The Joint Network Vehicle Routing Game

Mathijs van Zon[1], Remy Spliet, Wilco van den Heuvel

Econometric Institute, Erasmus School of Economics, Erasmus University Rotterdam
PO Box 1738, Rotterdam 3000 DR, The Netherlands

Econometric Institute Report Series - EI2019-03

Abstract

Collaborative transportation can significantly reduce transportation costs as well as greenhouse gas emissions. However, allocating the cost to the collaborating companies remains difficult. We consider the cost-allocation problem which arises when companies, each with multiple delivery locations, collaborate by consolidating demand and combining delivery routes. We model the corresponding cost-allocation problem as a cooperative game: the joint network vehicle routing game (JNVRG). We propose a row generation algorithm to determine a core allocation for the JNVRG. In this approach, we encounter a row generation subproblem which we model as a new variant of a vehicle routing problem with profits. Moreover, we propose two main acceleration strategies for the row generation algorithm. First, we generate rows by relaxing the row generation subproblem, exploiting the tight LP bounds for our formulation of the row generation subproblem. Secondly, we propose to also solve the row generation subproblem heuristically and to only solve it to optimality when the heuristic fails. We demonstrate the effectiveness of the proposed row generation algorithm and the acceleration strategies by means of numerical experiments for both the JNVRG as well as the traditional vehicle routing game, which is a special case of the JNVRG. We create and solve instances based on benchmark instances of the capacitated vehicle routing problem from the literature, ranging from 5 companies with a total of 79 delivery locations to 53 companies with a total of 53 delivery locations.

Keywords: Collaborative transportation, Cooperative game theory, Vehicle Routing, Row generation, Vehicle routing with profits, Branch-and-cut-and-price

1 Introduction

Consider a company which operates a distribution network with multiple delivery locations. The delivery locations are served by a fleet of vehicles from a central depot. Often, multiple of these distribution networks are located within the same area. For example, consider several retailers operating within the same city centres. Close proximity of their delivery locations provides companies with the opportunity to collaborate. Such collaborations can consist of consolidating demands and combining delivery routes, effectively joining their distribution networks. In over ten case studies with real life data (Guajardo and Rönnqvist, 2016), collaborative transportation provides potential cost savings ranging from 4% (Lehoux et al., 2011) to 46% (Engevall et al., 2004). Moreover, Ballot and Fontane (2010) report a reduction in CO_2 emissions by at least 25% as a consequence of collaborative transportation.

Despite the potential benefits, many collaboration opportunities are left unused in practice. One reason is that the remaining cost after collaboration still needs to be divided among the companies. Reaching an agreement can be difficult as the concerned distribution networks are

[1]Corresponding author. E-mail: vanzon@ese.eur.nl

likely to have different characteristics. For example, each company may have a different number of delivery locations and the demand volumes may differ as well. Nonetheless, all routing costs need to be allocated to the companies. In this paper, we study the cost allocation problem which arises from collaborative transportation of multiple companies, each having multiple delivery locations.

In various settings, cost allocation problems are modelled as cooperative games, see Guajardo and Rönnqvist (2016) for an overview of cooperative games in transportation. In cooperative game terminology, it is common to refer to the companies as players, and to refer to a group of players as a coalition. A cooperative game is characterised by a set of players, known as the grand-coalition, and a cost function which maps each coalition to a specific cost. The goal of a cooperative game is to allocate the cost of the grand-coalition to the players. To this end, it is common to look for a so-called core allocation. A core allocation ensures that each coalition is worse off if they decide not to cooperate in the grand-coalition. The set of all such allocations is known as the core.

Among the first collaborative transportation games is the travelling salesman game (TSG), in which each player corresponds to a single delivery location and all deliveries are made on a single distribution route. The cost of a coalition is defined as the optimal objective value of a travelling salesman problem (TSP). See Applegate et al. (2006) for more on the TSP. The TSG was originally proposed by Fishburn and Pollak (1983), and the core of the TSG has been studied by Dror (1990) and Potters et al. (1992). For a case study on the TSG, we refer the reader to Engevall et al. (1998), who determine core allocations for a single instance consisting of 5 players.

The vehicle routing game (VRG) is a generalisation of the TSG, where each player is assumed to have a certain demand which has to be satisfied by a fleet of vehicles with finite capacity. For the VRG, the cost of a coalition is given by the optimal objective value of a capacitated vehicle routing problem (CVRP). See Toth and Vigo (2014) for more on the CVRP. The problem of determining a core allocation for the VRG has been studied by Göthe-Lundgren et al. (1996), who present numerical experiments for instances with at most 25 players. Furthermore, Engevall et al. (2004) consider the VRG with a heterogeneous fleet and solve a single instance consisting of 21 players.

In order to determine a core allocation, the core is often modelled as a linear programming problem consisting of an exponential number of constraints, one for each coalition. For a small number of players, we might include all constraints and determine the cost for each coalition. Clearly, if the number of players is large or if computing the cost of a coalition is computationally expensive, this approach is no longer tractable. For example, in the TSG, a TSP has to be considered for each coalition, which is an NP-hard problem. Similarly, in the VRG, exponentially many CVRPs are considered. Note that computationally the CVRP seems more difficult than the TSP, since benchmark instances for the CVRP with up to 360 customers can be solved to optimality by state-of-the-art algorithms (Pecin et al., 2017a), whereas benchmark instances of up to 85900 customers have been solved to optimality for the TSP (Applegate et al., 2006).

Interestingly, Göthe-Lundgren et al. (1996) show that in order to describe the core of the VRG, only coalitions which can be served on a single distribution route have to be considered. As a result, one can solve a TSP instead of a CVRP to compute the cost of any relevant coalition. However, as the number of coalitions remains large, Göthe-Lundgren et al. (1996) propose a row generation algorithm to determine a core allocation. In their approach they assume that the core is non-empty. As a consequence, the row generation subproblem they encounter simplifies, and can be modelled as a travelling salesman problem with profits. Engevall et al. (2004) propose a row generation algorithm to determine the Nucleolus of a VRG with heterogeneous fleet, even if the core is empty. They show that the row generation subproblem in this case can be modelled as

a vehicle routing problem with profits. However, the authors do not propose a method to solve their row generation subproblem. Instead, they illustrate the effectiveness of the row generation generation technique in an instance with 21 players by enumerating the CVRP cost of over 2 million coalitions, for which they report a runtime of roughly 12 weeks.

In contrast to the TSG and VRG where each player corresponds to a single delivery location, in many real-life applications, each company has multiple delivery locations. In the following, we consider cooperative transportation games in which each player corresponds to multiple delivery locations. Krajewska et al. (2008) and Dai and Chen (2015) consider a cooperative game in which each player has a set of delivery requests. In that case, an order has to be collected from a pickup location before it is delivered. Krajewska et al. (2008) define the cost as the objective value of applying a heuristic to the pickup-and-delivery problem. The authors enumerate the cost for each coalition, and determine the Shapley value (Shapley, 1953) for instances consisting of 5 players, each with 50 delivery requests. For each allocation they also determine whether it is in the core of their game. As opposed to Krajewska et al. (2008), Dai and Chen (2015) define the cost as the optimal objective value to the pickup-and-delivery problem. They propose a row generation algorithm to determine a core allocation for the game. To this end, they formulate both the pickup-and-delivery problem and the row generation subproblem as MIPs, and use a general purpose MIP solver to solve these problems. Dai and Chen (2015) consider instances consisting of at most 5 players and 25 requests. For half of the instances consisting of 5 players, it took them over 12 hours to compute a core allocation.

To the best of our knowledge, the setting of Zakharov and Shchegryaev (2015) is closest to our work. The authors consider a setting in which each player is responsible for multiple delivery locations. Given a coalition, they determine the cost by heuristically solving a CVRP consisting of the delivery locations of the players in the coalition. Based on these heuristic costs, they determine a core allocation for an instance consisting of 4 players and 200 delivery locations by enumerating all coalitions.

Contrary to Zakharov and Shchegryaev (2015), we define the cost of a coalition as the optimal solution to a CVRP over the delivery locations of the players in the coalition. We call the corresponding cooperative game the joint network vehicle routing game (JNVRG). We define the cost as the optimal value for the following two reasons. First, a core allocation determined by heuristic costs instead of optimal costs may not be stable in practice. A coalition might be able to improve on the heuristic solution which could correspond to a cost that is lower than what is allocated. Secondly, when the cost of a coalition is determined by a heuristic, it is not clear how to find a core allocation without enumerating all coalitions. As a result, cooperative games using heuristic costs do not scale well for a large number of players. Alternatively, using optimal objective values allows for the application of row generation. The trade off is that more players can be considered while the number of delivery locations will be limited.

Our main contribution is as follows. We propose a row generation algorithm to determine core allocations for the JNVRG. In our solution approach, we encounter a row generation subproblem which we model as a generalisation of a vehicle routing problem with profits, which to our best knowledge has not been studied before. Furthermore, we propose two main acceleration strategies for the row generation algorithm. First, we introduce what we call coarse row generation, a technique where we exploit the tight bounds of the set-partitioning formulation of the CVRP. In particular, we relax the row generation subproblem to accelerate the identification of violated core constraints. Secondly, we solve the row generation subproblem heuristically before using an exact algorithm in order to limit the number of times the row generation subproblem has to be solved to optimality. Furthermore, we demonstrate the effectiveness of the proposed row generation algorithm by means of numerical experiments. We are able to solve instances ranging from 5 players and 79 delivery locations to 53 players and 53 delivery locations within 2

hours of computation time. Moreover, we demonstrate that the proposed acceleration strategies are also effective for the VRG.

The remainder of this paper is structured as follows. In Section 2 we introduce the JNVRG. Next, in Section 3 we propose a row generation algorithm to determine a core allocation in an efficient manner. We present several existing acceleration strategies for solving the row generation subproblem in Section 4. In Section 5 we propose new acceleration strategies for the row generation algorithm. Our computational results are presented and discussed in Section 6. Finally, we provide some concluding remarks and suggestions for future research in Section 7.

2 The Joint Network Vehicle Routing Game

In order to formally introduce the JNVRG, we first define the CVRP. Consider a complete directed graph $G = (V, A)$. The vertex set V is defined as $\{0\} \cup V'$, where 0 represents the depot and V' represents the set of all delivery locations, also referred to as customers. With each arc $(v, w) \in A$, a travel cost $c_{vw} \geq 0$ is associated, which we assume to adhere to the triangle inequality. Furthermore, each customer $v \in V'$ has a demand $d_v > 0$. An unlimited number of vehicles with capacity Q is available at the depot to satisfy the demand by visiting customers along routes. A route is defined as a simple cycle, starting and ending at the depot, such that the total demand does not exceed the capacity. We assume that $d_v \leq Q$ for all $v \in V'$. The CVRP is the problem of constructing routes in such a way that the total cost is minimised and every customer is visited exactly once.

Next, we define the JNVRG. Let $N = \{1, 2, \ldots, n\}$ be the set of players. The set V' is the combined set of the customers of these players. In particular, $V'(i) \subset V'$ represents the customers of player i where $\bigcup_{i \in N} V'(i) = V'$ and $V'(i) \cap V'(j) = \emptyset$ for $i, j \in N$ with $i \neq j$. The cost $C(S)$ of coalition $S \subseteq N$ is the minimal cost of the CVRP on the subgraph $G(S)$ induced by $V'(S) \cup \{0\}$, where $V'(S) = \bigcup_{i \in S} V'(i)$ represents all customers of the players in coalition S. We define the JNVRG as an n-person cooperative game $\langle N, C \rangle$, where the aim is to allocate the cost $C(N)$ to the players.

2.1 Characterisation of the core

As is common in cooperative game theory, we search for an allocation in the core. In order to define the core, we denote the cost allocated to player i as y_i and the cost allocated to the coalition $S \subseteq N$ as $y(S) = \sum_{i \in S} y_i$. A cost allocation $y \in \mathbb{R}^n$ is said to be efficient if $y(N) = C(N)$. Furthermore, a cost allocation is said to be rational if it satisfies the rationality constraints $y(S) \leq C(S)$ for all coalitions $S \subset N$. The rationality constraints ensure that no coalition has an incentive not to cooperate. The set of rational and efficient cost allocations is known as the core (Gillies, 1959) and is described as

$$\text{Core}(\langle N, C \rangle) = \{y \in \mathbb{R}^n : y(S) \leq C(S)\ \forall S \subset N, y(N) = C(N)\}.$$

If no rational and efficient cost allocation exists, the core is said to be empty. The core of the JNVRG is related to the core of the VRG, since the VRG is a special case of the JNVRG in which each player has exactly one customer. Göthe-Lundgren et al. (1996) show that for the VRG there exist instances with an empty core. Because the VRG is a special case of the JNVRG, we conclude that there also exist instances of the JNVRG for which the core is empty.

One way to determine whether the core of a specific JNVRG instance is non-empty is by applying the Bondareva-Shapley theorem of Bondareva (1963) and Shapley (1967). This theorem states that the core of a game is non-empty if and only if the game is balanced. For convenience, we define balancedness by means of an optimisation problem. A game is said to be balanced

if and only if $C(N)$ is smaller than or equal to the optimal objective value C^* of the following minimisation problem:

$$C^* = \min \quad \sum_{S \subset N} x_S C(S) \tag{2.1}$$

$$\text{s.t.} \quad \sum_{S \subset N} \alpha_S^i x_S = 1 \qquad \forall\, i \in N, \tag{2.2}$$

$$x_S \geq 0 \qquad \forall\, S \subset N, \tag{2.3}$$

where x_S is a non-negative decision variable providing the amount by which coalition S is selected and α_S^i is a binary parameter which equals 1 if and only if $i \in S$.

Göthe-Lundgren et al. (1996) implicitly use the Bondareva-Shapley theorem to determine whether the core of a VRG is non-empty. To do so, they make use of so-called feasible coalitions, which they define as coalitions of which the delivery locations can be visited on a single distribution route. They show that the cost of a non-feasible coalition is precisely the sum of the costs of the feasible coalitions corresponding to each route which is part of an optimal solution for the CVRP of the non-feasible coalition. As a result, only feasible coalitions have to be considered to describe the core, and all other coalitions can be disregarded. In this case, the computation of the cost of a coalition reduces to solving a TSP instead of a CVRP.

When only considering feasible coalitions, (2.1)-(2.3) reduces to the LP-relaxation of the set-partitioning formulation of the CVRP of the grand-coalition, in which the variable x_S corresponds to a single route. Moreover, it follows that $C^* = C(N)$ if and only if the LP-relaxation of the set-partitioning formulation of the CVRP of the grand-coalition has an integer optimal solution. Interestingly, it follows that by solving the LP-relaxation of the set-partitioning formulation of a CVRP, one can determine whether the core of the VRG is non-empty.

A similar result does not hold for the JNVRG, as coalitions which are served by multiple vehicles cannot be disregarded. This is because in contrast to the VRG, the customers of a single player might be visited on multiple routes. Therefore, we cannot determine a core allocation by only considering TSPs for a limited set of coalitions, instead we consider a CVRP for all coalitions.

These results also provide insight into how often one can expect a VRG core to be empty. It is well known that many CVRP instances do not have an integer solution to the LP-relaxation of the set-partitioning formulation. Hence, we can expect the core of the VRG to be empty quite often. This expectation does not directly apply to the JNVRG. To see this, construct an instance of the JNVRG by combining multiple players of the VRG into single players of the JNVRG. This means that the delivery locations of a single player in the JNVRG are in fact the combined delivery locations of multiple players of the VRG. Observe that the rationality constraints of this instance of the JNVRG are a subset of the rationality constraints of the VRG. Hence, the JNVRG is a restricted game with respect to the VRG, see Faigle (1989) for more on restricted games. As a result, the core of a VRG is contained in the core of a corresponding JNVRG. Put differently, if the core of the VRG is empty, this does not imply that the core of a corresponding JNVRG is empty, although the reverse implication does hold.

2.2 Cost allocation

Well-known allocations are the Nucleolus (Schmeidler, 1969), the Lorenz allocation (Arin, 2003) and the allocation given by the Equal Profit Method (EPM) of Frisk et al. (2010). All of these allocations can be found by solving linear programming problems that include all rationality constraints. In this paper, we use the EPM allocation for illustrative purposes, although our proposed approach could easily be applied to find other allocations as well.

An EPM allocation is a core allocation which minimises the maximum difference in allocated cost to each player, relative to their individual cost. The EPM allocation y can be determined by solving the linear programming problem:

$$\min \quad \theta \tag{2.4}$$

$$\text{s.t.} \quad \frac{y_i}{C(\{i\})} - \frac{y_j}{C(\{j\})} \leq \theta \qquad \forall\, i, j \in N, \tag{2.5}$$

$$y(S) \leq C(S) \qquad \forall S \subset N, \tag{2.6}$$

$$y(N) = C(N), \tag{2.7}$$

$$y_i \geq 0 \qquad i \in V', \tag{2.8}$$

$$\theta \in \mathbb{R}. \tag{2.9}$$

In an optimal solution, the maximum difference in relative costs is given by the decision variable θ as enforced by Constraints (2.5). The rationality constraints (2.6) and efficiency constraint (2.7) ensure that the allocation is in the core. Like Frisk et al. (2010), we include Constraints (2.8). However, note that due to the monotonicity of the costs of each coalition, these constraints are implied by the rationality constraints. Finally, Constraint (2.9) specifies the domain of the decision variable θ.

3 A row generation algorithm

We determine an EPM allocation for the JNVRG by solving (2.4)-(2.9). The inclusion of the rationality constraints (2.6) poses two challenges. First, there are exponentially many of these constraints. Second, to determine the cost of a coalition, a CVRP has to be solved, which is known to be NP-hard (Lenstra and Rinnooy Kan, 1981). To alleviate both challenges, we propose a row generation method in which these constraints are first omitted and then iteratively generated if they are violated. This approach is similar to the approach presented by Göthe-Lundgren et al. (1996) and Engevall et al. (2004), but differs in the row generation subproblem.

We first present the general idea of the row generation algorithm which we use to determine an EPM allocation. An initial cost allocation y is obtained by solving (2.4)-(2.9) with (2.6) only for the singleton coalitions. Note that the non-negativity constraints (2.8) are no longer implied in the absence of some rationality constraints, which is why we explicitly include them. Next, a row generation subproblem is solved to determine whether there exists a coalition for which the rationality constraint is violated by the current allocation y. If a violated rationality constraint is identified, the constraint is added to the relaxed problem. This procedure is repeated until no more violated constraints can be identified, indicating that the solution is optimal, or the problem becomes infeasible, indicating that the core is empty.

In the remainder of this section we present the row generation algorithm in more detail. A description of the row generation subproblem is given in Section 3.1. In Section 3.2, we propose a branch-and-price algorithm to solve the row generation subproblem.

3.1 The row generation subproblem

Consider a cost allocation y. A violated rationality constraint is characterised by a coalition $S \subset N$ such that $y(S) > C(S)$. This gives rise to the following row generation subproblem:

$$\mathrm{SP}(y) = \max_{S \subset N} \left\{ y(S) - C(S) \right\}. \tag{3.1}$$

If we find that the optimal objective value $\mathrm{SP}(y) > 0$ a violated rationality constraint has been identified, otherwise the current allocation is an EPM allocation. Note that the routing costs

$C(S)$ of the potentially violating coalitions have not been determined, and are also part of the row generation subproblem.

Next, we introduce an integer programming formulation for the row generation subproblem. Let R denote the set of feasible routes in $G(N)$. Define the coefficient a_r^v as the number of times customer v is included in route r, hence $a_r^v = 0$ if customer v is not included in r and $a_r^v = 1$ if customer v is included exactly once in r. We define the binary decision variable z_i such that $z_i = 1$ if and only if player i is included in the selected coalition, which we denote by $S(z) = \{i \in N : z_i = 1\}$. Furthermore, let $i(v)$ denote the player corresponding to customer v. Define the binary decision variable x_r such that $x_r = 1$ if and only if route r is selected. Let c_r be the cost of route r which is given by the sum of the arc costs on route r. We can formulate the row generation subproblem as follows:

$$\text{GCPTP}(y) = \max \quad \sum_{i \in N} y_i z_i - \sum_{r \in R} c_r x_r \tag{3.2}$$

$$\text{s.t.} \quad \sum_{r \in R} a_r^v x_r \geq z_{i(v)} \qquad \forall\, v \in V', \tag{3.3}$$

$$x_r \in \{0, 1\} \qquad \forall\, r \in R, \tag{3.4}$$

$$z_i \in \{0, 1\} \qquad \forall\, i \in N. \tag{3.5}$$

Objective (3.2) represents the amount by which the rationality constraint of the selected coalition $S(z)$ is violated. Constraints (3.3) enforce that all locations of a player are visited at least once if the player is included in the selected coalition. Finally, Constraints (3.4) and (3.5) specify the domains of the decision variables.

Consider an optimal solution (x^*, z^*). In this case, x^* is an optimal solution to the CVRP of coalition $S(z^*)$. Hence, after solving the row generation subproblem, the optimal routing cost of the most violated coalition has also been determined, and the corresponding rationality constraint $y(S(z^*)) \leq C(S(z^*))$ can directly be added to the relaxed problem without having to solve an additional CVRP.

The row generation subproblem can be seen as a generalisation of a capacitated profitable tour problem (CPTP) as studied by Archetti et al. (2013). In the CPTP not all customers need to be visited whereas a prize is collected for customers which are visited. The goal of the CPTP is to maximise the collected prizes minus the CVRP cost $C(S)$. Our row generation subproblem is a generalisation of the CPTP such that a prize y_i is collected if and only if all locations $V'(i)$ of player i are visited. We call this modified problem the grouped capacitated profitable tour problem (GCPTP).

3.2 Solving the row generation subproblem

In order to determine an optimal solution to the row generation subproblem, we propose a branch-and-price algorithm. In each node of the branch-and-bound tree we determine an upper bound for the row generation subproblem by solving the linear relaxation of (3.2)-(3.5), referred to as the master problem (MP). Since the number of route variables x_r is high, we apply column generation to solve the MP in each node. To this end, we define the restricted master problem (RMP) as the MP in which we restrict the set of variables. Then, we solve the RMP and search for any positive reduced cost variables by solving a pricing problem, which we introduce next. If such variables are found, they are added to the RMP and the process is repeated. If no more positive reduced cost variables exist, the solution to the RMP is also optimal for the MP and gives a valid upper bound for the row generation subproblem. Finally, if a solution is fractional, we branch on either fractional player variables or fractional arc flows.

3.2.1 The pricing problem

The pricing problem is the problem of identifying a positive reduced cost variable. In most CVRP literature employing column generation, the pricing problem is a minimisation problem to identify a negative reduced cost variable. Therefore, in an attempt to avoid confusion, we reverse the sign of the reduced cost and model the pricing problem as a minimisation problem as well, and search for negative reduced cost variables.

In order to guarantee optimality of the RMP, we must show that no routes with negative reduced cost exist. The reduced cost $\bar{c}_r$ of a route r can be expressed as the sum over the modified arc costs $\bar{c}_{vw}$:

$$\bar{c}_{vw} = c_{vw} - \frac{\mu_v + \mu_w}{2}, \quad v, w \in V' \cup \{0\} \tag{3.6}$$

where $\mu_v \geq 0$ is a dual variable associated with Constraints (3.3) and $\mu_0 = 0$. We view the pricing problem as an Elementary Shortest Path Problem with Capacity Constraints (ESPPCC) in the graph $G(N)$ with the modified arc cost. To efficiently solve the ESPPCC, Label-setting algorithms are often used. In the following we present a label-setting algorithm, which is similar to that of Martinelli et al. (2014).

A partial path in $G(N)$ starting at the depot and ending at vertex $v \in V' \cup \{0\}$ is represented by a label λ which is defined as follows:

$$\lambda = (v, C, D, U)\,. \tag{3.7}$$

Here, C is the cost of the partial path, the load D is the demand satisfied by the partial path and U is a binary vector consisting of entries U_w which indicate whether the partial path may be extended to vertex $w \in V(N) \cup \{0\}$. As proposed by Feillet et al. (2004), the entry U_w is set to 1 if vertex w has already been visited by the partial path or if the capacity constraint does not allow vertex w to be visited. The extension λ' of a partial path $\lambda = (v, C, D, U)$ from vertex v to vertex w is determined as

$$\lambda' = (w, C + \bar{c}_{vw}, D + d_w, f(U))\,, \tag{3.8}$$

where the resources U are updated by a function $f(U) : \mathbb{B}^{|V'|} \to \mathbb{B}^{|V'|}$. For $w' \in V(N) \cup \{0\}$ such that $w' \neq w$, $f_{w'}(U)$ equals 1 if and only if $U_{w'} = 1$ or $D + d_w + d_{w'} > Q$. Furthermore, $f_w(U)$ equals 1.

Next, we present a dominance rule to limit the number of partial paths which have to be considered. Consider two distinct labels $\lambda = (v, C, D, U)$ and $\lambda' = (v, C', D', U')$, both with end vertex v. If every completion of the partial path corresponding to λ' leads to a worse solution than the same completion of λ we say that λ dominates λ', in which case label λ' can be disregarded. Label λ dominates λ' if the following conditions hold:

$$C \leq C', \tag{3.9}$$
$$D \leq D', \tag{3.10}$$
$$U \leq U'. \tag{3.11}$$

For efficient implementation, we make use of the fact that demand is integer in the benchmark that instances we use and keep labels in buckets $B(v, q)$ based on the end vertex $v \in V'$ and load $q \in \{0, \ldots, Q\}$. We store the labels sorted on reduced cost to reduce the number of times we have to verify criterion (3.9). Initially, label $(0, 0, 0, \mathbf{0})$ is added to bucket $B(0, 0)$, the remaining buckets are initialised as empty. Then, in increasing order of load all partial paths are extended to neighbouring nodes. After the extension of a label, the dominance rule is utilised to verify if

the extended label dominates any of the existing labels with equal load q and end vertex v, if this is the case, the label is stored in bucket $B(v, q)$. Any existing labels which are dominated in the process are removed from the bucket. If no labels are dominated by the extended label, we verify if any of the existing labels dominate the extended label. If none of the existing labels dominate the extended label, it is added to the corresponding bucket $B(v, q)$.

Each negative reduced cost label included in a bucket associated to the depot corresponds to a negative reduced cost route. After the algorithm has been executed, the best ρ_E found negative reduced cost routes are added to the RMP.

3.2.2 Branching strategy

The optimal solution to the MP may be fractional. We propose a branching scheme to obtain an optimal integer solution. First, we branch on the sum of the variables z_i. Then, we branch on the most fractional z_i, i.e., the z_i which value is closest to $\frac{1}{2}$. If all z_i are integer, we branch on the number of vehicles used. Finally, we branch on the most fractional arc flow. Here we use the well-known result that integer arc flows also correspond to integer routes. When choosing a node to branch on, we consider the node with the highest upper bound.

4 Acceleration strategies for solving the subproblem

In the following, we discuss several acceleration strategies from the CVRP literature and apply them in our algorithm for the row generation subproblem which we model as a GCPTP. First, in Section 4.1 we describe ng-route relaxation (Baldacci et al., 2011) for the row generation subproblem in order to reduce the computational effort required to solve the pricing problem. Second, in Section 4.2 we describe decremental state space relaxation (Martinelli et al., 2014) to further accelerate the label-setting algorithm. In Section 4.3, we present a heuristic for the pricing problem. Finally, in Section 4.4 we describe the limited memory subset-row cuts (Pecin et al., 2017a) which we use to improve the LP bound of the MP.

4.1 ng-Routes

The formulation of the row generation subproblem is relaxed by allowing non-elementary routes, originally applied by Desrochers et al. (1992). Note that this does not alter the optimal objective value as it is never better to visit a customer multiple times. The benefit of allowing non-elementary routes is that it reduces the computational effort required to solve the pricing problem. However, this comes at the expense of weaker LP bounds. We consider a subset of non-elementary routes known as ng-routes, which were introduced by Baldacci et al. (2011) and typically provide a good trade-off between weaker LP bounds and reduced computation times for the pricing problem. An ng-route is defined as a route which is allowed to visit a vertex v multiple times, if and only if, between the visits, at least one vertex v' is visited such that $v \notin \Pi_{v'}$, where $\Pi_{v'} \subseteq V'$ represents the ng-neighbourhood of vertex v'.

We implement ng-routes by altering the label-setting algorithm. We modify the definition of a label such that at vertex v we only keep track of whether the label can be extended towards the vertices in the ng-neighbourhood Π_v. It is assumed that a label can be extended to all vertices not in its neighbourhood, if this does not violate the capacity constraints. When extending a label from vertex v to vertex w we update the label according to (3.8) except for the binary vector U, which is updated according to $f'(U)$. For $w' \in V'$ such that $w' \neq w$, $f'_{w'}(U)$ equals 1 if and only if $D + d_w + d_{w'} > Q$ or $w' \in \Pi_w$ and $U_{w'} = 1$. Moreover, $f'_w(U)$ equals 1. Finally, we alter the dominance rule as follows. We say that a label $\lambda = (v, C, D, U)$ dominates another

label $\lambda' = (v, C', D', U')$ if the following condition

$$U_w \leq U'_w \text{ for } w \in \Pi_v \tag{4.1}$$

holds along with conditions (3.9)-(3.10).

4.2 Decremental State Space Relaxation

To reduce the size of the ng-neighbourhoods, we apply decremental state space relaxation as proposed by Martinelli et al. (2014), which is demonstrated to be effective at improving the computational performance of label-setting algorithms. To this end, we execute the label-setting algorithm with auxiliary ng-neighbourhoods Γ_v instead of the real ng-neighbourhoods Π_v for each vertex $v \in V'$. Each time we solve a MP, the neighbourhood Γ_v is initialised as $\{v\}$ for all $v \in V'$. If the algorithm identifies a negative reduced cost route r, which is not an ng-route with respect to the neighbourhoods $\Pi_v = V'$, the neighbourhoods Γ_v are updated as follows. For each cycle in r which is not allowed, vertex v is added to the neighbourhoods of all vertices included in the cycle. In this manner, the cycle can no longer be generated in any subsequent iterations of the pricing problem. Then, the label-setting algorithm is restarted with the updated neighbourhoods.

4.3 Heuristic labelling

Before calling the label-setting algorithm, we solve the pricing problem heuristically. If one or multiple routes with negative reduced costs are found by the heuristic, the best ρ_H are added to the RMP. Otherwise, the exact label-setting algorithm is used. As a heuristic, we use a modified version of the label-setting algorithm, in which we only keep the label with the lowest reduced cost in each bucket not belonging to the depot in order to reduce the number of potential labels.

4.4 Limited memory subset row cuts

The bound of the RMP can be improved by including valid inequalities. An effective family of valid inequalities for the CVRP, which are also valid for our MP, are the limited memory subset-row cuts (LM-SRCs) as introduced by Pecin et al. (2017a). These cuts provide a good trade-off between computation time and strength of the cuts (Pecin et al., 2017b). They are an extension of the subset-row cuts (SRCs ; Jepsen et al., 2008) which we briefly explain first. Consider a fractional routing solution as visualised in Figure 4.1.

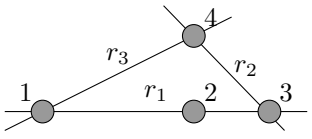

Figure 4.1: Example of a fractional routing solution.

Suppose routes $r_1 = (0, 1, 2, 3, 0), r_2 = (0, 3, 4, 0)$ and $r_3 = (0, 1, 4, 0)$ are all used $\frac{1}{2}$ times in an optimal solution to the RMP, where 0 denotes the depot. It is clear that these routes cannot be included together in any integral optimal solution to the MP, which is prevented by including SRCs. As is common, we limit ourselves to SRCs of size 3 with coefficient $\frac{1}{2}$. Given $W \subset V'$ of cardinality 3 and coefficient $\frac{1}{2}$, a SRC is formulated as

$$\sum_{r \in R} \left\lfloor \frac{1}{2} \sum_{v \in W} a_r^v \right\rfloor x_r \leq 1. \tag{4.2}$$

The labelling algorithm has to be modified to deal with additional dual variables from the inclusion of the SRC. An additional resource is added to a label for each included SRC and only a weaker form of dominance can be applied (Pecin et al., 2017b). To reduce the computational impact on the pricing algorithm, we consider the LM-SRC. Given $W \subset V'$ such that $|W| = 3$, a LM-SRC with coefficient $\frac{1}{2}$ can be formulated as follows:

$$\sum_{r \in R} \alpha(W, M, r) x_r \leq 1, \tag{4.3}$$

where $M \subseteq V'$ is the memory set and the coefficient α is determined in the following iterative manner. Set $\alpha = 0$, consider the first customer v on route r, if $v \in W$ we increase α by $\frac{1}{2}$, else if $v \in V' \backslash M$ we set $\alpha = \lfloor \alpha \rfloor$, otherwise nothing happens. We repeat this process for each customer in r in order of appearance. Consider the previous example again. Suppose that $W = \{1, 3, 4\}$ and $M = \{1, 3, 4\}$, it holds that $\alpha(W, M, r_1) = 0$ and $\alpha(W, M, r_2) = \alpha(W, M, r_3) = 1$. In this case, the cut does not prohibit the fractional solution. However, setting $M = \{1, 2, 3, 4\}$ yields $\alpha(W, M, r_1) = \alpha(W, M, r_2) = \alpha(W, M, r_3) = 1$, which does cut off the fractional solution. By increasing the size of the memory, the LM-SRCs become stronger at the cost of increased computational effort.

To account for the inclusion of LM-SRCs, the label-setting algorithm is modified. Each label is extended to hold an additional list of resources S consisting of resources S_ω for each LM-SRC $\omega \in \Omega$, where Ω consists of all LM-SRC identified so far for which the current associated dual σ_ω is non-zero and the current customer of the label is included in the memory set of ω. Furthermore, when extending a label λ from vertex v to vertex w, we iterate over the resources in Ω. If w is included in the customer set of the corresponding LM-SRC, the resource S_ω is increased by $\frac{1}{2}$. Then, if the resource is equal to 1, the resource value is reduced to 0 and the dual is subtracted from the reduced cost of the label. Otherwise, if w is not included in the memory set of the corresponding LM-SRC, the resource is also set to 0. Finally, the dominance rule is modified (Pecin et al., 2017a). Consider two distinct labels $\lambda = (C, D, U, S)$ and $\lambda' = (C', D', U', S')$. Label λ dominates λ' if the following condition holds:

$$C \leq C' + \sum_{\omega \in \Omega : S_\omega > S'_\omega} \sigma_\omega, \tag{4.4}$$

along with conditions (3.10) and (4.1). We separate the LM-SRC by enumeration, once the MP has been solved to optimality. To this end, we consider all subsets $W \subset V'$ where $|W| = 3$, $\forall\, v, w \in W$, $v \in \Pi_w$ and $w \in \Pi_v$. Given a subset W, we generate a LM-SRC if the corresponding SRC is violated by at least 0.1. Then $M \subseteq V'$ is determined as follows. For each route included in the current optimal solution, all customers between the first and second visit to a customer in W are included in the memory. The same is done for any visited customer between the third and fourth visit to a customer in W, and so on. For example, when considering Figure 4.1 once more, one can see that route r_1 requires customer 2 to be included in the memory.

5 Acceleration strategies for the row generation algorithm

In this section, we propose several acceleration strategies for the row generation algorithm. Note that these acceleration strategies can also be applied to determine a core allocation for the VRG. First, we present a relatively straightforward acceleration strategy after which we introduce the two main acceleration strategies.

We store all columns which are generated by the row generation subproblem algorithm in a column pool. Then, in each iteration of the row generation algorithm, the row generation

subproblem is initialised with all columns in the column pool. In this manner the row generation subproblem is provided with a better initial solution at no additional computational cost.

Secondly, we propose an alternative row generation procedure. By relaxing the row generation subproblem to allow fractional route variables, a potentially violated rationality constraint is identified. Then, the cost of the identified coalition is determined by solving a CVRP. The aim of this method is to separate the identification of a violated constraint and the computation of the right-hand side value, such that both procedures can be accelerated separately. This procedure, which we call coarse row generation, is explained in detail in Section 5.1.

Thirdly, we propose to first generate rows heuristically to potentially reduce the number of calls to the exact algorithm for the row generation subproblem. This procedure is elaborated upon in Section 5.2.

Finally, in Section 5.3 we present an overview of the different variants of row generation algorithms we consider, which follow from combining coarse and heuristic row generation.

5.1 Coarse row generation

Separating rationality constraints by solving the row generation subproblem to integer optimality might be computationally intensive. Rather than modelling the vehicle routing cost exactly in the row generation subproblem, we propose to use a lower bound on the routing cost. We truncate the branching procedure, by no longer branching on the arc flow, to obtain this lower bound. Effectively, we replace the routing cost $C(S)$ of the row generation subproblem by the lower bound $\underline{C(S)}$, where $\underline{C(S)}$ is given by the optimal objective value to the LP relaxation of the set-covering formulation for the CVRP including LM-SRCs. We call this row generation subproblem the coarse row generation subproblem, which can be formulated as follows:

$$\max_{S \subset N} \left(y(S) - \underline{C(S)} \right). \tag{5.1}$$

Note that $\max_{S \subset N}(y(S) - C(S)) \leq \max_{S \subset N} \left(y(S) - \underline{C(S)} \right)$. Hence, the coarse row generation subproblem yields an upper bound for the row generation subproblem. Given a cost allocation y, we call a coalition $S \subset N$ potentially violated if

$$\max_{S \subset N} \left(y(S) - \underline{C(S)} \right) > 0.$$

Observe that a potentially violated coalition $S \subset N$ does not necessarily correspond to a violated rationality constraint. In order to asses this we need to determine the cost $C(S)$ by solving a CVRP.

This approach provides the opportunity to accelerate the procedure of solving CVRPs by adding valid inequalities which we cannot add when solving the row generation subproblem. We add the well-known rounded capacity cuts. For a given subset of customers $W \subseteq V'$ such that $|W| \geq 2$, the rounded capacity cut is as follows:

$$\sum_{(v,w) \in A(N): v \in W, w \notin W} \left[\sum_{r \in R} b_r^{vw} x_r \right] \geq \left\lceil \frac{\sum_{w \in W} d_w}{Q} \right\rceil, \tag{5.2}$$

where b_r^{vw} denotes whether route r uses arc $(v, w) \in A$. This valid inequality puts a lower bound on the number of vehicles required to satisfy the demand of the customers in W. Note that, in contrast to the LM-SRCs, the structure of the pricing problem is not altered by the inclusion of the rounded capacity cuts.

We separate the rounded capacity cuts before separating the LM-SRC using the heuristic separation algorithms by Lysgaard et al. (2004). After adding cuts, we solve the RMP again to

optimality. Furthermore, we initialise the algorithm for the CVRP with the routes used in the optimal solution to the relaxed row generation subproblem. As such, the root node has already been solved and cuts can immediately be separated.

When branching to solve the coarse row generation subproblem, we may come across a solution where the player variables z are integer with a non-negative objective value, before finding the optimal solution to the coarse row generation subproblem. Rather than continuing to branch, we solve the corresponding CVRP and add the corresponding rationality constraint. If the constraint is not violated by the current solution, we continue branching. Otherwise, we solve the relaxed EPM problem to obtain a new allocation or show that the core is empty.

After we add a rationality constraint, we also add an anti-cycling constraint to the coarse row generation subproblem to prevent the algorithm from identifying the same constraint multiple times. This can occur due to the use of the lower bound for the routing cost. Given a coalition $S \subseteq N$, the anti-cycling constraint is as follows:

$$\sum_{i \in S} z_i - \sum_{i \in S \setminus N} z_i \leq |S| - 1. \tag{5.3}$$

Note that the anti-cycling constraints are also added for coalitions $S \subseteq N$ with $|S| = 1$ and the grand-coalition, which are included at initialisation.

5.2 Heuristic row generation

In order to heuristically solve the (coarse) row generation subproblem, we solve the pricing problem by only using the heuristic labelling algorithm as presented in Section 4.3. This leads to an upper bound on the routing cost, and in turn a lower bound on the (coarse) row generation subproblem. Hence, if a (potentially) violated rationality constraint is identified heuristically for the (coarse) row generation subproblem, it is guaranteed to also be (potentially) violated for the (coarse) row generation subproblem. However, as the heuristic gives a lower bound we cannot guarantee that no violated coalition exists if none can be identified by the heuristic. Hence, when the heuristic fails to find a violated coalition, we continue to solve the (coarse) row generation subproblem to optimality.

5.3 Variants of the row generation algorithm

The strategies described in Sections 5.1 and 5.2 can be applied and combined in multiple ways. First, we consider the basic row generation procedure as described in Sections 3 and 4, denoted by GEN_E. Note that this procedure does not include the acceleration methods proposed in Sections 5.1 and 5.2 but does include the column pool as mentioned at the start of Section 5. From now on, we will refer to the row generation subproblem as the regular subproblem to be able to clearly distinguish it from the coarse row generation subproblem, which we will call the coarse subproblem. We consider the row generation procedure GEN_C, which is similar to GEN_E but only performs coarse row separation. Next, we consider the inclusion of heuristics as described in Section 5.2. $\text{GEN}_{CH+EH+E}$ denotes the procedure in which we first solve the coarse subproblem heuristically. If no violated rationality is identified, we solve the regular subproblem heuristically. If by doing so we cannot identify a violated rationality constraint, we solve the regular subproblem to optimality. Finally, we consider the procedure denoted by $\text{GEN}_{CH+EH+C}$ which is similar to $\text{GEN}_{CH+EH+E}$ but where the coarse subproblem is solved to optimality instead of the regular subproblem. Note that all algorithms are guaranteed to give an optimal solution to the EPM if the core is non-empty. A summary of all methods can be found in Table 5.1.

Table 5.1: Solution methods for the row generation subproblem.

Abbreviation	Description
GEN_E	The regular subproblem is solved to optimality.
GEN_C	The coarse subproblem is solved to optimality.
$\text{GEN}_{CH+EH+E}$	First, the coarse subproblem is solved heuristically, then the regular subproblem is solved heuristically and finally the regular subproblem is solved to optimality.
$\text{GEN}_{CH+EH+C}$	First, the coarse subproblem is solved heuristically, then the regular subproblem is solved heuristically and finally the coarse subproblem is solved to optimality.

6 Numerical Results

The algorithms were implemented in C++. We used IBM ILOG CPLEX Optimiser 12.7.1 to solve all linear programming problems. The computations were conducted on a single core of an AMD Ryzen 7 2700X @ 3.7 GHz with 16 GB of RAM.

The arc costs for the instances we consider are symmetric, so we branch on edge flows rather than on arc flows. After each iteration of the (heuristic) pricing problem the best $\rho_E = 10$ ($\rho_H = 20$) negative reduced cost routes are added to the RMP. Furthermore, whenever we solve a CVRP for coalition $S \subset N$ we use the same implementation as used for the regular subproblem by solving GCPTP($\mathbf{0}$) while setting $z_i = 1$ for every $i \in S$ and $z_i = 0$ for $N \backslash S$. Moreover, we now also include the rounded capacity cuts. For every instance and algorithm, a computation time limit of 2 hours is used.

6.1 Problem instances

We generate instances of the JNVRG based on CVRP benchmark instances from the literature. To this end, we consider the A-set of Augerat (1995) and the Solomon instances C101-100 and C201-100 as introduced by Solomon (1987). We select these instances to consider both non-clustered and clustered instances, which could influence the difficulty to determine a core allocation. We modify the instances as follows. We consider 5, 10 and 15 players for each of the instances. Given the number of players n we assign customer i to player $(i \bmod n) + 1$. For the Solomon instances, this creates clusters in which customers of different players are included.

From the Augerat A-set we create JNVRG instances including all customers. However, from each Solomon instance we generate multiple JNVRG instances that contain a subset of customers. For each Solomon instance, we create 5 instances of 20 customers, 3 instances of 33 customers and 2 instances of 50 customers, where we split the customers of the original instance in equal sets starting with the first customer. For example, when considering 33 customers, the first set consists of customers 1 through 33, whereas the second set consists of customers 34 through 66. Additionally, we create instances of the VRG. They are the same as the JNVRG instances, but modified such that each individual customer corresponds to a separate player.

6.2 Numerical results for the JNVRG

We introduce an additional algorithm CORE_E to serve as a benchmark. This algorithm enumerates all coalitions, solves the corresponding CVRPs and then determines an EPM allocation by solving (2.4)-(2.9). Obviously, this method is not tractable when the number of players is high.

The results of our numerical experiments on the JNVRG instances are given in Tables 6.1 and 6.2 for the Augerat instances and Solomon instances, respectively. The first column provides the name of the instance. The column n provides the number of players and the column $|V'|$

provides the number of customers. The column C indicates whether the core is empty (E), non-empty (NE) or whether it is unknown (?) because the algorithm reached the time limit of 2 hours. For each of the algorithms the number of generated constraints is reported in the column m, where the number in between brackets represents the number of generated constraints which were actually violated, excluding the grand-coalition and singleton rationality constraints used at initialisation. The column T provides the total time spent in seconds to determine an EPM allocation, and the column T_{RG} represents the percentage of time which was spent on solving the row generation subproblem. For each instance, the fastest solution time has been made bold. Note that the Augerat instances A-n53-k07 to A-n80-k10 have been omitted from Table 6.1 as not many of these instances could be solved within a time limit of two hours. The results on these instances are presented in Table A.1 in the Appendix and are not considered in the remainder of this section. Note that among these instances, the largest we could solve has 5 players and 79 customers.

Using the enumerative algorithm CORE_E, 23 out of the 52 Augerat instances and 27 out of the 60 Solomon instances for the JNVRG were solved. By using this algorithm, almost all instances consisting of 5 players could be solved. However, only 19 out of 34 instances consisting of 10 players could be solved and no instance consisting of 15 players could be solved within the time limit. This clearly demonstrates that enumerating all coalitions is not tractable even for a moderate number of players. With the different variants of the row generation algorithms we are able to solve significantly more instances. Using the basic row generation algorithm GEN_E we could solve a total of 77 instances, 27 more than with the enumerative algorithm, which demonstrates the effectiveness of the row generation procedure. The other variants of the algorithms GEN_C, $\text{GEN}_{CH+EH+E}$ and $\text{GEN}_{CH+EH+C}$ could solve 81, 89 and 88 instances out of the 112 instances, respectively. Note that the row generation algorithms are able to solve significantly more of both the 10 player and 15 player instances.

Table 6.1: Numerical results on the JNVRG instances of the Augerat A-set.

				CORE_E	GEN_E			GEN_C			$\text{GEN}_{CH+EH+E}$			$\text{GEN}_{CH+EH+C}$		
Instance	n	$\lvert V'\rvert$	C	T	m	T	T_I	m	T	T_I	m	T	T_I	m	T	T_I
A-n32-k05	5	31	NE	30	2 (2)	1014	99	2 (2)	32	74	2 (2)	**23**	62	2 (2)	23	62
A-n32-k05	10	31	NE	1239	4 (4)	1867	100	20 (5)	628	86	16 (5)	**197**	70	20 (5)	338	80
A-n32-k05	15	31	NE	7200		7200			7200		117 (20)	**2191**	74	120 (20)	3058	80
A-n33-k06	5	32	NE	45	0 (0)	22	75	1 (0)	20	66	1 (0)	**15**	55	1 (0)	15	56
A-n33-k06	10	32	NE	1702	5 (5)	818	99	20 (8)	870	56	16 (7)	**676**	21	17 (8)	685	29
A-n33-k06	15	32	NE	7200	13 (13)	2009	100	45 (16)	2257	90	46 (20)	**531**	73	54 (20)	795	78
A-n34-k05	5	33	NE	179	0 (0)	41	85	2 (0)	41	72	2 (0)	**23**	54	2 (0)	23	54
A-n34-k05	10	33	E	1606	11 (11)	3963	100	12 (7)	449	42	12 (9)	**71**	5	12 (9)	71	5
A-n34-k05	15	33	E	7200	14 (14)	2585	100	27 (22)	1305	76	26 (26)	**145**	11	26 (26)	146	11
A-n36-k05	5	35	NE	149	2 (2)	1017	99	2 (2)	100	30	2 (2)	**77**	37	2 (2)	80	36
A-n36-k05	10	35	NE	4038	7 (7)	1786	99	17 (8)	840	74	17 (7)	743	73	16 (7)	**585**	62
A-n36-k05	15	35	?	7200		7200			7200			7200			7200	
A-n37-k05	5	36	NE	71	0 (0)	34	72	0 (0)	34	72	0 (0)	**32**	71	0 (0)	33	71
A-n37-k05	10	36	NE	5937	1 (1)	2244	100	5 (2)	332	88	5 (2)	**176**	73	5 (2)	178	73
A-n37-k05	15	36	NE	7200		7200			7200		48 (15)	**2280**	47	55 (15)	4581	36
A-n37-k06	5	36	NE	496	0 (0)	475	3	0 (0)	**473**	3	0 (0)	476	3	0 (0)	478	3
A-n37-k06	10	36	NE	6048	5 (5)	1516	80	14 (5)	1556	34	14 (5)	1216	22	14 (5)	**1124**	30
A-n37-k06	15	36	NE	7200		7200			7200		108 (24)	6317	40	94 (21)	**5859**	59
A-n38-k05	5	37	NE	265	0 (0)	238	16	1 (0)	**228**	11	0 (0)	235	15	1 (0)	230	11
A-n38-k05	10	37	NE	2212		7200		15 (3)	915	66	11 (4)	**461**	36	15 (4)	504	38
A-n38-k05	15	37	?	7200		7200			7200			7200			7200	
A-n39-k05	5	38	NE	441	0 (0)	**257**	13	1 (0)	265	12	0 (0)	259	13	1 (0)	267	12
A-n39-k05	10	38	NE	7200		7200			7200		10 (7)	**1423**	21		7200	
A-n39-k05	15	38	?	7200		7200			7200			7200			7200	
A-n39-k06	5	38	NE	268	0 (0)	**149**	23	2 (0)	219	20	0 (0)	153	25	2 (0)	219	20
A-n39-k06	10	38	NE	2916	6 (6)	1539	91	18 (7)	1053	48	17 (8)	**691**	27	19 (8)	837	37
A-n39-k06	15	38	?	7200		7200			7200			7200			7200	
A-n44-k06	5	43	NE	308	0 (0)	34	64	0 (0)	35	64	0 (0)	**34**	64	0 (0)	35	64
A-n44-k06	10	43	NE	7200	5 (5)	2927	100	30 (7)	4440	58	27 (4)	**2306**	33	30 (8)	2797	35
A-n44-k06	15	43	NE	7200		7200			7200		53 (16)	**4608**	47	51 (14)	6278	54
A-n45-k06	5	44	NE	207	0 (0)	94	32	1 (0)	107	34	0 (0)	**93**	31	1 (0)	106	34
A-n45-k06	10	44	NE	5459	2 (2)	2094	97	15 (2)	1661	69	3 (2)	**614**	87	15 (2)	1131	41
A-n45-k06	15	44	?	7200		7200			7200			7200			7200	
A-n45-k07	5	44	NE	322	1 (1)	819	88	1 (1)	**155**	32	1 (1)	820	88	1 (1)	156	32
A-n45-k07	10	44	NE	7200		7200			7200		43 (13)	6293	19	44 (13)	**5823**	23
A-n45-k07	15	44	?	7200		7200			7200			7200			7200	
A-n46-k07	5	45	NE	143		7200		1 (1)	77	75	1 (1)	**68**	65	1 (1)	68	66
A-n46-k07	10	45	NE	7200		7200		20 (7)	2354	91	17 (7)	**956**	78	18 (7)	986	77
A-n46-k07	15	45	?	7200		7200			7200			7200			7200	
A-n48-k07	5	47	NE	828	0 (0)	307	91	2 (0)	177	50	0 (0)	305	91	2 (0)	**154**	43
A-n48-k07	10	47	NE	7200	2 (2)	6569	100		7200		19 (3)	**4804**	30	22 (3)	5740	20
A-n48-k07	15	47	?	7200		7200			7200			7200			7200	

Table 6.2: Results on the JNVRG Solomon instances.

				CORE_E		GEN_E			GEN_C			$\text{GEN}_{CH+EH+E}$			$\text{GEN}_{CH+EH+C}$	
Instance	n	$\lvert V'\rvert$	C	T	m	T	T_I	m	T	T_I	m	T	T_I	m	T	T_I
C101-0	5	20	NE	19	0 (0)	13	66	0 (0)	13	66	0 (0)	13	66	0 (0)	**13**	66
C101-1	5	20	NE	**9**	1 (1)	11	77	2 (1)	14	76	1 (1)	9	66	2 (1)	12	71
C101-2	5	20	NE	28	0 (0)	**21**	53	0 (0)	21	53	0 (0)	21	53	0 (0)	21	53
C101-3	5	20	NE	25	0 (0)	14	38	0 (0)	14	38	0 (0)	14	38	0 (0)	**13**	38
C101-4	5	20	NE	10	0 (0)	5	64	0 (0)	5	64	0 (0)	5	64	0 (0)	**5**	64
C101-0	10	20	NE	448	1 (1)	47	85	1 (1)	53	75	1 (1)	**37**	80	1 (1)	42	69
C101-1	10	20	E	252	8 (8)	81	98	10 (10)	39	81	13 (13)	11	11	13 (13)	**11**	11
C101-2	10	20	NE	593	0 (0)	**24**	27	0 (0)	24	27	0 (0)	24	27	0 (0)	24	27
C101-3	10	20	NE	617	2 (2)	77	91	12 (4)	148	77	5 (2)	**58**	73	13 (4)	92	59
C101-4	10	20	NE	250	1 (1)	**9**	80	1 (1)	11	76	1 (1)	10	73	1 (1)	10	73
C101-0	15	20	NE	7200	1 (1)	**147**	98	3 (1)	286	95	1 (1)	160	98	3 (1)	195	88
C101-1	15	20	E	7200	9 (9)	123	98	16 (16)	64	82	14 (14)	**11**	13	14 (14)	11	13
C101-2	15	20	NE	7200	6 (6)	1966	99	13 (9)	2786	98	13 (9)	**980**	94	13 (9)	1166	96
C101-3	15	20	NE	7200	2 (2)	201	98	67 (3)	1369	88	19 (2)	**193**	69	67 (3)	507	62
C101-4	15	20	NE	7200	1 (1)	**28**	95	1 (1)	32	93	1 (1)	32	91	1 (1)	31	91
C101-0	5	33	NE	**100**	0 (0)	135	69	1 (0)	151	64	0 (0)	135	68	1 (0)	145	62
C101-1	5	33	NE	116	0 (0)	27	26	0 (0)	**27**	27	0 (0)	27	27	0 (0)	27	26
C101-2	5	33	NE	56	0 (0)	20	57	0 (0)	**20**	57	0 (0)	21	57	0 (0)	20	57
C101-0	10	33	E	7200		7200		13 (9)	5437	14	12 (10)	4370	11	11 (10)	**2677**	13
C101-1	10	33	NE	7200	0 (0)	**111**	77	6 (0)	198	72	0 (0)	112	77	6 (0)	173	68
C101-2	10	33	NE	1322	1 (1)	**97**	90	2 (1)	111	82	1 (1)	103	90	2 (1)	106	81
C101-0	15	33	E	7200		7200		14 (13)	1397	48	10 (10)	1115	18	13 (13)	**812**	27
C101-1	15	33	NE	7200	3 (3)	438	97	19 (3)	633	78	12 (3)	**283**	73	19 (3)	427	67
C101-2	15	33	NE	7200	3 (3)	**400**	98	6 (4)	513	92	7 (5)	478	93	8 (5)	437	88
C101-0	5	50	NE	549	0 (0)	**304**	54	0 (0)	305	54	0 (0)	307	54	0 (0)	305	54
C101-1	5	50	NE	368	0 (0)	269	20	0 (0)	269	20	0 (0)	270	20	0 (0)	**268**	20
C101-0	10	50	NE	7200	0 (0)	**232**	36	0 (0)	233	36	0 (0)	234	36	0 (0)	233	36
C101-1	10	50	NE	6588	2 (2)	1233	83	10 (2)	1501	67	10 (2)	1205	64	10 (2)	**1148**	58
C101-0	15	50	NE	7200	1 (1)	3519	96	1 (1)	3501	96	1 (1)	3249	96	1 (1)	**3147**	95
C101-1	15	50	NE	7200		7200			7200		30 (6)	**3013**	66	40 (8)	3408	65
C201-0	5	20	NE	161	0 (0)	54	42	0 (0)	**54**	42	0 (0)	54	42	0 (0)	54	42
C201-1	5	20	NE	64	1 (1)	**58**	43	1 (1)	69	33	1 (1)	61	45	1 (1)	70	34
C201-2	5	20	NE	134	0 (0)	39	61	0 (0)	**39**	61	0 (0)	39	61	0 (0)	39	61
C201-3	5	20	NE	204	0 (0)	65	37	0 (0)	**65**	38	0 (0)	65	38	0 (0)	65	38
C201-4	5	20	NE	115	0 (0)	91	27	0 (0)	**91**	27	0 (0)	92	27	0 (0)	91	27
C201-0	10	20	NE	1391	2 (2)	**81**	70	2 (2)	1080	3	2 (2)	1065	2	2 (2)	1072	2
C201-1	10	20	NE	1146	1 (1)	**46**	31	1 (1)	68	17	2 (2)	63	23	2 (2)	63	23
C201-2	10	20	NE	7200	0 (0)	47	55	0 (0)	**47**	55	0 (0)	47	55	0 (0)	47	55
C201-3	10	20	NE	7200	1 (1)	66	27	1 (1)	102	19	1 (1)	**66**	27	1 (1)	101	18
C201-4	10	20	NE	4351	0 (0)	60	22	0 (0)	60	22	0 (0)	**60**	23	0 (0)	60	22
C201-0	15	20	NE	7200	3 (3)	**161**	55	3 (3)	258	42	4 (4)	193	56	4 (4)	251	37
C201-1	15	20	NE	7200	1 (1)	**75**	79	1 (1)	106	67	1 (1)	121	55	1 (1)	121	55
C201-2	15	20	NE	7200	0 (0)	26	44	0 (0)	**25**	44	0 (0)	26	44	0 (0)	26	43
C201-3	15	20	NE	7200	1 (1)	54	51	1 (1)	145	19	1 (1)	**54**	52	1 (1)	145	19
C201-4	15	20	NE	7200	1 (1)	208	56	1 (1)	210	55	1 (1)	**180**	49	1 (1)	183	48
C201-0	5	33	NE	4620	0 (0)	**763**	32	0 (0)	767	31	0 (0)	766	31	0 (0)	765	31
C201-1	5	33	NE	7200	0 (0)	**855**	33	0 (0)	856	33	0 (0)	856	33	0 (0)	857	33
C201-2	5	33	NE	3321	0 (0)	2777	42	0 (0)	2810	41	0 (0)	2803	41	0 (0)	**2773**	42
C201-0	10	33	NE	7200	0 (0)	672	27	0 (0)	675	27	0 (0)	675	27	0 (0)	**672**	27
C201-1	10	33	NE	7200	0 (0)	1393	85	0 (0)	1398	85	0 (0)	1399	85	0 (0)	**1390**	85
C201-2	10	33	NE	7200	0 (0)	**2420**	77	0 (0)	2463	77	0 (0)	2454	77	0 (0)	2431	77
C201-0	15	33	NE	7200	2 (2)	**400**	20	2 (2)	1914	4	2 (2)	483	34	2 (2)	1912	5
C201-1	15	33	NE	7200	1 (1)	**1843**	70	1 (1)	6094	21	1 (1)	1884	70	1 (1)	5985	21
C201-2	15	33	NE	7200	0 (0)	648	30	0 (0)	651	30	0 (0)	651	30	0 (0)	**648**	30
C201-0	5	50	?	7200		7200			7200			7200			7200	
C201-1	5	50	NE	7200	0 (0)	3613	41	0 (0)	3626	41	0 (0)	3627	41	0 (0)	**3609**	41
C201-0	10	50	?	7200		7200			7200			7200			7200	
C201-1	10	50	?	7200		7200			7200			7200			7200	
C201-0	15	50	?	7200		7200			7200			7200			7200	
C201-1	15	50	?	7200		7200			7200			7200			7200	

When comparing run times between two algorithms, we compare the total time taken to solve all instances which have been solved using both algorithms. When considering the 5 player instances, the row generation algorithm GEN_E, GEN_C, $\text{GEN}_{CH+EH+E}$ and $\text{GEN}_{CH+EH+C}$ are 47%, 104%, 87% and 108% faster than CORE_E, respectively. On the 10 player instances, the row generation algorithms are 161% to 543% faster than the enumerative algorithm. For all Augerat instances, GEN_C was 61% faster than GEN_E, which shows the effectiveness of the coarse row generation. Furthermore, $\text{GEN}_{CH+EH+E}$ was 134% faster whereas $\text{GEN}_{CH+EH+C}$ was 112% faster than GEN_E on all Augerat instances, which demonstrates the effectiveness of solving the (coarse) subproblem problem heuristically. However, on the Solomon instances, GEN_E was 38% and 23% faster than GEN_C and $\text{GEN}_{CH+EH+C}$, respectively, and only 1% slower than $\text{GEN}_{CH+EH+E}$. This result can be explained by the low number of rationality constraints which are generated in the Solomon instances in comparison to the Augerat instances. This low number of generated rationality constraints can be attributed to the structure of the Solomon instances. These instances contain clusters consisting of customers of different players, making a

collaboration highly profitable, and as a result, any efficient and individually rational allocation is likely to lie in the core.

In general, the row generation algorithms $\text{GEN}_{CH+EH+E}$ and $\text{GEN}_{CH+EH+C}$ solved the most instances in the least amount of time. Comparing both algorithms on the 5 player instances solved by both algorithms, $\text{GEN}_{CH+EH+C}$ was on average 7% faster. On the 10 players instances both algorithms were equally fast. Finally, on the 15 player instances, $\text{GEN}_{CH+EH+E}$ was on average 38% faster.

6.3 Numerical Results for the VRG

The results of our numerical experiments for the VRG instances are given in Tables 6.3 and 6.4 for the Augerat instances and Solomon instances, respectively. Note that CORE_E is not included in the results as the algorithm is no longer tractable for these high number of players.

When considering the Augerat instances, using the row generation algorithms GEN_E, GEN_C, $\text{GEN}_{CH+EH+E}$ and $\text{GEN}_{CH+EH+C}$ we could solve a total of 6, 10, 14 and 16 out of 17 instances, respectively. Moreover, $\text{GEN}_{CH+EH+E}$ and $\text{GEN}_{CH+EH+C}$ were on these instances 1714% and 1728% faster than GEN_E, respectively. Again these results illustrate the effectiveness of the coarse row generation as well as heuristic row generation, also for the VRG. We find similar results for the Solomon instances, using the row generation algorithms GEN_E, GEN_C, $\text{GEN}_{CH+EH+E}$ and $\text{GEN}_{CH+EH+C}$ we could solve a total of 12, 12, 14 and 13 out of 20 instances, respectively. We were not able to solve any of the instances based on the Solomon instance C201 with 33 players or more. This may be attributed to the high vehicle capacity for these instances, when compared to the customer demand. As a result, most customers can be visited on a single distribution route. Our column generation algorithm is not suitable for these instances, as the computation time of the labelling algorithm used to solve the pricing problem becomes high in the case of many customers on one route.

Table 6.3: The results on given instances for the different varieties of the row generation algorithm.

				GEN_E			GEN_C			$\text{GEN}_{CH+EH+E}$			$\text{GEN}_{CH+EH+C}$		
Instance	n	$\lvert V' \rvert$	C	m	T	T_I	m	T	T_I	m	T	T_I	m	T	T_I
A-n32-k05	31	31	E	54 (54)	1239	100	42 (42)	281	86	52 (52)	**50**	7	52 (52)	50	7
A-n33-k06	32	32	E	71 (71)	1410	99	64 (64)	330	86	68 (68)	49	8	68 (68)	**49**	8
A-n34-k05	33	33	E	41 (41)	2721	100	52 (52)	662	94	42 (41)	**39**	15	47 (47)	43	16
A-n36-k05	35	35	E		7200		60 (60)	1691	93	85 (85)	227	6	85 (85)	**222**	6
A-n37-k05	36	36	E	51 (51)	3660	100	59 (59)	1010	89	56 (56)	118	6	56 (56)	**114**	6
A-n37-k06	36	36	E		7200		69 (69)	3717	83		7200		119 (119)	**935**	32
A-n38-k05	37	37	E	45 (45)	1954	90	44 (44)	983	74	52 (52)	254	3	52 (52)	**250**	3
A-n39-k05	38	38	E		7200		48 (48)	1734	75	48 (48)	321	6	48 (48)	**316**	7
A-n39-k06	38	38	E	54 (54)	5232	95	48 (48)	1114	70	76 (76)	384	2	76 (76)	**381**	2
A-n44-k06	43	43	E		7200			7200		171 (171)	**319**	13	185 (185)	344	15
A-n45-k06	44	44	E		7200		91 (91)	4329	94	70 (70)	224	4	70 (70)	**216**	4
A-n45-k07	44	44	E		7200			7200			7200		149 (149)	**1481**	65
A-n46-k07	45	45	E		7200			7200		225 (225)	508	9	225 (225)	**490**	9
A-n48-k07	47	47	E		7200			7200		267 (267)	**612**	9	278 (278)	643	12
A-n53-k07	52	52	E		7200			7200		225 (223)	2762	26	225 (223)	**2709**	26
A-n54-k07	53	53	E		7200			7200		165 (165)	2978	32	174 (174)	**2460**	17

Table 6.4: Numerical Results on the VRG solomon instances.

				GEN_E			GEN_C			$\text{GEN}_{CH+EH+E}$			$\text{GEN}_{CH+EH+C}$		
Instance	n	$\|V'\|$	C	m	T	T_I	m	T	T_I	m	T	T_I	m	T	T_I
C101-0	20	20	NE	99 (99)	1102	99	80 (80)	343	89	52 (52)	**66**	37	37 (37)	95	78
C101-1	20	20	E	21 (21)	168	99	15 (15)	26	73	16 (16)	10	7	16 (16)	**9**	7
C101-2	20	20	NE	16 (16)	696	98	17 (17)	627	93	16 (16)	**347**	88	15 (15)	406	89
C101-3	20	20	E	64 (64)	193	96	66 (66)	70	63	63 (59)	**37**	14	60 (59)	51	15
C101-4	20	20	NE	4 (4)	6	73	4 (4)	10	58	3 (3)	**5**	34	3 (3)	5	32
C101-0	33	33	E	12 (12)	1053	95	14 (14)	379	57	27 (27)	263	42	19 (19)	**188**	14
C101-1	33	33	E		7200			7200		50 (49)	**253**	6	50 (49)	271	5
C101-2	33	33	E	92 (92)	1575	99	80 (80)	1047	62	126 (123)	664	24	136 (135)	**486**	10
C101-0	50	50	?		7200			7200			7200			7200	
C101-1	50	50	E		7200			7200		96 (96)	1489	-000	96 (96)	**1489**	-000
C201-0	20	20	NE	7 (7)	**255**	91	7 (7)	562	50	8 (8)	274	69	8 (8)	338	58
C201-1	20	20	NE	5 (5)	125	74	6 (6)	207	37	5 (5)	**104**	35	5 (5)	133	21
C201-2	20	20	NE	1 (1)	**54**	33	1 (1)	89	18	1 (1)	54	33	1 (1)	89	18
C201-3	20	20	NE	1 (1)	76	79	3 (3)	212	31	1 (1)	**75**	79		7200	
C201-4	20	20	NE	2 (2)	443	91	3 (3)	474	88	3 (3)	399	86	3 (3)	**364**	83
C201-0	33	33	?		7200			7200			7200			7200	
C201-1	33	33	?		7200			7200			7200			7200	
C201-2	33	33	?		7200			7200			7200			7200	
C201-0	50	50	?		7200			7200			7200			7200	
C201-1	50	50	?		7200			7200			7200			7200	

7 Concluding remarks

We presented a row generation algorithm to allocate costs for instances of the JNVRG with a large number of players. Furthermore, we proposed coarse and heuristic row generation to accelerate our algorithm. The results of our numerical experiments suggest that the proposed row generation algorithm is effective for the JNVRG as well as for the VRG, as we can determine core allocations for instances ranging from 5 players and 79 customers to instances of 53 players and 53 customers, within 2 hours of computation time.

In practice, companies may have more delivery locations than we can currently solve with our algorithm. It requires further research to design a methodology for these cases. Furthermore, observe that our algorithm applies to settings in which the depots of the companies are the same or located near each other. However, in practice this might not be the case. Similarly, time windows or other case specific features could impact the routing cost. Further research is required to determine the effectiveness of row generation for these cases.

References

D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The traveling salesman problem: a computational study.* Princeton university press, 2006.

C. Archetti, N. Bianchessi, and M. Speranza. Optimal solutions for routing problems with profits. *Discrete Applied Mathematics*, 161(4):547 – 557, 2013. Seventh International Conference on Graphs and Optimization 2010.

J. Arin. Egalitarian distributions in coalitional models: The lorenz criterion. IKERLANAK 2003-02, Universidad del País Vasco - Departamento de Fundamentos del Análisis Económico I, 2003.

P. Augerat. *Approche polyédrale du problème de tournées de véhicules.* Institut national polytechnique, 1995.

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

E. Ballot and F. Fontane. Reducing transportation co_2 emissions through pooling of supply networks: perspectives from a case study in french retail chains. *Production Planning & Control*, 21(6):640–650, 2010.

O. N. Bondareva. Some applications of linear programming methods to the theory of cooperative games. *Problemy kibernetiki*, 10:119–139, 1963.

B. Dai and H. Chen. Proportional egalitarian core solution for profit allocation games with an application to collaborative transportation planning. *European Journal of Industrial Engineering*, 9(1):53–76, 2015.

M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.*, 40(2):342–354, Mar. 1992.

M. Dror. Cost allocation: the traveling salesman, binpacking, and the knapsack. *Applied Mathematics and Computation*, 35(2):191 – 207, 1990.

S. Engevall, M. Göthe-Lundgren, and P. Värbrand. The traveling salesman game: An application ofcost allocation in a gas and oil company. *Annals of Operations Research*, 82(0):203–218, 1998.

S. Engevall, M. Göthe-Lundgren, and P. Värbrand. The heterogeneous vehicle-routing game. *Transportation Science*, 38(1):71–85, 2004.

U. Faigle. Cores of games with restricted cooperation. *Zeitschrift für Operations Research*, 33 (6):405–422, Nov 1989.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

P. C. Fishburn and H. O. Pollak. Fixed-route cost allocation. *The American Mathematical Monthly*, 90(6):366–378, 1983.

M. Frisk, M. Göthe-Lundgren, K. Jörnsten, and M. Rönnqvist. Cost allocation in collaborative forest transportation. *European Journal of Operational Research*, 205(2):448 – 458, 2010.

D. B. Gillies. *Solutions to general non-zero-sum games.* Number 40 in Annals of Mathematics Studies. Princeton University Press, Princeton, 1959.

M. Göthe-Lundgren, K. Jörnsten, and P. Värbrand. On the nucleolus of the basic vehicle routing game. *Mathematical Programming*, 72(1):83–100, 1996.

M. Guajardo and M. Rönnqvist. A review on cost allocation methods in collaborative transportation. *International Transactions in Operational Research*, 23(3):371–392, 5 2016.

M. A. Krajewska, H. Kopfer, G. Laporte, S. Ropke, and G. Zaccour. Horizontal cooperation among freight carriers: Request allocation and profit sharing. *The Journal of the Operational Research Society*, 59(11):1483–1491, 2008.

N. Lehoux, S. D'Amours, Y. Frein, A. Langevin, and B. Penz. Collaboration for a two-echelon supply chain in the pulp and paper industry: the use of incentives to increase profit. *Journal of the Operational Research Society*, 62(4):581–592, Apr 2011.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, Jun 2004.

R. Martinelli, D. Pecin, and M. Poggi. Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research*, 239(1):102 – 111, 2014.

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

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

J. A. M. Potters, I. J. Curiel, and S. H. Tijs. Traveling salesman games. *Mathematical Programming*, 53(1):199–211, 1992.

D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 17(6):1163–1170, 1969.

L. S. Shapley. A value for n-person games. *Contributions to the theory of games*, 2:307–317, 1953.

L. S. Shapley. On balanced sets and cores. *Naval Research Logistics Quarterly*, 14(4):453–460, 1967.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

P. Toth and D. Vigo. *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

V. V. Zakharov and A. N. Shchegryaev. Stable cooperation in dynamic vehicle routing problems. *Automation and Remote Control*, 76(5):935–943, May 2015.

Appendix

A Remaining results on the Augerat instances

Table A.1: Results on the remaining Augerat instances.

				CORE_E		GEN_E			GEN_C			$\text{GEN}_{CH+EH+E}$			$\text{GEN}_{CH+EH+C}$	
Instance	n	$\lvert V'\rvert$	C	T	m	T	T_I	m	T	T_I	m	T	T_I	m	T	T_I
A-n53-k07	5	52	NE	3546	0 (0)	1368	48	3 (0)	2223	9	0 (0)	**1291**	44	3 (0)	1586	9
A-n53-k07	10	52	?	7200		7200			7200			7200			7200	
A-n53-k07	15	52	?	7200		7200			7200			7200			7200	
A-n54-k07	5	53	NE	4410	0 (0)	**1082**	11	1 (0)	1102	10	0 (0)	1084	11	1 (0)	1108	9
A-n54-k07	10	53	?	7200		7200			7200			7200			7200	
A-n54-k07	15	53	?	7200		7200			7200			7200			7200	
A-n55-k09	5	54	NE	1161	0 (0)	862	10	1 (0)	**859**	9	0 (0)	861	10	1 (0)	868	9
A-n55-k09	10	54	NE	7200		7200		14 (5)	4425	69	13 (5)	**2620**	47	13 (5)	2881	38
A-n55-k09	15	54	?	7200		7200			7200			7200			7200	
A-n55-k09	54	54	?			7200			7200			7200			7200	
A-n60-k09	5	59	NE	7200	0 (0)	5543	28	2 (0)	4383	5	0 (0)	5574	28	2 (0)	**4358**	4
A-n60-k09	10	59	?	7200		7200			7200			7200			7200	
A-n60-k09	15	59	?	7200		7200			7200			7200			7200	
A-n60-k09	59	59	?			7200			7200			7200			7200	
A-n61-k09	5	60	?	7200		7200			7200			7200			7200	
A-n61-k09	10	60	?	7200		7200			7200			7200			7200	
A-n61-k09	15	60	?	7200		7200			7200			7200			7200	
A-n61-k09	60	60	?			7200			7200			7200			7200	
A-n62-k08	5	61	NE	7200	0 (0)	**4123**	3		7200		0 (0)	4150	3		7200	
A-n62-k08	10	61	?	7200		7200			7200			7200			7200	
A-n62-k08	15	61	?	7200		7200			7200			7200			7200	
A-n62-k08	61	61	?			7200			7200			7200			7200	
A-n63-k09	5	62	NE	7200	0 (0)	768	54	1 (0)	886	14	0 (0)	**766**	54	1 (0)	887	14
A-n63-k09	10	62	?	7200		7200			7200			7200			7200	
A-n63-k09	15	62	?	7200		7200			7200			7200			7200	
A-n63-k09	62	62	?			7200			7200			7200			7200	
A-n63-k10	5	62	?	7200		7200			7200			7200			7200	
A-n63-k10	10	62	?	7200		7200			7200			7200			7200	
A-n63-k10	15	62	?	7200		7200			7200			7200			7200	
A-n63-k10	62	62	?			7200			7200			7200			7200	
A-n64-k09	5	63	?	7200		7200			7200			7200			7200	
A-n64-k09	10	63	?	7200		7200			7200			7200			7200	
A-n64-k09	15	63	?	7200		7200			7200			7200			7200	
A-n65-k09	5	64	NE	3592		7200		3 (1)	787	31		7200		3 (1)	**746**	25
A-n65-k09	10	64	NE	7200		7200		11 (2)	**4304**	41		7200		11 (2)	4995	21
A-n65-k09	15	64	?	7200		7200			7200			7200			7200	
A-n69-k09	5	68	NE	7200	0 (0)	**4864**	3	1 (0)	7157	3	0 (0)	4934	3		7200	
A-n69-k09	10	68	?	7200		7200			7200			7200			7200	
A-n69-k09	15	68	?	7200		7200			7200			7200			7200	
A-n80-k10	5	79	NE	7200	0 (0)	**6586**	7		7200		0 (0)	6658	6		7200	
A-n80-k10	10	79	?	7200		7200			7200			7200			7200	
A-n80-k10	15	79	?	7200		7200			7200			7200			7200	