NoPrimalHeu, 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 inclusion 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 12% faster for 100-customer instances. While Default and OldCutAdd solve the same total number of instances, Default surpasses OldCutAdd by being 20% faster for instances of size 140.

Table 1 Summary of the results for the CMTVRPTW.

n	#Inst	Default		NoDeLuxing		NoVarRelax		OldCutAdd		NoPrimalHeu	
		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

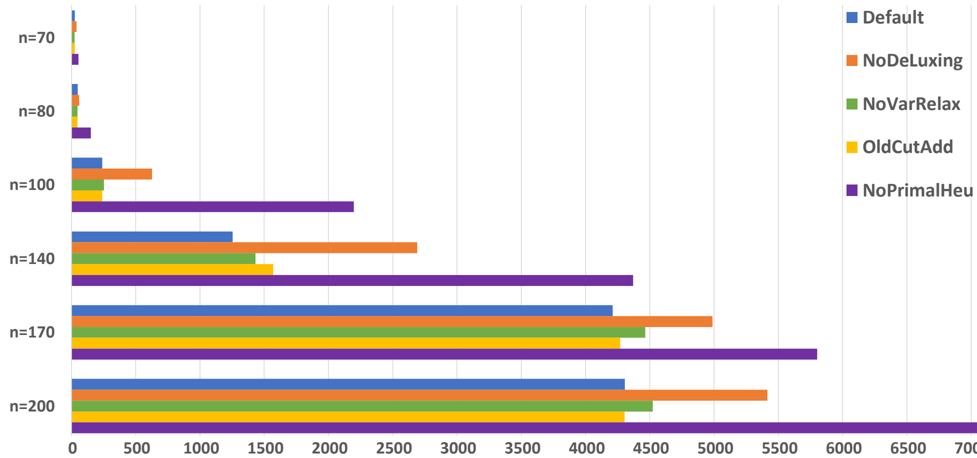

Figure 3 Comparison on CPU. Each number is the average value taken over instances of the same size.

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 et al. (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 et al. (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 4, 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.2 to EC.4 in Section EC.2 of the e-companion. The CPU values presented in Tables 2 to 4 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 et al. (2022) only reporters 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 et al. (2022) for instances of sizes 70 and 100. In contrast, Yang (2023) and Roboredo et al. (2023) can only solve 65 and 55, respectively, out of the 81 instances to optimality and both of them are more than 20 times slower than our method.

Table 2 Comparison on the CMTVRPTW.

n	#Inst	This Paper (Default)			Yang (2023)			Roboredo et al. (2023)			Zhang et al. (2022)		
		Solved	Gap%	CPU	Solved	Gap%	CPU	Solved	Gap%	CPU	Solved	Gap%	CPU
70	27	27	0.0	26.7	27	0.0	1230.3	22	—	3443.6	27	0.0	343.9
80	27	27	0.0	49.6	24	1.0	2197.5	18	—	4446.9	—	—	—
100	27	27	0.0	240.5	14	1.3	7122.5	11	—	6357.9	27	0.0	1686.4

7.3.2. Comparison on the CMTVRPTW-LT According to Table 3, 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 et al. (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.

Table 3 Comparison on the CMTVRPTW-LT.

n	#Inst	This Paper (Default)			Yang (2023)			Zhang et al. (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

7.3.3. Comparison on the CMTVRPTW-R Table 4 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 et al. (2022) and Yang (2023).

Table 4 Comparison on the CMTVRPTW-R.

n	κ	#Inst	This Paper (Default)			Yang (2023)			Zhang et al. (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 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 5 summarizes the results and more details can be found in Tables EC.2 to EC.4 in Section EC.2 of the e-companion.

Table 5 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

According to Table 5, 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 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 and proves the optimality of these two in 5 hours. 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.

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.

Currently, in the subroutine of DeLuxing (Algorithm 2), we employ the dual simplex method to solve the LP formulation $\widehat{\mathrm{F}}(\mathcal{R}_1, \tilde{\mathcal{J}})$ 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 dual simplex 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 $\mathcal{D}\backslash\tilde{\mathcal{J}}$ 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 DAMO Academy [Grant AGR00022274].

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 (2016) The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science* 50(2):676–693.

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.

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.

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 (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 (2005) *Shortest path problems with resource constraints* (Springer).

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.

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.

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.

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).

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.

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

Zhang S, et al. (2022) Solving the capacitated multi-trip vehicle routing problem with time windows. Technical report, 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 ($|\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}|$), 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	$\lvert\underline{\mathcal{R}}\rvert$	$\lvert\underline{\mathcal{R}}_2^{\boldsymbol{\pi}}\rvert$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
				$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU
C201	70	69,562	9,341	86.3	4.7	88.3	37.9	82.8	2.5	80.4	2.1	80.7	3.0
C202	70	167,231	17,318	91.2	14.4	91.7	220.9	89.3	8.3	87.1	7.2	86.9	8.3
C203	70	2,148	862	70.9	0.5	92.2	1.0	75.8	0.4	59.2	0.4	69.2	0.5
C204	70	2,384	1,203	58.4	0.8	87.6	1.2	59.1	0.6	70.9	0.4	46.3	0.7
C205	70	241,821	21,775	92.6	22.2	95.0	371.1	90.5	11.0	91.0	9.7	90.6	13.2
C206	70	198,754	20,027	91.4	19.7	92.8	278.8	86.2	9.5	87.3	8.8	87.8	10.4
C207	70	232,952	27,594	88.1	30.1	92.9	460.5	88.2	14.9	87.2	13.3	84.7	16.6
C208	70	2,590	958	77.6	0.5	92.7	1.1	77.4	0.4	64.6	0.3	77.1	0.5
R201	70	11,097	1,774	89.6	1.0	90.3	3.7	82.3	0.4	81.5	0.5	87.3	0.8
R202	70	61,021	7,130	84.4	7.6	88.9	79.9	83.2	5.0	77.6	3.2	81.2	4.3
R203	70	54,387	6,897	90.6	7.9	91.2	75.2	85.9	4.4	83.7	3.8	86.6	5.2
R204	70	108,401	12,046	89.7	17.9	90.6	309.5	86.9	13.0	82.1	7.7	84.2	9.2
R205	70	86,765	9,129	83.9	17.6	84.2	232.7	78.9	10.9	71.8	6.7	76.2	8.4
R206	70	111,271	11,024	88.2	18.5	89.5	331.4	84.5	10.8	81.4	8.8	82.3	10.2
R207	70	65,745	7,935	87.6	12.1	89.8	120.0	83.7	8.3	81.0	5.7	83.1	6.6
R208	70	113,628	12,424	89.4	14.2	90.6	256.2	87.3	9.9	84.5	7.5	84.9	7.7
R209	70	37,046	4,555	89.5	5.4	90.5	37.6	84.0	3.3	80.8	2.7	84.7	3.6
R210	70	52,710	6,757	88.8	11.6	89.9	113.2	84.3	6.8	78.8	5.3	83.8	6.6
R211	70	69,533	8,261	88.0	9.3	89.3	117.1	84.7	5.6	80.6	4.2	82.7	5.5
RC201	70	5,806	1,160	79.4	0.6	89.3	2.0	81.0	0.4	76.3	0.4	75.5	0.6
RC202	70	6,423	1,409	76.8	0.7	82.5	2.5	72.6	0.4	66.9	0.4	74.5	0.6
RC203	70	19,780	3,984	74.6	2.7	86.5	12.4	74.2	1.7	67.2	0.9	74.0	2.0
RC204	70	41,386	7,811	86.4	6.7	90.8	31.7	88.2	4.7	82.6	3.7	82.0	4.7
RC205	70	9,659	2,444	79.3	2.4	82.0	8.5	73.2	1.6	63.1	1.4	74.4	1.9
RC206	70	12,448	2,666	87.9	2.3	89.2	10.1	85.1	1.6	81.6	1.7	84.8	1.8
RC207	70	28,008	4,919	87.4	4.4	89.5	28.3	86.6	3.1	81.0	2.5	83.3	3.1
RC208	70	48,135	8,129	87.2	8.2	91.3	55.2	86.6	6.0	80.0	4.4	83.0	5.4
C201	80	295,358	24,793	91.8	26.1	94.6	392.1	92.0	15.3	90.3	11.3	87.0	15.7
C202	80	356	260	49.4	0.1	76.4	0.2	52.0	0.1	57.0	0.1	49.4	0.1
C203	80	484	316	57.0	0.1	73.8	0.2	66.9	0.1	60.3	0.1	57.0	0.1
C204	80	656,380	36,786	94.8	25.9	95.7	367.7	93.7	15.0	89.8	8.2	92.6	14.7
C205	80	264	75	87.9	0.0	88.3	0.0	87.1	0.0	85.6	0.0	87.9	0.0
C206	80	380,825	19,787	93.2	12.1	96.5	159.8	92.7	5.9	92.4	4.9	92.1	8.4
C207	80	389,157	37,695	87.5	27.6	93.8	377.5	91.3	16.6	86.4	12.8	85.6	16.6
C208	80	449,968	23,097	94.9	15.0	95.2	212.0	91.9	7.8	90.9	5.4	92.9	8.7
R201	80	68,398	7,588	93.0	9.4	93.6	127.1	89.0	5.6	87.2	4.8	88.6	6.0
R202	80	70,136	7,965	88.0	10.2	89.5	120.5	83.6	5.7	81.9	4.8	81.5	6.0
R203	80	121,315	11,818	92.4	18.5	92.7	349.7	89.7	12.2	87.5	10.0	87.6	10.4
R204	80	10,718	3,955	68.3	7.9	78.2	23.1	58.3	4.7	49.1	4.0	61.3	5.7
R205	80	118,485	11,828	86.3	23.3	86.9	386.6	82.1	13.5	79.8	11.0	79.0	11.0
R206	80	136,015	14,122	88.4	30.1	89.4	590.7	84.7	18.9	83.4	15.3	81.7	15.1
R207	80	193,472	16,674	91.7	33.7	92.3	884.9	89.4	22.2	86.3	14.8	86.4	17.9
R208	80	1,041	731	47.7	0.7	75.2	1.0	40.1	0.4	50.1	0.3	47.7	0.7

Continued on next page

Table EC.1 – *Continued from previous page*

Name	n	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{R}_2^{\pi}\rvert$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
				$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU	$\mathcal{D}\%$	CPU
R209	80	113,636	11,458	89.0	24.5	89.7	402.7	83.1	12.1	83.2	11.7	83.0	12.9
R210	80	98,047	10,691	87.7	22.4	88.6	371.6	83.5	14.6	80.3	10.2	81.0	11.1
R211	80	12,271	4,022	65.5	7.1	74.3	23.1	55.4	3.7	50.3	3.5	59.9	5.3
RC201	80	7,950	1,808	86.6	1.1	90.1	4.3	83.6	0.8	81.9	0.8	84.8	1.0
RC202	80	13,712	2,640	85.0	2.1	87.3	9.3	79.7	1.3	72.6	1.1	80.4	1.6
RC203	80	24,426	3,816	86.0	2.2	87.8	11.2	83.8	1.5	76.7	1.2	81.4	1.6
RC204	80	39,195	5,394	89.8	2.9	93.6	13.7	90.3	2.0	88.0	2.1	87.5	2.0
RC205	80	21,824	3,972	82.1	4.1	83.3	19.4	75.9	2.6	71.2	2.4	77.5	3.0
RC206	80	20,353	3,411	87.9	3.3	88.5	18.2	83.4	2.0	82.1	2.0	84.3	2.8
RC207	80	3,261	1,390	45.6	0.7	69.2	1.6	42.0	0.4	32.4	0.3	44.2	0.7
RC208	80	36,018	4,782	83.9	2.5	91.9	9.7	84.1	1.9	82.9	1.6	83.5	1.7
C201	100	313,754	28,662	95.3	33.8	95.6	694.3	95.0	23.1	93.6	18.5	92.8	21.0
C202	100	596	373	52.7	0.2	83.9	0.2	47.2	0.1	47.0	0.1	52.7	0.2
C203	100	735,149	51,877	94.6	48.8	95.2	909.4	94.6	48.8	94.3	37.3	90.2	13.5
C204	100	827	347	68.3	0.1	90.5	0.1	68.2	0.1	78.2	0.1	68.3	0.1
C205	100	243	198	77.8	0.1	77.4	0.1	75.7	0.1	71.6	0.1	77.8	0.1
C206	100	334,947	52,320	93.1	32.8	94.6	332.3	92.5	22.7	90.6	18.2	89.3	18.3
C207	100	318	191	77.4	0.0	80.8	0.0	47.5	0.0	80.5	0.0	77.4	0.0
C208	100	290	184	73.1	0.1	79.3	0.1	74.1	0.1	72.1	0.0	73.1	0.1
R201	100	362,623	25,327	89.1	59.2	89.8	1729.4	87.6	38.4	84.7	25.4	80.6	24.0
R202	100	892,622	60,239	85.1	352.4	85.8	8268.1	85.1	334.0	82.2	207.1	61.6	48.1
R203	100	43,969	11,380	55.9	37.8	61.3	159.3	44.9	21.4	32.7	13.0	46.1	19.3
R204	100	18,178	5,707	70.1	17.3	84.2	56.6	67.6	10.8	60.5	8.7	67.1	11.4
R205	100	490,622	31,381	88.1	100.1	88.1	3099.2	84.9	55.1	82.1	39.1	78.7	35.0
R206	100	541,592	42,314	81.0	124.0	80.5	2978.6	81.0	126.5	78.6	96.7	56.2	17.5
R207	100	32,326	8,203	75.5	37.2	80.0	131.9	63.6	21.0	57.6	16.7	65.6	19.3
R208	100	10,876	4,210	72.4	10.6	82.9	27.9	64.7	6.1	58.4	5.5	66.3	7.9
R209	100	15,708	4,733	66.8	12.7	72.1	39.6	49.3	6.4	42.6	4.5	59.5	7.5
R210	100	27,332	7,188	65.2	29.0	76.8	120.4	51.7	14.1	49.6	12.5	58.0	16.5
R211	100	73,958	13,997	81.7	55.9	84.4	399.7	73.8	29.0	72.8	24.5	72.7	24.2
RC201	100	9,551	1,932	76.7	1.6	79.8	5.8	70.1	1.0	65.4	0.8	73.6	1.2
RC202	100	100,395	11,242	85.2	16.5	86.1	253.2	81.1	10.1	75.9	7.4	77.9	9.0
RC203	100	108,682	13,054	92.1	20.5	92.9	332.8	89.7	13.5	87.3	9.7	87.8	12.1
RC204	100	235,679	23,142	94.6	32.0	95.3	729.2	93.2	20.5	91.9	16.6	90.9	17.5
RC205	100	188,739	19,565	83.7	50.2	84.0	1085.3	79.0	27.9	77.1	23.0	74.6	21.7
RC206	100	109,935	12,147	84.7	20.2	84.6	339.9	78.7	10.9	77.7	8.9	78.3	10.8
RC207	100	60,775	8,129	86.6	14.8	87.7	149.8	82.3	9.7	79.3	7.8	81.2	8.0
RC208	100	208,715	22,256	91.4	31.9	92.8	612.8	89.5	21.8	88.2	18.0	85.5	18.0
C2_2_01	140	30,507	5,035	77.7	6.3	85.0	25.8	78.6	4.2	70.5	2.5	73.5	4.4
C2_2_02	140	79,763	11,962	79.4	21.7	81.1	199.2	74.5	13.3	71.9	10.4	71.8	12.3
C2_2_03	140	103,992	13,880	90.1	23.7	90.7	330.4	88.2	16.7	86.2	13.4	84.9	13.9
C2_2_04	140	1,470	925	54.3	1.1	73.5	2.0	52.6	0.7	47.7	0.6	49.5	1.0
C2_2_05	140	39,657	6,303	81.5	9.1	83.1	48.0	77.8	5.7	72.6	4.6	76.3	6.0
C2_2_06	140	114,300	15,117	80.4	28.7	81.1	357.1	77.9	19.5	73.3	12.7	72.2	15.0
C2_2_07	140	86,989	11,225	83.6	17.5	83.3	174.2	81.0	11.4	76.7	7.6	76.5	9.9
C2_2_08	140	214,605	25,333	81.9	46.1	82.8	881.6	79.4	30.1	75.0	19.0	74.2	21.4
C2_2_09	140	398,209	39,695	85.8	109.2	86.1	2212.4	84.2	73.7	80.7	45.0	76.8	45.2
C2_2_10	140	227,213	26,170	81.0	50.6	81.4	815.3	78.8	33.2	73.6	19.9	71.5	24.1
R2_2_01	140	193,870	16,440	77.7	32.7	78.1	677.0	74.4	17.9	70.2	12.2	67.5	14.7
R2_2_02	140	84,668	9,409	76.4	14.8	77.8	176.6	72.9	10.1	67.3	6.0	68.6	7.7
R2_2_03	140	166,339	22,680	72.5	121.8	73.2	2090.5	67.4	78.2	63.3	57.4	61.5	48.9
R2_2_04	140	56,430	7,650	79.7	7.7	86.4	48.5	83.2	6.0	77.4	4.0	76.7	5.1
R2_2_05	140	278,203	23,707	83.6	54.1	83.6	1448.5	81.1	33.8	77.8	20.7	75.1	25.0
R2_2_06	140	552,662	38,409	82.4	55.8	83.5	2155.3	82.4	54.4	77.6	29.8	65.6	12.6
R2_2_07	140	882,247	57,874	85.4	192.3	85.9	5567.2	85.4	189.2	81.9	113.8	67.8	30.0
R2_2_08	140	37,185	5,788	77.3	5.3	85.2	24.4	75.5	3.4	73.3	2.8	73.7	3.5
R2_2_09	140	379,979	29,070	77.0	58.5	80.8	1967.2	73.7	35.7	55.6	7.2	68.4	30.3
R2_2_10	140	974,093	57,093	80.3	150.5	81.1	4200.5	80.3	157.4	78.8	111.2	56.6	20.9
RC2_2_01	140	329,296	31,979	88.1	83.8	88.8	2020.8	86.6	56.3	84.8	43.5	80.1	35.7
RC2_2_02	140	737,639	94,056	77.1	865.4	78.3	10800.3	77.1	862.1	76.6	844.7	33.7	72.3
RC2_2_03	140	234,689	34,292	86.6	239.8	88.0	3797.1	83.7	166.5	76.8	97.8	78.0	104.8
RC2_2_04	140	16,701	5,506	69.4	16.2	82.3	45.6	67.7	11.5	59.5	8.2	66.0	11.6

Continued on next page

Table EC.1 – *Continued from previous page*

Name	n	$\|\mathcal{R}\|$	$\|\mathcal{R}_2^{\pi}\|$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
				$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU
RC2_2_05	140	643,875	57,682	84.6	192.8	85.4	3946.4	84.6	197.6	82.0	140.9	65.2	40.2
RC2_2_06	140	1,073,216	102,595	85.7	1131.0	83.0	10819.2	85.7	1172.5	83.1	774.1	56.2	102.2
RC2_2_07	140	229,745	23,267	84.8	64.4	85.5	1229.7	82.6	44.5	78.3	28.2	76.8	29.3
RC2_2_08	140	349,177	51,598	86.0	405.0	85.6	5733.0	81.6	224.4	78.7	210.7	75.7	170.0
RC2_2_09	140	221,394	36,329	87.0	247.4	86.6	3315.0	84.2	162.0	79.5	107.4	78.9	109.2
RC2_2_10	140	756,533	71,592	84.0	283.3	85.9	4495.8	84.0	278.6	81.2	157.9	69.9	51.4
C2_2_01	170	162,493	19,681	80.3	56.5	82.6	742.9	76.7	35.1	73.4	20.9	72.2	26.6
C2_2_02	170	115,713	14,967	87.6	38.8	89.6	443.6	83.0	23.6	77.8	14.8	78.2	22.4
C2_2_03	170	11,130	4,316	59.1	7.4	73.0	17.1	54.1	5.0	43.4	3.8	57.4	5.6
C2_2_04	170	10,131	4,109	62.2	9.4	81.4	20.4	55.4	5.9	58.1	4.7	57.9	7.0
C2_2_05	170	289,587	27,646	86.1	72.7	87.2	1457.0	84.1	50.1	81.1	31.5	78.1	30.6
C2_2_06	170	77,616	10,088	79.3	16.4	84.4	133.8	77.4	9.6	72.8	6.4	74.0	10.4
C2_2_07	170	91,257	11,084	81.8	19.3	85.3	162.7	80.6	12.9	75.8	8.3	75.5	11.6
C2_2_08	170	104,173	12,218	89.1	20.1	90.1	195.2	82.8	11.0	81.7	10.1	84.3	14.6
C2_2_09	170	1,427	870	45.3	1.0	74.9	1.4	49.5	0.6	38.2	0.5	45.3	1.0
C2_2_10	170	547	424	32.9	0.3	68.6	0.3	29.4	0.2	26.1	0.2	32.9	0.3
R2_2_01	170	870,862	68,260	73.2	655.6	70.4	10809.9	73.2	672.1	68.0	419.2	40.0	38.3
R2_2_02	170	1,075,281	78,286	76.2	978.1	72.2	10822.1	76.2	948.5	74.7	671.7	38.2	57.7
R2_2_03	170	119,149	13,238	84.3	35.6	86.5	475.7	81.6	23.6	79.0	16.9	78.2	18.6
R2_2_04	170	440,199	34,103	88.5	85.2	89.5	2669.2	86.8	52.4	84.2	38.6	81.6	46.4
R2_2_05	170	1,108,306	80,580	72.4	874.1	69.0	10800.4	72.4	902.3	69.6	592.2	31.2	43.3
R2_2_06	170	1,458,809	99,309	77.5	1476.9	69.4	10822.1	77.5	1473.9	71.4	710.7	43.2	86.2
R2_2_07	170	264,562	22,349	87.8	63.0	88.1	1334.9	85.8	38.7	82.9	27.7	80.7	31.4
R2_2_08	170	432,248	33,744	88.4	83.8	88.5	2109.3	86.9	54.2	83.2	32.6	80.8	42.9
R2_2_09	170	922,050	70,477	74.8	605.1	74.7	10800.4	74.8	531.5	71.6	373.9	33.7	35.9
R2_2_10	170	929,278	52,301	79.8	186.6	79.7	5901.7	79.8	187.6	75.6	116.3	55.5	20.4
RC2_2_01	170	2,510,141	185,642	73.7	2958.7	61.3	10874.3	73.7	2920.6	71.2	2018.9	38.4	124.2
RC2_2_02	170	140,353	21,674	77.7	133.4	77.9	1788.9	72.3	84.6	65.5	56.0	69.0	61.1
RC2_2_03	170	2,363,139	194,616	77.4	4048.3	65.8	10853.3	77.4	4034.0	73.4	2497.0	44.7	216.8
RC2_2_04	170	262,193	36,896	78.4	299.0	79.3	4491.1	75.4	232.9	65.1	143.9	66.4	123.6
RC2_2_05	170	414,124	53,899	82.2	600.7	82.6	9933.5	79.7	451.6	74.2	288.1	71.0	228.0
RC2_2_06	170	221,690	32,635	79.4	314.9	80.2	4896.8	76.4	215.9	70.2	140.9	69.8	126.5
RC2_2_07	170	513,743	69,264	80.8	639.3	82.0	9443.9	80.8	647.7	76.5	392.4	57.2	108.3
RC2_2_08	170	2,680,336	238,054	77.8	6092.9	58.9	10960.6	77.8	6042.9	71.1	2498.2	43.3	288.8
RC2_2_09	170	113,256	17,365	74.6	48.2	82.5	645.6	74.1	36.0	64.7	16.8	67.0	24.1
RC2_2_10	170	868,556	96,623	78.8	1661.6	73.9	10800.4	78.8	1633.2	73.6	904.2	48.8	169.6
C2_2_01	200	14,760	4,986	68.0	10.1	73.6	30.5	62.4	7.2	50.3	5.7	63.2	7.2
C2_2_02	200	24,626	10,236	53.0	35.5	56.1	95.3	37.6	20.8	25.9	12.8	43.3	20.2
C2_2_03	200	7,098	3,148	50.2	8.2	74.1	15.3	52.4	5.0	49.6	4.6	51.7	6.8
C2_2_04	200	52,665	15,476	71.7	60.5	77.1	272.4	68.9	51.6	62.1	40.2	60.0	34.9
C2_2_05	200	8,922	3,471	50.0	5.9	71.5	14.4	47.5	3.2	42.1	3.1	47.6	4.6
C2_2_06	200	15,128	5,522	50.8	12.8	58.4	32.3	42.0	7.6	26.2	4.7	44.5	8.6
C2_2_07	200	1,215	793	41.5	1.1	68.0	2.0	36.9	0.6	39.6	0.6	41.5	1.1
C2_2_08	200	652	511	32.7	0.4	67.3	0.6	35.7	0.3	36.7	0.3	32.7	0.4
C2_2_09	200	191,241	22,002	85.7	55.2	86.8	962.0	84.1	39.2	79.8	30.0	78.4	29.2
C2_2_10	200	1,152	798	42.4	1.0	65.5	1.5	43.5	0.7	37.4	0.6	42.4	1.0
R2_2_01	200	2,186,524	133,332	74.7	3410.9	59.7	10816.5	74.7	3399.6	72.2	2546.0	35.4	143.5
R2_2_02	200	317,183	33,884	76.7	304.5	77.3	6982.1	74.8	212.7	70.2	141.7	65.9	120.8
R2_2_03	200	99,222	13,041	82.6	34.8	84.3	452.1	80.2	22.3	75.6	17.1	75.4	19.9
R2_2_04	200	169,281	26,527	80.9	189.0	81.7	2800.5	78.1	144.9	72.5	98.6	70.0	82.7
R2_2_05	200	1,265,511	87,735	78.1	1289.0	69.1	10816.1	78.1	1940.4	72.3	777.6	47.1	83.2
R2_2_06	200	749,779	63,666	76.0	663.6	73.6	10821.7	76.0	717.5	73.2	492.1	45.3	62.9
R2_2_07	200	579,843	42,921	84.7	108.6	85.7	3866.2	84.7	112.7	82.1	72.6	67.7	27.9
R2_2_08	200	173,780	27,619	80.6	211.3	81.5	3291.2	76.9	155.0	71.5	110.7	69.1	93.7
R2_2_09	200	4,020,994	221,799	76.7	6927.2	50.5	10943.5	76.7	6925.9	74.2	4327.7	37.7	210.2
R2_2_10	200	267,272	29,237	76.5	210.6	76.0	3896.5	73.5	144.8	66.9	89.3	62.6	78.2
RC2_2_01	200	776,563	63,300	88.0	312.9	87.7	6561.9	88.0	310.6	86.5	218.8	70.5	54.7
RC2_2_02	200	240,386	34,388	76.0	392.5	77.5	6167.9	71.0	281.9	67.3	191.9	65.6	155.9
RC2_2_03	200	437,521	43,994	81.1	241.1	82.8	4628.9	78.4	178.8	75.5	124.9	70.2	129.7
RC2_2_04	200	769,750	67,584	86.7	329.0	87.3	5942.8	86.7	316.2	81.3	187.0	63.6	60.0
RC2_2_05	200	1,016,354	100,181	78.4	1226.4	75.9	10808.9	78.4	1237.5	74.1	695.3	44.2	155.8
RC2_2_06	200	1,926,998	174,312	80.9	4142.4	64.6	10865.5	80.9	4138.1	78.3	2633.6	42.9	342.3

Continued on next page

Table EC.1 – *Continued from previous page*

Name	n	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{R}_2^\pi\rvert$	Benchmark		FullForm		ClustByNorm		Random		NoDeepSearch	
				$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU	$\mathcal{D}$%	CPU
RC2_2_07	200	1,654,753	166,848	80.8	5216.5	68.9	10930.0	80.8	5196.5	75.0	2385.8	57.4	447.8
RC2_2_08	200	698,876	77,658	84.3	1019.0	82.1	10802.7	84.3	971.8	78.5	523.1	65.7	135.6
RC2_2_09	200	1,839,184	161,452	83.2	3234.2	75.8	10820.1	83.2	3237.4	80.9	1943.5	58.3	296.8
RC2_2_10	200	98,714	18,102	79.7	144.7	82.1	1262.9	78.3	128.7	69.7	81.7	72.0	74.6

EC.2. Detailed Results for the Last Two Sets of Experiments

Tables EC.2 to EC.4 present the detailed results for each individual instance of the last 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), the optimality gap $\frac{ub-lb}{ub} \times 100$ at termination (Gap), and the computational time in seconds (CPU). 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.2: Detailed results for the CMTVRPTW.

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
C201	70	1052.2	0.00	4.7	80	1182.5	0.00	27.1	100	1473.3	0.00	28.3
C202	70	1047.7	0.00	19.4	80	1178.4	0.00	26.0	100	1464.1	0.00	39.0
C203	70	1040.4	0.00	28.4	80	1172.1	0.00	35.4	100	1456.3	0.00	38.7
C204	70	1036.8	0.00	45.3	80	1163.1	0.00	36.0	100	1448.7	0.00	59.4
C205	70	1047.9	0.00	22.9	80	1170.6	0.00	20.5	100	1460.2	0.00	36.2
C206	70	1042.0	0.00	22.6	80	1168.9	0.00	19.8	100	1455.1	0.00	22.8
C207	70	1040.3	0.00	35.1	80	1167.2	0.00	23.8	100	1454.5	0.00	39.2
C208	70	1040.3	0.00	29.0	80	1167.2	0.00	26.6	100	1451.9	0.00	31.7
R201	70	1118.4	0.00	5.1	80	1201.5	0.00	29.5	100	1399.6	0.00	191.9
R202	70	1041.1	0.00	32.1	80	1121.2	0.00	82.9	100	1304.7	0.00	1396.4
R203	70	958.0	0.00	26.7	80	1034.6	0.00	43.4	100	1204.8	0.00	338.5
R204	70	921.8	0.00	37.1	80	1002.1	0.00	171.3	100	1162.2	0.00	162.3
R205	70	1033.4	0.00	89.7	80	1103.6	0.00	92.7	100	1267.3	0.00	436.5
R206	70	985.9	0.00	94.7	80	1055.4	0.00	184.8	100	1220.9	0.00	1650.4
R207	70	942.0	0.00	29.2	80	1011.3	0.00	77.7	100	1182.5	0.00	233.2
R208	70	917.5	0.00	34.1	80	993.5	0.00	88.8	100	1157.5	0.00	185.2
R209	70	955.3	0.00	24.4	80	1034.9	0.00	54.4	100	1205.4	0.00	132.0
R210	70	980.4	0.00	27.0	80	1052.8	0.00	47.4	100	1211.8	0.00	333.6
R211	70	914.8	0.00	19.9	80	999.0	0.00	116.4	100	1160.6	0.00	214.0
RC201	70	1364.5	0.00	3.4	80	1545.8	0.00	7.1	100	1806.8	0.00	8.7
RC202	70	1284.6	0.00	4.6	80	1458.3	0.00	13.3	100	1680.2	0.00	56.8
RC203	70	1230.5	0.00	9.5	80	1392.3	0.00	8.7	100	1601.0	0.00	125.2
RC204	70	1206.6	0.00	14.1	80	1366.5	0.00	13.2	100	1574.6	0.00	55.9
RC205	70	1335.3	0.00	16.4	80	1516.8	0.00	20.4	100	1732.6	0.00	252.3
RC206	70	1285.5	0.00	6.6	80	1455.6	0.00	10.4	100	1698.1	0.00	107.2
RC207	70	1236.5	0.00	8.8	80	1402.9	0.00	49.0	100	1640.7	0.00	261.0
RC208	70	1208.2	0.00	30.9	80	1364.1	0.00	12.8	100	1570.7	0.00	56.6
C2_2_01	140	3436.2	0.00	34.9	170	4048.3	0.00	275.5	200	4623.9	0.00	98.7
C2_2_02	140	3380.3	0.00	84.4	170	3976.9	0.00	54.5	200	4562.2	0.00	267.1
C2_2_03	140	3304.6	0.00	46.8	170	3932.0	0.00	102.1	200	4517.1	0.00	261.9
C2_2_04	140	3289.5	0.00	57.0	170	3914.0	0.00	157.9	200	4505.3	0.00	429.1
C2_2_05	140	3382.9	0.00	36.3	170	3987.2	0.00	307.3	200	4558.3	0.00	120.9
C2_2_06	140	3367.4	0.00	123.5	170	3964.3	0.00	56.0	200	4543.6	0.00	156.0
C2_2_07	140	3362.5	0.00	63.4	170	3958.6	0.00	44.5	200	4528.9	0.00	110.6
C2_2_08	140	3354.6	0.00	263.6	170	3937.0	0.00	33.2	200	4517.3	0.00	97.5
C2_2_09	140	3345.1	0.00	449.7	170	3931.6	0.00	63.7	200	4512.8	0.00	126.1

Continued on next page

Table EC.2 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
C2_2_10	140	3337.4	0.00	271.1	170	3930.7	0.00	55.4	200	4511.1	0.00	122.7
R2_2_01	140	3998.9	0.00	392.6	170	4631.2	0.41	10800.1	200	5295.8	0.47	10800.0
R2_2_02	140	3734.7	0.00	76.4	170	4379.8	0.00	9107.6	200	5021.4	0.00	2784.7
R2_2_03	140	3601.9	0.00	3409.5	170	4194.6	0.00	234.3	200	4860.0	0.00	465.3
R2_2_04	140	3473.3	0.00	72.8	170	4099.7	0.00	483.4	200	4761.1	0.00	1532.8
R2_2_05	140	3859.3	0.00	251.1	170	4476.5	0.43	10800.3	200	5119.5	0.33	10800.1
R2_2_06	140	3671.9	0.00	663.8	170	4296.6	0.41	10800.4	200	4950.1	0.22	10800.1
R2_2_07	140	3558.3	0.00	3558.4	170	4170.2	0.00	501.9	200	4847.6	0.00	1358.7
R2_2_08	140	3468.4	0.00	52.1	170	4098.5	0.00	464.8	200	4760.4	0.00	1766.7
R2_2_09	140	3779.6	0.00	759.2	170	4378.1	0.27	10800.1	200	5034.4	0.48	10800.0
R2_2_10	140	3693.5	0.00	1727.4	170	4306.9	0.00	1708.5	200	4942.0	0.00	3466.3
RC2_2_01	140	3718.2	0.00	253.3	170	4404.3	0.61	10800.2	200	4902.0	0.00	2177.3
RC2_2_02	140	3573.6	0.11	10800.1	170	4231.7	0.00	1717.9	200	4795.5	0.00	2486.4
RC2_2_03	140	3487.5	0.00	1291.5	170	4160.2	0.31	10800.0	200	4733.5	0.55	2768.1
RC2_2_04	140	3449.3	0.00	254.7	170	4126.3	0.00	2212.8	200	4688.9	0.51	1597.6
RC2_2_05	140	3598.7	0.00	730.2	170	4302.0	0.00	3388.3	200	4841.8	0.00	8746.5
RC2_2_06	140	3622.4	0.00	6169.8	170	4297.2	0.00	2196.3	200	4844.7	0.29	10800.0
RC2_2_07	140	3565.6	0.00	270.6	170	4242.6	0.00	7240.6	200	4790.3	0.30	10800.1
RC2_2_08	140	3539.1	0.00	3312.3	170	4231.2	0.49	10800.2	200	4776.0	0.00	4374.7
RC2_2_09	140	3532.4	0.00	1325.1	170	4236.6	1.21	10800.2	200	4752.0	0.00	10218.8
RC2_2_10	140	3511.4	0.00	819.8	170	4197.6	0.00	9514.9	200	4739.4	0.00	1619.4

Table EC.3: 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
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

Continued on next page

Table EC.3 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
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.4: 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
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

Continued on next page

Table EC.4 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
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
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

Continued on next page

Table EC.4 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
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
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

Continued on next page

Table EC.4 – *Continued from previous page*

Name	n	ub	Gap%	CPU	n	ub	Gap%	CPU	n	ub	Gap%	CPU
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