Computers & Industrial Engineering 172 (2022) 108583

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Nucleolus based cost allocation methods for a class of constrained lane covering games

Nihat Öner [a], Gültekin Kuyzu [a,b,*]

[a] Department of Industrial Engineering, TOBB University of Economics and Technology, Ankara, Turkey
[b] Convoy Inc., Seattle, WA, United States

ARTICLE INFO

Keywords:
Collaborative logistics
Column and row generation
Cost allocation
Cooperative games
Nucleolus

ABSTRACT

In truckload transportation procurement networks, shippers purchasing the services of truckload carriers seek to form multi-company tours which consist of regularly scheduled shipments with minimal empty truck movements with the aim of getting better rates from the carriers in return. Identifying the minimum cost collaborative solution and sharing the costs in a fair manner are two interrelated and critical tasks for the success of such networks. The restrictions on the tours which can be formed make both of these tasks very challenging. In this paper, we take a cooperative game theory approach and study the resulting constrained lane covering game. We first show the set of conditions which must be satisfied for the game to have a non-empty core. We propose nucleolus based cost allocation methods. We also develop column and row generation methods for solving these mathematical models. We compare our proposed cost allocation models with alternative cost allocation methods and each other using a set of core stability metrics through computational experiments on randomly generated instances from the literature.

1. Introduction

Collaboration has emerged as a promising new strategy in logistics, since it takes a broader system-wide view which provides opportunities for increased efficiency that are impossible to achieve with an internal focus. Collaborative approaches in logistics aim to minimize the inefficiencies in transportation systems. The main source of inefficiency for truckload carriers is the empty repositioning movements of trucks between loads. In collaborative truckload transportation procurement (CTTP), a group of shippers purchasing the services of truckload carriers seeks to form regularly scheduled tours with minimal empty truck movements in the hope of getting better rates from the carriers in return.

Establishing a successful logistics collaboration involves three primary tasks: (i) selecting participants; (ii) identifying and exploiting synergies between participants; and (iii) sharing of the resulting gains among the collaborators. Synergy is typically measured by total cost savings achieved by the collaboration, so one of the tasks is identifying the minimum-cost system-wide solution which amounts to solving an optimization problem. The system-wide solution could be used to determine which participant should be included in the collaboration. The system-wide cost must be allocated to each participant, or even each activity, to determine the final gain of each participant. The system-wide cost must be distributed in a fair manner for sustained collaboration among partners.

The state-of-the-art cost allocation mechanisms proposed for collaborative logistics typically utilizes concepts from cooperative game theory. The focus is on identifying cost allocations which are *stable* (or *group-strategy proof*) and *budget-balanced* (or *efficient*). A stable cost allocation guarantees that no subset of the members of the coalition is better off by breaking away and forming a coalition on their own. A budget-balanced (or *efficient*) cost allocation distributes the total cost to the members of the coalition without creating any budget deficit or surplus. A cost allocation which is budget-balanced and stable is said to be in the *core*. In other words, the core of a cooperative game is the set of budget-balanced and stable cost allocations. The core of a cooperative game may be empty.

More formally, a cooperative game in characteristic function form is defined by the pair (N, F) where N is the set of players and $F : 2^N \to \mathbb{R}$ is the characteristic function. $F(S)$ gives the (optimal) cost of the subset of players S. A cost allocation vector $w = (w_1, w_2, \dots, w_{|N|})$, indexed by the set of players, is said to be in the core if and only if the following conditions are satisfied: $\sum_{i\in N} w_i = F(N)$ and $\sum_{i\in S} w_i \le F(S)\ \forall S \subset N$.

One of the most widely known cost allocation methods in cooperative game theory is the Shapley value (Shapley, 1953). The Shapley

* Corresponding author at: Department of Industrial Engineering, TOBB University of Economics and Technology, Ankara, Turkey.
E-mail address: gkuyzu@etu.edu.tr (G. Kuyzu).

https://doi.org/10.1016/j.cie.2022.108583
Received 3 February 2020; Received in revised form 29 December 2021; Accepted 14 August 2022
Available online 19 August 2022

value for a player is the weighted average of the player's marginal contribution to each subset of the grand coalition. It can be interpreted as the average contribution of each player if the grand coalition was built by adding one player at a time (Young, 1985). We refer the reader to Section 8.2 for a mathematical definition of the Shapley value. Unfortunately, the cost allocation obtained by the Shapley value may not be in the core even if the core is non-empty. Another major drawback of using the Shapley value in collaborative logistics is that it requires solving an exponential number of optimization problems, which tend to be NP-Hard, when calculating the marginal contribution of each customer to each subset.

The nucleolus (Schmeidler, 1969) is another well-known cost allocation concept. The nucleolus is the unique cost allocation that lexicographically maximizes the minimal gain over all of the subsets of the collaboration. We present a rigorous description of the nucleolus and its computation in Section 5. The nucleolus is in the core if the core is non-empty. It is considered to be more desirable than others when there are multiple cost allocations in the core. The nucleolus exists even when the core is empty. Similar to the Shapley value, identifying the nucleolus requires heavy computational effort.

In many collaborative situations, the cost of a coalition is usually calculated by solving a covering type integer program. The classic Bondareva–Shapley theorem (Bondareva, 1963; Shapley, 1967) implies that for covering games in which the cost of a coalition (or subcoalition) is given by the minimum cost solution to a covering integer program, the core is nonempty if and only if the linear relaxation of the cost calculation IP has no integrality gap (Pál & Tardos, 2003). Thus, constructing the coalition in a way to maximize the total savings will result in an unstable coalition in most cases of logistics collaborations.

In CTTP, the minimum cost system-wide solution is determined by solving a lane covering problem (LCP), which is formally defined as the problem of identifying a set of cycles covering a specially designated subset of the arcs (corresponding to regularly scheduled truckload movements) of a directed graph, called lanes, with minimum total cost. The base (unconstrained) LCP, i.e. without any constraints on the (simple) cycles which can be used in the cover, can be solved in polynomial time as a minimum cost flow circulation problem or a bipartite matching problem. In practice, one or more constraints could be imposed on the cycles, e.g. maximum number of lanes or maximum duration. The LCP variants with such constraints on the cycles turn out to be NP-Hard (Ergun et al., 2007a, 2007b; Kuyzu, 2017).

The cost allocation problem arising in the CTTP setting can be formulated as a cooperative game, i.e. a lane covering game (LCG) in which the operational cost of each possible coalition is calculated by solving a lane covering problem on the coalition in question. For the base LCG, in which there are no constraints on the set of cycles that can be used in the cover, Özener and Ergun (2008) show that the optimal dual prices of a minimum cost flow circulation formulation can be used to find a cost allocation in the core. As an extension of this work, Hezarkhani et al. (2014) show that the optimal dual prices of the minimum cost flow circulation formulation, hence the allocated costs to the lanes, are non-negative.

The constrained LCP variants can be solved with the help of a set covering (or partitioning) type integer linear program formulated over the entire set of feasible cycles, which do not have zero integrality gap in general. Hence, by the Bondareva–Shapley theorem, the core of a constrained LCG may be empty.

In this paper, we extend the previous line of research on the base LCGs to a class of constrained LCGs based on a set of constrained LCP variants found in the literature. Just like the previous works on the base LCGs, we treat each lane as a player since a shipper may choose to remove each of its lanes from the coalition. We show the set of conditions a coalition of CTTP lanes must satisfy in order to have a non-empty core when the aforementioned constraints are imposed on the cycles that can be used in the cover. These conditions imply that an arbitrarily selected set of lanes under practical constraints on the cycles is highly likely to result in an empty core. Furthermore, we propose nucleolus-based cost allocation methods which aim to find a cost allocation with as little stability violation as possible. We compare our proposed allocation methods with distance proportional cost allocation and Shapley value. We adopt minimum percentage savings and minimum payment requirements into our cost allocation methods. Since the benchmark methods we use do not consider these conditions, some players may not achieve any cost savings while some may be allocated zero cost. We also modify the objective function in the nucleolus to lexicographically minimize the maximum percent violation instead of amount.

When the core of a cooperative game is empty, either the budget balance or the stability conditions can be relaxed. We can satisfy the budget-balance condition but relax the stability conditions, and identify an ϵ-core (Jain & Mahdian, 2007). Alternatively, we can satisfy the stability conditions but incur a budget-deficit (Pál & Tardos, 2003), which must be subsidized by an outside entity such as the governmental organization in a practical application. Since CTTP does not involve such an entity, we relax the stability conditions. We aim to find a cost allocation which violates stability conditions as little as possible without incurring any budget-deficit. Our approach identifies a cost allocation in the core, when the core is non-empty. The approach we employ consists of two phases:

P1: Find the minimum system-wide cost.
P2: Allocate the minimum system-wide cost with as little stability violation as possible.

For P1, we use an integer program from the literature (Ergun et al., 2007a, 2007b) which can identify the lane covering solution minimum total cost. We impose cycle length and cycle cardinality constraints on the underlying LCP for the sake of generality. Note that all of these constraints can be found in the existing literature on solving lane covering problems. Additionally, we develop an efficient algorithm based on column generation to solve the integer program.

For P2, we propose cost allocation methods based on the nucleolus. Since the nucleolus requires the operational cost of every possible coalition, which means solving an exponential number of NP-Hard optimization problems, finding the exact nucleolus solution within reasonable time is extremely challenging. Therefore, we propose and evaluate nucleolus based methods which work with a limited number of coalitions instead of the entire set of coalitions. We incorporate restrictions on the allocated costs which are not considered in the standard nucleolus method. When it is not possible to generate all coalitions, we develop a row generation method to solve the proposed methods over the generated coalitions.

Although our study was motivated by the question of fair cost allocation in the context of CTTP, in which LCGs were first studied, the developed methods can be applied to the rapidly growing concept of digital freight matching (DFM). DFM companies aim to match shipper loads (demand) with carriers (supply) to reduce empty traveling of trucks with the help of technology, especially mobile applications. In addition to offering single loads, DFM also offers bundles of full truck loads that allow a carrier return to its starting point with minimal repositioning. Another practice where LCGs can be applied is yet another recent innovation of DFM called drop-and-hook trailer pools which allow small FTL carriers to accept power-only shipments. Since a trailer in the pool may be used to move loads from multiple shippers, and it must be returned to its initial pickup point, the resulting routes are multi-company load bundles that form tours which ideally should contain minimal empty repositioning. Optimal matching of the bundles and carriers as well as determining the cost of serving each load in the bundles fairly are critical for the operation of the DFM network.

We can summarize our contributions in this paper as follows:

- We are the first to study a constrained LCG, which impose restrictions on the set of cycles that can be used in the lane cover, and we define necessary and sufficient conditions for core stability in a class of constrained LCGs.
- We develop and compare nucleolus based cost allocation methods that incorporate additional considerations such as minimum savings and minimum payment requirements as well as lexicographically minimizing maximum percent stability violation.
- We develop an efficient column generation approach to solve the integer program for P1 and an efficient row generation approach to solve the proposed mathematical models for P2 when it is not possible to generate all feasible coalitions in reasonable time.
- We provide insights about the stability performance of the cost allocation methods we employ with the help of different metrics.

The rest of the paper is organized as follows. In Section 2, we provide a review of the related literature. In Section 3, we discuss the necessary and sufficient conditions for core stability in constrained LCGs. We describe our column generation heuristic in Section 4. In Sections 5 and 6, we discuss the nucleolus and our proposed cost allocation methods, respectively. We present a row generation algorithm to solve proposed allocation methods in Section 7. In Sections 8 and 9, we present two alternative cost allocation methods as benchmarks and present our stability assessment metrics, respectively. We present the results of our computational experiments in Section 10, followed by conclusions in Section 11.

2. Literature review

Collaboration is utilized as an efficiency improvement strategy in a wide array of supply chain functions such as purchasing, demand planning, inventory management and logistics. Erhun and Keskinocak (2011) provide a review of several kinds of collaborative approaches in the supply chain along with the possible benefits of each. Collaboration may occur between different units of the same enterprise (intra-enterprise) or between units of separate enterprises (inter-enterprise). It can also be between the elements of the same supply chain with different functions (vertical), or between the elements performing the same function in different supply chains (horizontal). Accordingly, CTTP can be classified as a form of inter-enterprise horizontal logistics collaboration.

CTTP is a form of shipper collaboration with the differentiating feature that the participants of a CTTP network do not carry out their transportation operations using their own assets. They procure the services of truckload carriers to do so. It should be noted that, most of the types of horizontal logistics collaboration found in the literature are not pure examples of shipper collaboration. The reason is that they typically involve the sharing of a private fleet of vehicles among a group of collaborating companies who are not for-hire carriers.

Ergun et al. (2007a, 2007b) are the first to study CTTP and the optimization problems arising in CTTP networks managed by 3PL companies in the U.S., and introduce the LCPs as well as greedy heuristics for some NP-Hard variants. Özener and Ergun (2008) study the cost/benefit allocation game of the LCP and design methods for allocating the system-wide cost to individual lanes based on optimal dual prices with the goal of maintaining the collaboration. Hezarkhani et al. (2014) further characterize the theoretical properties of dual based methods for allocating the cost of the LCP.

The nucleolus (Schmeidler, 1969) is a well-known cost allocation concept in cooperative game theory. There are many different applications of the nucleolus for different problems. Göthe-Lundgren et al. (1996) work on the nucleolus for allocating costs to visited nodes in vehicle routing. They propose a method for calculating the nucleolus using a restricted set of coalitions corresponding to the set of feasible routes, to which they refer as feasible coalitions. They claim that their method calculates the nucleolus exactly. However, Chardaire (2001) shows that some of the assumptions made by Göthe-Lundgren et al. (1996) are incorrect, and points out that using only the set of feasible coalitions does not guarantee finding the exact nucleolus.

There are an increasing number of works in the literature that focus on allocating costs in collaborative logistics. Frisk et al. (2010) study cost allocation methods for a large fleet sharing application in southern Sweden with eight forest companies. Audy et al. (2010) allocate costs among four Canadian furniture companies. Lozano et al. (2013) compare different cooperative game theory concepts for allocating benefits of horizontal cooperation among four different companies using a simple linear cost-saving model.

Vanovermeire and Sörensen (2014a) develop an iterative coalition building heuristic that integrates the Shapley value and due date change penalties into the cost-minimization problem in a collaborative setting involving the synchronized consolidation of transportation orders. In another work (Vanovermeire & Sörensen, 2014b), they devise methods for measuring and rewarding flexibility in collaborative logistics and incorporate them into existing cost allocation mechanisms, such as the Shapley value and the nucleolus.

Kimms and Kozeletskyi (2016) suggest a cost allocation method for the cooperative traveling salesman problem. This method gives a cost allocation in the core if the core is not empty. In case of an empty core, it gives a least-core solution. Instead of generating all coalitions, they propose a coalition generation procedure. Defryn et al. (2016) work on a cost allocation based on the compensation for non-delivery cost (CND) in the selective vehicle routing problem. CND defines an individual cost allocation for each player or customer when the customer is out of the grand coalition.

Hadas et al. (2017) use an approach based on cooperative game theory to find the contribution of each node to a transportation network. Wang et al. (2018) study a fair profit allocation in the green pickup and delivery problem which aims to minimize total carbon emission. De Vos and Raa (2018) study the Shapley value, the nucleolus, the pre-nucleolus and other cost allocation methods in the inventory routing problem.

Lu and Quadrifoglio (2019) propose a coalition generation procedure for finding the nucleolus and an approximate nucleolus in ride-sharing. Their methods are based on the concept of feasible coalitions proposed by Göthe-Lundgren et al. (1996) for the vehicle routing game. The approximate nucleolus is calculated through a constraint generation procedure utilizing one of two coalition generation sub-problems depending on whether the core is empty or not. The procedure is tested on two 10-node instances from the literature.

Wang et al. (2017) consider stable matching in the ride-sharing problem. They formulate the problem as a maximum weight bipartite matching model which aims to find a stable matching for both riders and drivers such that total weight is maximized. Wang and Yang (2019) define a framework which combines different variables, decisions, objectives and strategies for ride-sharing systems which gather passengers and drivers.

Rasulkhani and Chow (2019) try to find a stable pricing for network transportation systems by using assignment game which aims to maximize total payoffs. In order to find a stable cost allocation to the game, they propose a solution method based on an integer programming problem. Standing et al. (2019) give a wide range of literature review about sharing economy in the transportation. They also discuss how sharing economy create opportunities for sustainability on transportation. Tae et al. (2020) develop an algorithm to find the nucleolus in the vehicle routing game with time windows when the core is not empty.

In this paper, we extend the existing work on LCGs by taking into account limitations on cycles which can be found in constrained LCPs and by proposing nucleolus based cost allocation methods. We use an integer programming model from the literature to determine the total cost of the grand coalition, which can also be used for determining the cost of any coalition. For allocating the cost of the grand coalition, we formulate optimization models based on the nucleolus which feature

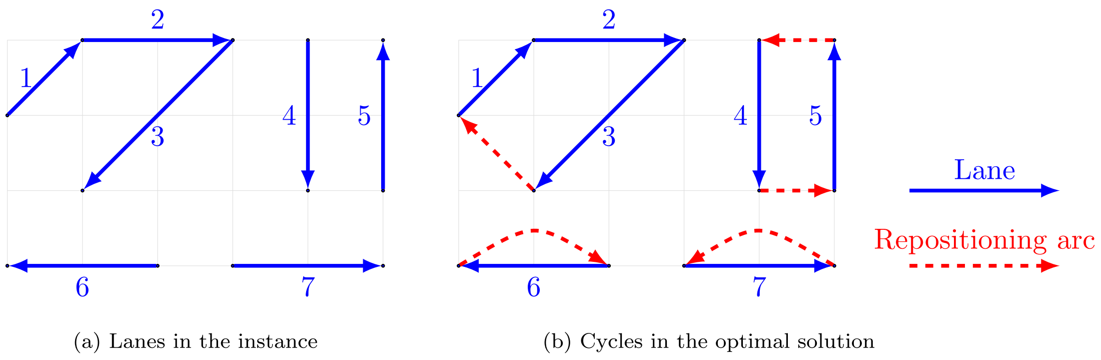


Fig. 1. A simple CL-LCP instance and its optimal solution.

different objective functions and constraints. Similar to Göthe-Lundgren et al. (1996) and Lu and Quadrifoglio (2019), our cost allocation models use only a restricted set of coalitions. We compare the cost allocation methods we propose using problem instances from the literature.

3. Constrained lane covering games and the core

In the problem setting we consider, a set of lanes (i.e. regularly scheduled truckload movements) from multiple shippers will be covered by tours with minimal total repositioning. A set of restrictions, which can be found in the literature on constrained LCPs (Ergun et al., 2007a, 2007b), limit the set of acceptable tours. There is an upper bound on the number lanes in each tour, and an upper bound on the length of each tour. In this setting, determining the maximum possible savings of the collaboration amounts to solving a cardinality-and-length-constrained LCP (CL-LCP). Note that each of the above-mentioned restrictions on the cycles single-handedly turn the underlying LCP into an NP-Hard problem. Therefore, the CL-LCP is also an NP-Hard problem.

Mathematically, the CL-LCP can be defined as follows; given a directed graph $G = (N, A)$ with node set N, arc set A, non-negative arc lengths f_a $\forall a \in A$, and non-negative lane lengths f_l $\forall l \in L \subset A$, find a set of directed simple cycles covering the lanes in L with minimum total cost such that:

1. each cycle contains at most K lanes,
2. each cycle has a length of at most T.

We present a simple CL-LCP instance with seven lanes and its optimal solution in Fig. 1. In Fig. 1(a), there are seven lanes defined by one origin node and one destination node on a graph. We assume that the underlying directed graph is complete. For ease of exposition, the nodes are placed on a unit square grid. Let $K = 3$ lanes and $T = 8$ distance units. Assume that the repositioning cost coefficient (ρ_l) and the movement cost coefficient (β_l) be 1 for lane $l \in L$. The objective of the CL-LCP is to cover all of the lanes with cycles of minimum total cost. For the given example, the selected cycles in the optimal solution can be seen in Fig. 1(b). The blue lines represent the lanes or fully-loaded movement. On the other hand, the red dashed lines correspond to empty movements or deadhead arcs (repositioning arcs). When representing the cycles, deadhead arcs can omitted since they can be inferred from the set of lanes included in each cycle. In Fig. 1(b), there are four distinct cycles: {1,2,3}, {4,5}, {6} and {7} with cost of 7.66, 6, 4, and 4, respectively. The selected cycles in the optimal solution satisfy both cardinality and length constraints imposed by the parameters K and T, respectively, and also cover all of the lanes in the example.

The CL-LCP can be solved with the help of a set covering problem or a set partitioning problem over the set of all feasible cycles. We summarize the notation for the sets, the parameters and the decision variables needed to construct the set partitioning formulation in Table 1.

Table 1
Summary of notations for the CL-LCP.

Sets	
A	Set of arcs
L	Set of lanes
S	Subset of lanes
$C(L)$	Set of all feasible cycles over the lane set L
L_c	Set of lanes covered by cycle $c \in C(L)$
Parameters	
f_c	Cost of feasible cycle $c \in C(L)$
$F(L)$	Objective function value of the problem over the lane set L
s_{lc}	Binary parameter: 1 if lane $l \in L$ is included in cycle $c \in C(L)$, 0 otherwise
Decision variables	
w_l	Allocated cost to lane $l \in L$
x_c	1 if feasible cycle $c \in C(L)$ is selected in the optimal solution, 0 otherwise

We assume that the cost of travel is proportional to the distance traveled. Furthermore, we assume that the cost of traversing an arc is determined by whether the traversal corresponds to an empty truckload movement or a loaded truck movement. Arcs in $A \setminus L$ cannot be traversed as a loaded truck movement. Arcs in L can be traversed as either a loaded truck movement or an empty truck movement, or both. An arc in L must be traversed as a loaded truck movement exactly once, with any further traversals corresponding to empty truck movements. The cost of traversing any arc $a \in A$ as an empty truck movement is equal to $\rho_a f_a$, where f_a is the length of the arc and $\rho_a \in (0, 1]$. Recall that $L \subseteq A$. The cost of traversing a lane arc $l \in L$ as a loaded truck movement is equal to f_l. The cost of a cycle is the sum of empty travel costs and loaded travel costs over it.

The resulting set partitioning formulation is given by Eqs. (1)–(3).

$$\text{SPP:} \quad \min \sum_{c \in C(L)} f_c x_c \tag{1}$$

$$\text{s.t.} \quad \sum_{c \in C(L)} s_{cl} x_c = 1 \qquad \forall l \in L \tag{2}$$

$$x_c \in \{0, 1\} \qquad \forall c \in C(L) \tag{3}$$

In the formulation, the objective (1) is to minimize the sum of the costs of the selected cycles. Constraints (2) ensure that every lane is covered by exactly one cycle. Constraints (3) are the integrality constraints.

Allocating the total cost of the collaboration to the elements of the system yields the benefit of each member of the collaboration. Let $F(L)$ be the optimum objective value of the SPP on the lane set L. We first investigate whether we can distribute $F(L)$ in a fair manner. We model the cost allocation problem as a cooperative game in which the set of lanes corresponds to the set of players, and explore whether we can find a cost allocation in the core. We will refer to the cost allocation game of the CL-LCP as the CL-LCG. Similar to previous works on LCGs (Hezarkhani et al., 2014; Özener & Ergun, 2008), we want to allocate a cost w_l to each lane $l \in L$, such that the allocated costs satisfy the following:

$$\sum_{l\in L} w_l = F(L) \tag{4}$$

$$\sum_{l\in S} w_l \le F(S) \qquad \forall S \subseteq L \tag{5}$$

Recall that a cost allocation in the core must be both budget-balanced (4) and stable (5). If a coalition has a non-empty core, we call it *core stable*. We can simplify the core stability conditions with the help of the following proposition which was shown to hold for the vehicle routing game by Göthe-Lundgren et al. (1996) and the ridesharing game by Lu and Quadrifoglio (2019).

Proposition 1. *Replacing (5) with the inequalities*

$$\sum_{l\in L_c} w_l \le f_c \qquad \forall c \in C(L) \tag{6}$$

does not change the set of cost allocations in the core.

Proof. Since each cycle corresponds to a set of lanes, a cost allocation $w_l, l \in L$ satisfying (5) automatically satisfies (6). Furthermore, the cost of each subset of lanes is the sum of the costs of a set of feasible cycles. Hence, if the allocated costs satisfy (6), they will also satisfy (5). □

Thus, the core of the CL-LCG is non-empty if and only if there exists a cost allocation satisfying Eqs. (4) and (6). Proposition 1 also helps us show an important result.

Proposition 2. *The core of the CL-LCG is non-empty if and only if the SPP has a zero integrality gap.*

Proof. Note that propositions similar to the above have been proven in the literature on set covering type cooperative games (e.g. Göthe-Lundgren et al., 1996; Jain & Mahdian, 2007). We will adapt such proofs to our case. Eqs. (6) can be interpreted as the feasible region of the dual of the linear relaxation of the SPP. In this case, the allocated costs w_l correspond to the dual variables. Let $H(L)$ be the optimal objective value of the following linear program (over the lane set L):

$$\text{SPD:} \qquad \max \quad \sum_{l\in L} w_l \tag{7}$$

$$\text{s.t.} \quad \sum_{l\in L_c} w_l \le f_c \qquad \forall c \in C(L) \tag{8}$$

$$w_l \in \mathbb{R} \qquad \forall l \in L \tag{9}$$

Let $w_l^{SPD}, l \in L$ comprise an optimal solution of the linear program SPD. Clearly, $H(L) = \sum_{l\in L} w_l^{SPD}$. Because of well-known duality theorems, we have $\sum_{l\in L} w_l \le H(L) \le F(L)$ for any cost allocation satisfying the stability conditions. Furthermore, $\sum_{l\in L} w_l^{SPD} = F(L)$ if and only if the SPP has a zero integrality gap. Hence, we cannot find a cost allocation that is both budget-balanced and stable unless the LP relaxation of the SPP is optimal for the SPP. It follows that the core is non-empty if and only if the SPP has zero integrality gap. □

Proposition 2 implies that, in the CL-LCG, keeping an arbitrarily selected set of lanes L together as a core stable coalition is unlikely. It should be noted that finding the exact solution of the SPP for large instances requires sophisticated algorithms such as branch-and-price, which require considerable computational effort. A commonly used approach is applying fast heuristics which do not guarantee finding an optimal solution. The total cost of a heuristic solution is likely to be strictly higher than $F(L)$. Therefore, finding a core cost allocation for a suboptimal solution is not possible.

The above analysis points to an important property of cost allocations in the core:

Proposition 3. *In a core cost allocation, the sum of the costs allocated to the lanes of each cycle in the lane covering solution is equal to the cost of that cycle.*

Proof. Let $C^*(L)$ be the set of cycles in the lane covering solution. Let w^{core} be a cost allocation vector in the core indexed by the lane set L. By (6), the total cost allocated to the cycle $c \in C^*(L)$ is less than or equal to f_c, i.e. $\sum_{l\in L_c} w_l^{core} \le f_c, \forall c \in C^*(L)$. Since $\sum_{c\in C^*(L)} f_c = F(L) \ge 0$, we must have $\sum_{l\in L_c} w_l^{core} = f_c, \forall c \in C^*(L)$. □

It is straightforward to show that the above results hold if we use a set covering formulation to solve the CL-LCP, which would additionally restrict w_l to be non-negative.

4. A heuristic procedure based on column generation

Solving the SPP to optimality is a challenging task since it has an exponential number of columns. Identifying high-quality integer feasible solutions efficiently also turns out to be a challenging task. Therefore, we focus on a solution procedure by solving the SPP over a restricted set of cycles, which we build by solving the LP relaxation of the SPP. The SPP includes one column for each feasible cycle. Since the number of feasible cycles grows rapidly as the problem size grows, we employ a column generation method for solving the LP relaxation of the SPP. We then solve the SPP over the set of cycles corresponding to the columns generated in the process.

We obtain the LP relaxation of the SPP by replacing constraints (3) of the SPP with (10).

$$x_c \ge 0 \qquad \forall c \in C \tag{10}$$

Because of constraints (2), the upper bound on the variables x_c is one. Thus, relaxation (10) is valid for the SPP.

In the rest of the paper, we refer to the LP relaxation of the SPP as the master linear program (MLP). In our solution procedure, we repeatedly solve the restricted master linear program (RMLP), which includes only a portion of the cycle selection variables of the MLP at any time during the run of the algorithm. We initially generate all of the feasible cycles with two lanes, which comprise the cycle columns of the RMLP in the beginning. We start the RMLP with these cycles.

Let $C(L)^{col}$ be the set of feasible cycles corresponding to the cycle selection variables included in the RMLP. We grow $C(L)^{col}$ by finding columns with negative reduced cost. At each iteration of our algorithm, we repeatedly search for new columns with negative reduced cost. We continue the iterations as long as at least one column is identified in the column generation. We terminate the iterations if we do not have a cycle with negative reduced cost.

We can generate new columns with the help of a sub-problem, which aims to find a cycle with the most negative reduced cost. In the next section, we explain the column pricing sub-problem. Since the sub-problem is an integer program, we apply column generation heuristics before solving the respective sub-problem. We explain our column generation heuristic in Section 4.2.

4.1. Column generation

The goal of column generation is to find a feasible cycle with a negative reduced cost in $C(L) \setminus C(L)^{col}$ and add it to $C(L)^{col}$. We first provide a mathematical expression for the reduced cost of a cycle column. Let α_l be the dual variables corresponding to constraints (2) of the MLP, respectively. Then, the reduced cost of a cycle c is given by:

$$\overline{f_c} = f_c - \sum_{l\in L} s_{cl}\alpha_l \tag{11}$$

Let s_{ca}^{dh} be equal to 1 if a cycle c includes arc $a \in A$ as a deadhead arc, 0 otherwise. By re-arranging the terms of (11), we get another expression for the reduced cost of cycle c:

$$\overline{f_c} = \sum_{l\in L} \left(\beta_l f_l - \alpha_l\right) s_{cl} + \sum_{a\in A} \rho_a f_a s_{ca}^{dh} \tag{12}$$

We can use Eq. (12) in our search for negative reduced cost columns. Each cycle c is identified by a unique combination of values of s_{cl}, and s^{dh}_{ca}. Furthermore, $\overline{f_c}$ is a linear function of these terms. We can formulate an integer program to search the possible combinations of these values for the lowest possible reduced cost, subject to the condition that they correspond to a feasible cycle. If the lowest possible reduced cost is positive, then we can stop searching for new columns.

We define decision variables corresponding to s_{cl}, and s^{dh}_{ca}. We define r_l as the binary decision variable which equals 1 if lane $l \in L$ is traversed as a loaded truck movement and equals 0 otherwise. We defined t_a as the binary decision variable which equals 1 if $a \in A$ if arc $a \in A$ is traversed as an empty truck movement and equals 0 otherwise. The resulting column pricing subproblem (CPSP) is given below.

$$\text{CPSP:}\quad \min \sum_{l\in L}(\beta_l f_l - \alpha_l) r_l + \sum_{a\in A}\rho_a f_a t_a \tag{13}$$

$$\text{s.t.}\quad \sum_{(m,n)\in L} r_{(m,n)} + \sum_{(m,n)\in A} t_{(m,n)} - \sum_{(n,m)\in L} r_{(n,m)} - \sum_{(n,m)\in A} t_{(n,m)} = 0 \qquad \forall n \in N \tag{14}$$

$$\sum_{(m,n)\in L} r_{(m,n)} + \sum_{(m,n)\in A} t_{(m,n)} \le 1 \qquad \forall n \in N \tag{15}$$

$$\sum_{(n,m)\in A} t_{(n,m)} + \sum_{(k,n)\in A} t_{(k,n)} \le 1 \qquad \forall n \in N \tag{16}$$

$$\sum_{l\in L} r_l \le K \tag{17}$$

$$\sum_{l\in L} f_l r_l + \sum_{a\in A} f_a t_a \le T \tag{18}$$

$$r_l \in \{0,1\} \qquad \forall l \in L \tag{19}$$

$$t_a \in \{0,1\} \qquad \forall a \in A \tag{20}$$

The objective (13) of the CPSP is to identify the cycle(s) with minimum reduced cost. Constraints (14) make the number of arcs (loaded plus empty truck movements) entering and leaving each node equal to each other. Constraints (15) ensure that any feasible solution consists only of simple cycles. Constraints (16) prevent to select two consecutive deadhead arcs. Constraints (17) and (18) enforce the cardinality and length constraints, respectively. Constraints (19) and (20) are the integrality constraints.

The solution of the CPSP may include more than one simple cycle. Additional constraints can be added to the formulation to prevent multiple cycles. However, these constraints will have a negative impact on the tractability of the pricing subproblem. In fact, we can easily show that the single cycle CPSP is NP-Hard by a reduction from the asymmetric traveling salesman problem. In addition, any solution with a single cycle will be a feasible solution to the above integer program. Hence, the formulation presented above is a relaxation of the true pricing subproblem. Consequently, its objective value cannot be worse. If the optimal objective value of the CPSP is non-negative, then we can conclude that there does not exist any new feasible simple cycles which can improve the objective value of the RMLP. If the optimal objective value of the CPSP is negative, then the solution may include more than one simple cycle with a negative reduced cost. In that case, we add the one with most negative reduced cost to the RMLP.

4.2. *Pricing heuristics*

The CPSP must be solved several times when searching for the optimal solution of the MLP. Solving the sub-problem, which is an integer program, several times amounts to a considerable cumulative computational effort. In order to generate the columns more efficiently, we apply four heuristics for generating new columns before solving the CPSP. We have adopted these pricing heuristics from Kuyzu (2017) and Xu et al. (2003). Since these heuristics add cycles to $C(L)^{col}$, we will refer to them as cycle generation heuristics in the rest of this section. The heuristics try to minimize the objective value of the CPSP when looking for new columns.

The first pricing heuristic, *Insert1*, aims to obtain new columns with negative reduced cost by adding a lane into existing non-positive reduced cost columns of the RMLP. *Insert1* is equivalent to merging a cycle with a cycle containing only one lane. All possibilities are evaluated and the one with the most negative reduced cost is used to generate a new column. Since each feasible cycle has at most K lanes, there are $O(|L|^K)$ non-positive reduced cost columns, and hence cycles, in the RMLP. There are at most $|L|$ candidate lanes for insertion into a cycle. A lane can be inserted into a cycle at one of at most K positions. Therefore, each run of *Insert1* has a runtime complexity of $O(K \times |L|^{K+1})$.

The second cycle generation heuristic, *Merge*, searches for new columns by merging pairs of disjoint cycles corresponding to the columns of the RMLP with non-positive reduced cost. *Merge* essentially performs a 2-opt over two disjoint cycles such that one deadhead arc is removed from each of the two cycles involved and replaced with two deadhead arcs connecting the resulting directed paths into a single simple cycle. A pair of cycles may be merged in more than one way depending on the deadhead arcs removed, so *Merge* finds the best way of merging a given pair of cycles by trying all possible deadhead arc pairs. Each run of *Merge* has a runtime complexity of $O(K^2 \times |L|^{2K})$.

The third pricing heuristic, *Cross-Merge*, tries to find new columns by merging two disjoint cycles such that one has a non-positive reduced cost while the other has a positive reduced cost. Each run of *Cross-Merge* has a runtime complexity of $O(K^2 \times |L|^{2K})$.

The fourth pricing heuristic, *Insert2*, searches for new columns with negative reduced cost by inserting each lane into the existing cycles at all possible position. This heuristic evaluates all alternatives to find the one resulting in the most negative reduced cost. Hence, since processing this heuristic might take a little long time than the others, this heuristic is used as the last alternative. Each run of *Insert2* has a runtime complexity of $O(K \times |L|^{K+1})$.

Searching the columns with negative reduced cost starts with *Insert1* and goes with the others in the order given above, as needed. Each of the heuristics looks for the best move in its neighborhood, i.e. the move resulting in the column with the most negative reduced cost. If the pricing heuristics cannot find a new column with a negative reduced cost, the CPSP is solved to optimality using a commercial solver. If solving the CPSP does not yield a new column, the SPP is solved to optimality over the union of the cycle sets $C(L)$ and $C(L)^{col}$. Note that this procedure is a heuristic procedure. Because of that, it does not guarantee that it yields an optimal solution. We evaluate our proposed solution procedure in Section 10. We will refer to this solution approach as the CGH in the rest of the paper.

5. The nucleolus

The nucleolus is the unique cost allocation that lexicographically minimizes the maximum amount of stability violation over all possible coalitions. Calculating the nucleolus requires solving a sequence of linear programs (LPs). Since the linear model may have alternative optimal solutions, the nucleolus is sometimes calculated incorrectly. In order to avoid this miscalculation, we will adopt the method described by Guajardo and Jörnsten (2015), which recommends the use of dual variables and complementary slackness conditions.

Let $F(L)$ denote the system-wide cost from the SPP and $F(S)$ denote the cost of coalition $S \subset L$. As before, the decision variables $w_l, l \in L$ denote the allocated costs. The decision variable ε_1 denotes an upper bound on the maximum stability violation amount over all possible coalitions. The following LP is solved at iteration 1 of the procedure for calculating the nucleolus.

$$\text{LP}_1: \quad \min\ \varepsilon_1 \tag{21}$$

$$\text{s.t.}\ \sum_{l\in L} w_l = F(L) \tag{22}$$

$$\sum_{l\in S} w_l \le F(S)+\varepsilon_1 \qquad \forall S\subset L \tag{23}$$

$$w_l \in \mathbb{R} \qquad \forall l \in L \tag{24}$$

$$\varepsilon_1 \in \mathbb{R} \tag{25}$$

The objective (21) of the model is to minimize the maximum amount of stability violation. Constraints (22) ensure that the total allocated cost is equal to the total cost from the SPP. Constraints (23) ensure that ε_1 is greater than or equal to the violation amount of any coalition. All of the decision variables in this model are unrestricted, and they are defined over all real numbers.

Let $\hat{S}_i$ be the set of coalitions for which the corresponding optimal dual variables of constraints (23) are negative in the optimal solution of LP_i, which will be satisfied with equality by complementary slackness. After defining $\hat{S}_i$, the nucleolus model used in iteration $k \ge 2$ is defined by (26)–(31). Similar to ε_1, the decision variable ε_k denotes an upper bound on the maximum stability violation at iteration $k \ge 2$.

$$\text{LP}_k: \qquad \min\ \varepsilon_k \tag{26}$$

$$\text{s.t.} \sum_{l\in L} w_l = F(L) \tag{27}$$

$$\sum_{l\in S} w_l \le F(S)+\varepsilon_k \qquad \forall S\subset L : S \notin \bigcup_{i=1}^{k-1} \hat{S}_i \tag{28}$$

$$\sum_{l\in S} w_l = F(S)+\varepsilon_i^* \qquad \forall S \in \bigcup_{i=1}^{k-1} \hat{S}_i \tag{29}$$

$$w_l \in \mathbb{R} \qquad \forall l \in L \tag{30}$$

$$\varepsilon_k \in \mathbb{R} \tag{31}$$

Similar to constraints (23), constraints (28) also ensure that ε_k is greater than or equal to the violation amount of any coalition which has not been identified as part of $\hat{S}_i$ in any previous iteration $i < k$. Constraints (29) ensure that the violation amounts of $\hat{S}_i$ in previous iterations $i < k$ are maintained. The parameters ε_i^* in these constraints show the optimal value of the decision variable ε_k at iteration $i < k$.

The iterations continue until the associated LP has a unique solution. Constraints (28) and (29) are updated at the end of each iteration, as described above. The final solution at the end of this process yields the nucleolus, which is unique.

6. Proposed cost allocation methods

In this section, we discuss our proposed nucleolus-based cost allocation methods. After the SPP is solved, these methods are used to allocate the total cost among the lanes. The SPP provides the total system-wide cost and the set of cycles for the cost allocation methods.

6.1. Semi-nucleolus

Calculating the nucleolus of the CL-LCG requires generating the set of all possible coalitions, which corresponds to the set of all proper subsets of the lane set L, and solving an NP-Hard optimization problem (i.e. CL-LCP) on each possible coalition. Real-life LCP instances are expected to have a large number of lanes. Therefore, calculating the exact nucleolus is not a viable option. Due to this computational difficulty, we propose to generate only all feasible cycles which satisfy all of the constraints on the cycles. The set of all feasible cycles $C(L)$ corresponds to the set of all feasible coalitions. In that case, constraints (32) are used in LP_1 instead of constraints (22).

$$\sum_{l\in L_c} w_l \le f_c+\varepsilon_1 \qquad \forall c \in C(L) \tag{32}$$

Additionally, let $\hat{C}_i$ be the set of equality constraints to constraints (32) at iteration i and λ_c be the dual variable corresponding to the constraint (32). Next, constraints (28) and (29) are replaced with constraints (33) and (34) in the nucleolus model, respectively. After these changes, we call the resulting cost allocation method the semi-nucleolus (s-nucleolus).

$$\sum_{l\in L_c} w_l \le f_c+\varepsilon_k \qquad \forall c \in C(L) : c \notin \bigcup_{i=1}^{k-1} \hat{C}_i \tag{33}$$

$$\sum_{l\in L_c} w_l = f_c+\varepsilon_i^* \qquad \forall c \in \bigcup_{i=1}^{k-1} \hat{C}_i \tag{34}$$

6.2. Modified semi-nucleolus

Both the nucleolus and the s-nucleolus may cause some lanes to achieve large savings while causing some lanes to achieve zero savings. Another possibility is that they could allocate zero or negative cost to some lanes. In order to deal with these drawbacks, we propose some changes in the s-nucleolus.

We need to define additional parameters to transform them into our proposed model. Let f_l be the cost of one-way loaded truck movement for lane $l \in L$. Let π_l be a coefficient for the lower bound of the cost allocated to lane $l \in L$. We add two constraints, given by (35) and (36), for each lane. Constraints (35) are individual rationality which guarantees that allocation cost to a lane cannot exceed its individual cost. Constraints (36) impose a lower bound on the allocated cost for each lane. We call the resulting cost allocation the modified s-nucleolus (ms-nucleolus).

$$w_l \le F(\{l\}) \qquad \forall l \in L \tag{35}$$

$$w_l \ge \pi_l f_l \qquad \forall l \in L \tag{36}$$

6.3. Percentage s-nucleolus and percentage ms-nucleolus

As mentioned before, the s-nucleolus lexicographically minimizes the maximum amount of stability violation. In that case, the maximum percentage of stability violation could be high. In this section, we propose to lexicographically minimize the maximum stability violation percentage instead of lexicographically minimizing the maximum stability violation amount for both the semi-nucleolus and ms-nucleolus. To do that, we replace constraints (32), (33) and (34) with constraints (37), (38) and (39), respectively. The rest of the constraints remain the same.

$$\sum_{l\in L_c} w_l \le f_c(1+\varepsilon_1/100) \qquad \forall c \in C(L) \tag{37}$$

$$\sum_{l\in L_c} w_l \le f_c(1+\varepsilon_k/100) \qquad \forall c \in C(L) : c \notin \bigcup_{i=1}^{k-1} \hat{C}_i \tag{38}$$

$$\sum_{l\in L_c} w_l = f_c(1+\varepsilon_i^*/100) \qquad \forall c \in \bigcup_{i=1}^{k-1} \hat{C}_i \tag{39}$$

We refer to the cost allocations resulting from altering s-nucleolus and ms-nucleolus as percentage s-nucleolus (%s-nucleolus) and percentage ms-nucleolus (%ms-nucleolus), respectively. In both the %s-nucleolus and the %ms-nucleolus, the decision variable ε_k $(k = 1, 2, \dots)$ denotes the stability violation as a percentage of the cost of the cycle in the stability constraint for each coalition.

7. Row generation procedure for the nucleolus based models

The naive way of solving the nucleolus based models is to generate all feasible cycles or coalitions and then solve them over these cycles. However, like the SPP, the formulations of the nucleolus based cost allocation models contain a row for each cycle because the stability constraints are defined over the set of feasible cycles $C(L)$. When the number of cycles, which grows exponentially, becomes too high, generating all feasible cycles and applying the nucleolus based methods become extremely difficult if not impossible.

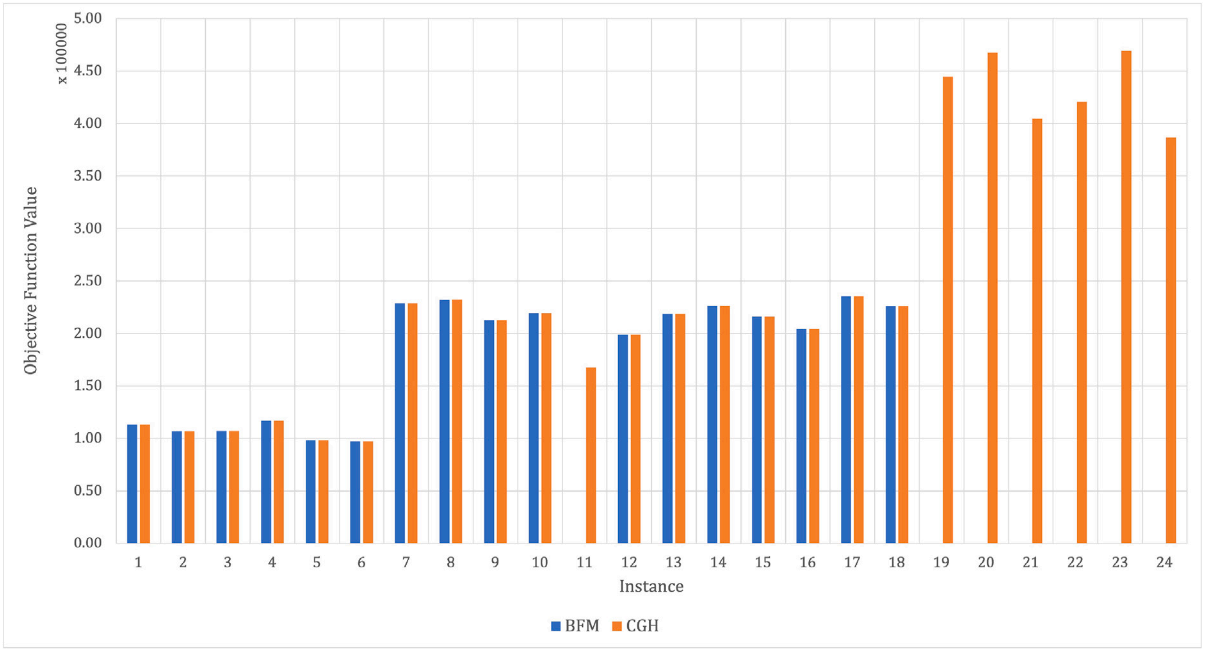


Fig. 2. Comparison results between objective function values of the brute force and the CGH.

Table 2
Instances used in the experiments.

Instance	$\lvert N\rvert$	$\lvert L\rvert$	CPR
1–3	100	100	0.5
4–6	100	100	0.8
7–9	100	200	0.5
10–12	100	200	0.8
13–15	200	200	0.5
16–18	200	200	0.8
19–21	200	400	0.5
22–24	200	400	0.8

As a remedy, we start the nucleolus based models with a small number of stability constraints and add more as needed. Let $\tilde{w}$ be a budget balanced cost allocation vector indexed by the lane set L. That is, $(\tilde{w})$ is the cost allocated to lane $l \in L$. Then, the *stability violation* of cycle $c \in C(L)$ is calculated as in (40).

$$\sum_{l\in L_c} \tilde{w}_l - f_c \tag{40}$$

Let $C(L)^{row}$ denote the set of generated rows. The objective is to find the constraint with the highest stability violation and add to $C(L)^{row}$. In order to find such cycles, we solve the row generation subproblem (RGSP), which is very similar to the CPSP, to optimality using a commercial solver.

$$\text{RGSP:}\quad \min \quad \sum_{l\in L}(\beta_l f_l - \tilde{w}_l) r_l + \sum_{a\in A} \rho_a f_a t_a \tag{41}$$

$$\text{s.t.} \quad (14)–(20)$$

Since the RGSP is the same as the CPSP except its objective function, we adapt the column pricing heuristics to identify violated stability constraints. We solve the RGSP if none of the heuristics is successful. In the row generation stage, we consider both pairs of disjoint cycles with positive stability violation and pairs of disjoint cycles with negative stability violation.

The RGSP seeks a set of feasible cycle(s) with the maximum stability violation. If the optimal objective value of the RGSP is non-negative, then the current cost allocation of the respective proposed model satisfies all of the stability constraints. Like the CPSP, the RGSP may return more than one simple cycle. The objective value of the RGSP is the negative of the sum of the stability violations of the cycles in the solution. Hence, if the optimal objective value of the RGSP is negative, then all of the cycles in the solution must have zero or positive stability violation.

If the RGSP solution has a negative objective value, we add the stability constraint(s) of (all of) the positive stability violation cycle(s) in the solution to $C(L)^{row}$. That is, we add the row(s) corresponding to the cycle(s) into the nucleolus based models. Once a new row is added to the models, we solve them over the current cycle set $(C(L)^{row})$.

If the RGSP solution has a non-negative objective value and the stopping conditions are satisfied, then it means that we cannot find any cycle that violates the stability condition for the respective proposed model. In the rest of the paper, we refer to our row generation procedure as RGA. Note that RGA is an exact solution procedure which means that when an allocation is given, it can find all non-generated rows (cycles) which violate the stability condition.

The RGA is used to solve the nucleolus based models over the limited cycles instead of generating all feasible cycles. After each iteration of the nucleolus based models, we use the RGA to find the cycles which violate the stability condition over the allocation found in the models. We add the one with most violating stability condition to the models at each iteration. After adding such cycle, the nucleolus based models are solved again until the RGA does not find such cycles and the stopping conditions of the models are satisfied.

8. Alternative cost allocation methods for comparison

In this section, we discuss two basic cost allocation methods we will utilize in our computational experiments for comparison purposes. In these methods, we first obtain a solution for the SPP and then allocate the costs. We require that the sum of the costs allocated to the lanes of each selected cycle equals the cost of the cycle.

8.1. Distance proportional

One of the first ideas that come to mind when faced with a cost allocation question is allocating costs based on resource consumption.

Table 3
The CGH versus the brute force method in solving the SPP.

Instance	Objective value			$\|C(L)^{SPP}\|$			Process time		$\|\hat{C}(L)\|$		$\|\hat{L}\|$	
	BF	CGH	Diff.	BF	CGH	Diff.	BF	CGH	BF	CGH	BF	CGH
1	113,210.86	113,210.86	0.00	86,399	3,278	83,121	6.73	19.57	36	36	96	96
2	106,937.84	106,937.84	0.00	301,241	3,951	297,290	27.36	35.83	34	34	96	96
3	107,163.47	107,163.47	0.00	166,085	3,666	162,419	15.87	53.72	33	33	98	98
4	117,249.92	117,249.92	0.00	85,105	2,750	82,355	6.20	22.42	37	37	97	97
5	98,630.18	98,635.73	5.55	332,606	3,877	328,729	23.78	48.43	31	31	99	99
6	97,201.06	97,201.06	0.00	272,819	4,413	268,406	15.90	58.63	32	32	98	98
7	228,884.08	228,895.78	11.70	608,667	11,451	597,216	107.16	225.97	73	73	195	194
8	232,166.18	232,180.50	14.32	467,757	11,122	456,635	65.02	210.97	70	71	193	192
9	212,532.28	212,532.37	0.09	865,391	12,982	852,409	196.24	357.85	68	68	197	196
10	219,269.23	219,269.23	0.00	3,760,950	13,705	3,747,245	2700.44	447.14	65	65	191	191
11	–	167,774.37	–	–	19,565	–	–	740.83	–	60	–	197
12	199,002.28	199,002.28	0.00	3,689,916	15,508	3,674,408	1804.08	463.62	66	68	197	197
13	218,755.83	218,763.75	7.92	1,118,167	12,323	1,105,844	81.80	381.72	67	67	198	198
14	226,254.32	226,271.77	17.45	628,507	11,763	616,744	54.87	562.44	69	68	195	195
15	215,983.73	215,997.47	13.74	5,488,114	14,643	5,473,471	353.35	578.47	63	65	191	194
16	204,486.47	204,486.47	0.00	1,476,653	13,620	1,463,033	101.34	577.81	66	65	199	199
17	235,422.64	235,432.30	9.66	536,656	9,436	527,220	44.39	338.07	72	71	198	198
18	225,787.70	225,795.47	7.77	1,064,140	11,895	1,052,245	79.35	506.03	69	70	197	197
19	–	444,590.38	–	–	49,674	–	–	5,865.17	–	134	–	389
20	–	467,361.91	–	–	40,979	–	–	5,093.11	–	141	–	394
21	–	404,561.90	–	–	54,084	–	–	8,847.98	–	126	–	396
22	–	420,551.04	–	–	51,318	–	–	7,971.37	–	132	–	399
23	–	469,278.77	–	–	41,718	–	–	4,640.64	–	140	–	392
24	–	386,627.27	–	–	61,576	–	–	12,148.17	–	120	–	394

BF: Brute Force
CGH: Column Generation Heuristic
Diff.: Difference.

Table 4
The RGA versus the brute force method in computing the s-nucleolus.

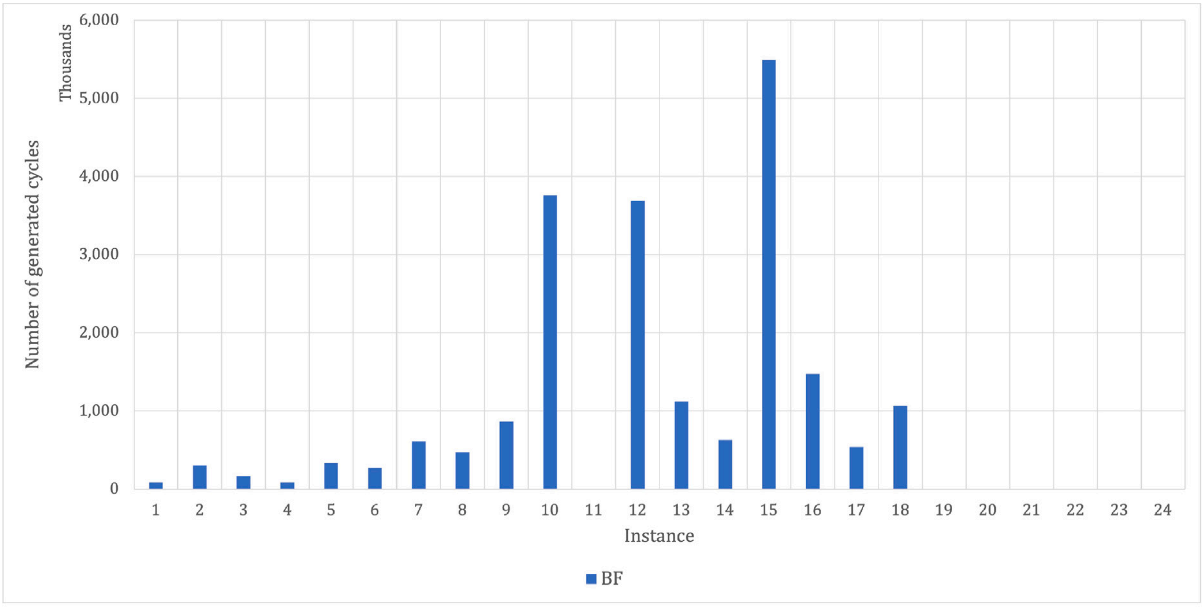
Instance	NoSV			Process time			MaxPSV		Avg. $Uc(l)$	
	BF	RGA	Diff.	BF	RGA	Diff.	BF	RGA	BF	RGA
1	65	65	0	6.67	16.27	9.60	0.03	0.03	1.20	1.20
2	0	0	0	7.12	5.95	−1.17	0.00	0.00	1.29	1.28
3	57	57	0	43.80	41.55	−2.25	0.03	0.03	1.19	1.19
4	52	52	0	3.88	3.52	−0.36	0.00	0.00	1.14	1.15
5	83	99	16	44.20	37.77	−6.43	0.03	0.06	1.17	1.17
6	0	0	0	6.62	5.81	−0.81	0.00	0.00	1.14	1.14
7	201	218	17	2,413.66	287.23	−2,126.43	0.09	0.20	1.25	1.25
8	214	237	23	5,834.70	335.83	−5,498.87	0.01	0.09	1.26	1.27
9	210	210	0	2,083.07	300.99	−1,782.08	0.04	0.04	1.22	1.22
10	424	192	−232	58,453.21	621.44	−57,831.77	0.00	0.00	1.22	1.22
11	–	942	–	–	16,244.26	–	–	0.00	–	1.25
12	0	0	0	193.01	18.99	−174.02	0.00	0.00	1.19	1.19
13	186	190	4	1,901.59	174.30	−1,727.29	0.06	0.13	1.22	1.22
14	183	198	15	294.71	437.75	143.04	0.34	0.55	1.22	1.22
15	156	176	20	16,606.13	396.55	−16,209.58	0.01	0.06	1.22	1.23
16	170	170	0	5,630.07	401.14	−5,228.93	0.06	0.05	1.14	1.14
17	187	199	12	1,120.44	450.67	−669.77	0.09	0.15	1.12	1.11
18	114	127	13	206.88	368.31	161.43	0.01	0.02	1.21	1.21
19	–	432	–	–	2,541.47	–	–	0.14	–	1.23
20	–	434	–	–	3,217.64	–	–	0.24	–	1.23
21	–	699	–	–	8,667.56	–	–	0.09	–	1.19
22	–	701	–	–	3,793.87	–	–	0.08	–	1.14
23	–	452	–	–	3,010.86	–	–	0.04	–	1.16
24	–	576	–	–	6,490.60	–	–	0.02	–	1.16

BF: Brute Force, RGA: Row Generation Approach, Diff.: Difference, Avg.: Average
NoSV: The Number of Violated Stability Constraints
MaxPSV: The Maximum of Percent stability Violations
$Uc(l)$: The unit allocated cost ($\frac{\bar{w}_l}{\beta_l f_l}$).

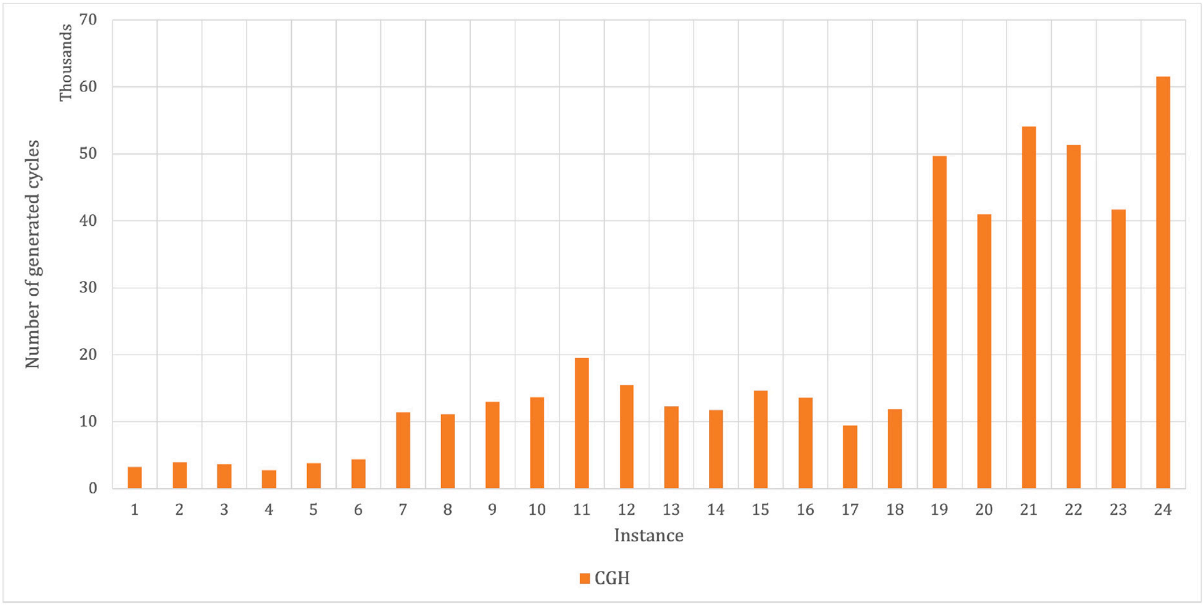
In a collaborative tour, the consumption of resources depends on the distance traveled, which includes both loaded and empty truck movements. Hence, we include a distance proportional cost allocation method in our comparison.

The direct resource consumption of a lane is the loaded travel along the lane. Therefore, we distribute the cost of each selected cycle to its lanes proportional to the lane distance. This way, each lane in a cycle is allocated the same cost per loaded unit distance. More formally, we calculate the cost allocated to lane l in a selected cycle c as in Eq. (42).

$$w_l^{c,prop} = \beta_l f_l \left(\frac{f_c}{\sum_{l_c \in L_c} f_{l_c}} \right) \tag{42}$$



(a) The number of generated cycles in the brute force



(b) The number of generated cycles in the CGH

Fig. 3. The number of generated cycles in the solution methods.

8.2. *Modified Shapley value*

The second method we include in our comparison is based on the Shapley value. The Shapley value of each lane l is calculated as in Eq. (43). This calculation requires the solution of the SPP on each and every subset of the lane set that include the lane l, which would require an exponentially increasing level of computational effort.

$$w_l^{shap} = \sum_{S \subseteq L: l \in S} \left(\frac{(|S|-1)!(|L|-|S|)!}{|L|!} (F(S) - F(S - \{l\})) \right) \quad (43)$$

Because of the heavy computational burden involved with calculating the Shapley value as in Eq. (43), we calculate an approximate value using a limited number of subsets. More specifically, we only consider the subsets consisting of the lanes of the cycle covering lane l in the SPP solution. The allocated cost is given by Eq. (44). We call the resulting cost allocation method the modified Shapley value.

$$w_l^{c,shap} = \sum_{S \subseteq L_c: l \in S} \left(\frac{(|S|-1)!(|L_c|-|S|)!}{|L_c|!} (F(S) - F(S - \{l\})) \right) \quad (44)$$

9. Stability assessment

We have shown in Section 3 that strong conditions exist for the grand coalition to be core stable. Therefore, the core may be empty for the grand coalition. In such a case, our proposed allocation methods and the alternative cost allocation methods we compare may satisfy the stability conditions to varying degrees.

We assess the stability of the solutions provided by the cost allocation mechanisms we consider using a method that is inspired by the stability assessment method of Özener et al. (2013). We use the number of cycles violating the stability constraints, the maximum

Table 5
The CGH and the RGA versus the brute force method in terms of total process time.

Instance	BF			CGH + RGA			Difference
	SPP	s-Nucleolus	TPT	SPP	s-Nucleolus	TPT	
1	6.73	6.67	13.40	19.57	16.27	35.84	22.44
2	27.36	7.12	34.48	35.83	5.95	41.78	7.30
3	15.87	43.80	59.67	53.72	41.55	95.27	35.60
4	6.20	3.88	10.08	22.42	3.52	25.94	15.86
5	23.78	44.20	67.98	48.43	37.77	86.20	18.22
6	15.90	6.62	22.52	58.63	5.81	64.44	41.92
7	107.16	2,413.66	2,520.82	225.97	287.23	513.20	−2,007.62
8	65.02	5,834.70	5,899.72	210.97	335.83	546.80	−5,352.92
9	196.24	2,083.07	2,279.31	357.85	300.99	658.84	−1,620.47
10	2,700.44	58,453.21	61,153.65	447.14	621.44	1,068.58	−60,085.07
11	–	–	–	740.83	16,244.26	16,985.09	–
12	1,804.08	193.01	1,997.09	463.62	18.99	482.61	−1,514.48
13	81.80	1,901.59	1,983.39	381.72	174.30	556.02	−1,427.37
14	54.87	294.71	349.58	562.44	437.75	1,000.19	650.61
15	353.35	16,606.13	16,959.48	578.47	396.55	975.02	−15,984.46
16	101.34	5,630.07	5,731.41	577.81	401.14	978.95	−4,752.46
17	44.39	1,120.44	1,164.83	338.07	450.67	788.74	−376.09
18	79.35	206.88	286.23	506.03	368.31	874.34	588.11
19	–	–	–	5,865.17	2,541.47	8,406.64	–
20	–	–	–	5,093.11	3,217.64	8,310.75	–
21	–	–	–	8,847.98	8,667.56	17,515.54	–
22	–	–	–	7,971.37	3,793.87	11,765.24	–
23	–	–	–	4,640.64	3,010.86	7,651.50	–
24	–	–	–	12,148.17	6,490.60	18,638.77	–

BF: Brute Force
TPT: Total Process Time
CGH: Column Generation Heuristic
RGA: Row Generation Approach.

Table 6
Stability assessment results for cost allocation methods.

Instance	DPCA			MSV			s-Nucleolus			%s-Nucleolus			ms-Nucleolus			%ms-Nucleolus		
	NoSV	AvgPSV	MaxPSV	NoSV	AvgPSV	MaxPSV	NoSV	AvgPSV	MaxPSV	NoSV	AvgPSV	MaxPSV	NoSV	AvgPSV	MaxPSV	NoSV	AvgPSV	MaxPSV
1	4,326	4.54	36.80	3,503	4.06	137.62	65	0.00	0.03	65	0.00	0.00	61	0.00	0.01	61	0.00	0.00
2	24,159	5.43	37.55	13,129	4.57	38.94	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00
3	3,474	3.63	30.54	3,052	3.56	25.10	57	0.00	0.03	57	0.00	0.00	54	0.00	0.01	54	0.00	0.00
4	8,928	5.89	36.39	8,603	6.34	59.32	52	0.00	0.00	52	0.00	0.00	49	0.00	0.00	49	0.00	0.00
5	6,878	3.90	24.97	5,114	3.07	28.20	99	0.00	0.06	100	0.00	0.01	99	0.00	0.03	99	0.00	0.01
6	12,166	4.79	34.49	10,295	4.53	35.99	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00
7	35,645	4.91	43.13	44,406	5.29	113.99	218	0.00	0.20	231	0.00	0.01	220	0.00	0.03	267	0.00	0.01
8	41,427	5.13	35.25	38,709	4.39	29.32	237	0.00	0.09	292	0.00	0.01	230	0.00	0.02	236	0.00	0.01
9	38,048	4.57	30.70	39,769	4.10	211.81	210	0.00	0.04	205	0.00	0.00	194	0.00	0.00	253	0.00	0.00
10	218,113	4.95	35.09	192,339	4.78	65.22	192	0.00	0.00	192	0.00	0.00	182	0.00	0.00	182	0.00	0.00
11	–	–	–	–	–	–	942	0.00	0.00	418	0.00	0.00	938	0.00	0.00	468	0.00	0.00
12	168,708	5.05	39.23	203,832	5.27	129.82	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00	0	0.00	0.00
13	28,360	4.58	32.71	31,652	4.69	175.06	190	0.00	0.13	187	0.00	0.01	185	0.00	0.02	180	0.00	0.01
14	31,334	5.79	42.61	27,444	5.34	136.19	198	0.00	0.55	199	0.00	0.02	191	0.00	0.05	190	0.00	0.02
15	376,128	6.41	50.67	358,650	5.77	115.79	176	0.00	0.06	176	0.00	0.01	163	0.00	0.02	163	0.00	0.01
16	30,651	3.24	31.16	67,471	5.01	87.26	170	0.00	0.05	169	0.00	0.00	167	0.00	0.01	230	0.00	0.00
17	17,832	4.82	38.69	20,442	4.88	40.51	199	0.00	0.15	195	0.00	0.01	195	0.00	0.07	192	0.00	0.01
18	59,874	5.41	35.15	49,058	4.87	59.06	127	0.00	0.02	128	0.00	0.01	123	0.00	0.01	124	0.00	0.01
19	–	–	–	–	–	–	432	0.00	0.14	492	0.00	0.01	433	0.00	0.01	476	0.00	0.01
20	–	–	–	–	–	–	434	0.00	0.24	487	0.00	0.01	419	0.00	0.02	427	0.00	0.01
21	–	–	–	–	–	–	699	0.00	0.09	772	0.00	0.01	706	0.00	0.03	769	0.00	0.01
22	–	–	–	–	–	–	701	0.00	0.08	682	0.00	0.01	645	0.00	0.02	685	0.00	0.01
23	–	–	–	–	–	–	452	0.00	0.04	444	0.00	0.01	432	0.00	0.01	502	0.00	0.01
24	–	–	–	–	–	–	576	0.00	0.02	634	0.00	0.00	620	0.00	0.01	622	0.00	0.00
Average	**65,061.82**	**4.88**	**36.18**	**65,733.41**	**4.74**	**87.60**	**267.75**	**0.00**	**0.08**	**257.38**	**0.00**	**0.01**	**262.75**	**0.00**	**0.02**	**259.54**	**0.00**	**0.01**

DPCA: Distance Proportional Cost Allocation, MSV: Modified Shapley Value
NoSV: The Number of Violated Stability Constraints
AvgPSV: The Average of Percent stability Violations
MaxPSV: The Maximum of Percent stability Violations.

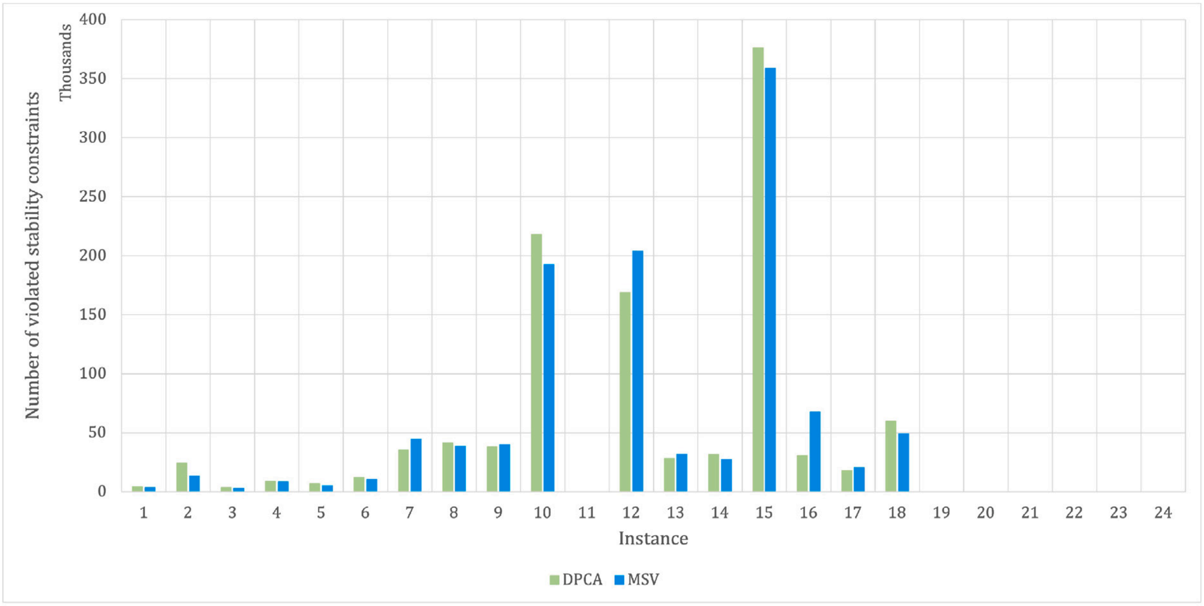
percent stability violation, and the average percent stability violation as stability assessment metrics. We calculate the percent violation for each stability constraint as in Eq. (45), where c is the cycle corresponding to the stability constraint. Özener et al. (2013) calculate these metrics over a large set of randomly generated subsets. We, on the other hand, identify all of the violated stability constraints (6) for a given cost allocation $\tilde{w}_l, l \in L$ by generating feasible cycles as needed.

$$PSV(c) = \max\{0, \frac{\sum_{l \in L_c} \tilde{w}_l - f_c}{f_c}\} \tag{45}$$

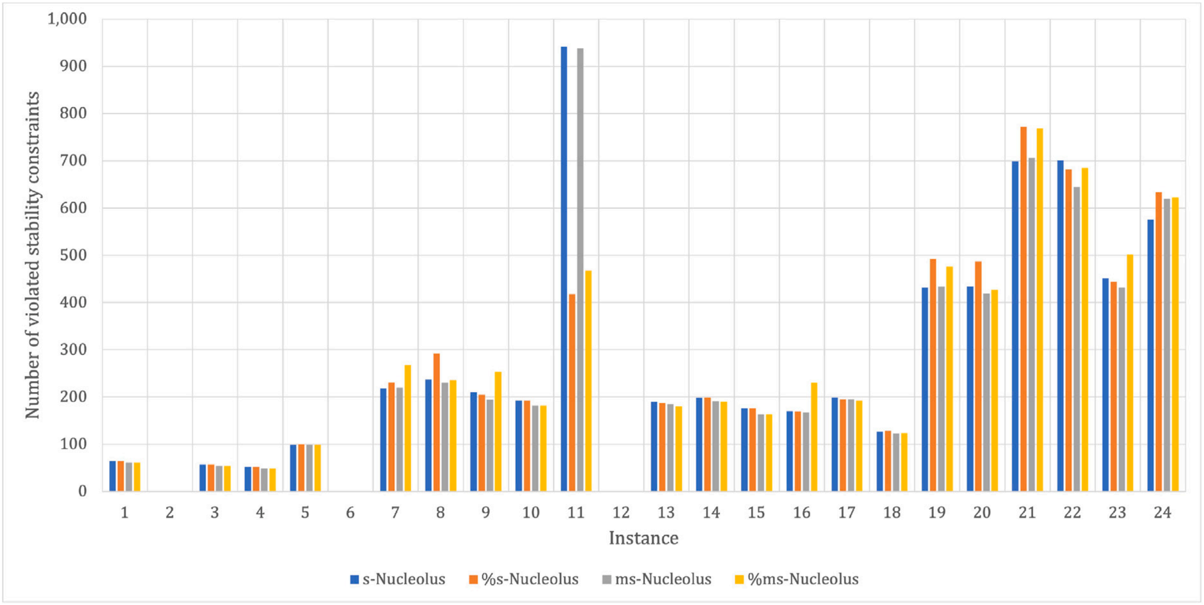
In addition, we calculate statistics on percentage cost savings and unit cost for lanes for each method in order to evaluate the performance of the proposed allocation methods and the alternative cost allocation methods. The unit cost of a lane represents the ratio of the allocated cost to the length of the lane. We calculate this unit cost by Eq. (46) and the percentage cost savings by Eq. (47) with given cost allocation $\tilde{w}_l$, fully-loaded cost $\beta_l f_l$ and individual cost $g_l, l \in L$.

$$Uc(l) = \frac{\tilde{w}_l}{\beta_l f_l} \tag{46}$$

$$Pcs(l) = \left(\frac{g_l - \tilde{w}_l}{g_l} \right) \times \%100 \tag{47}$$



(a) The number of stability violations for distance proportional cost allocation (DPCA) and the modified Shapley value (MSV)



(b) The number of stability violations for nucleolus based methods

Fig. 4. The number of stability violations for each cost allocation method.

10. Computational experiments

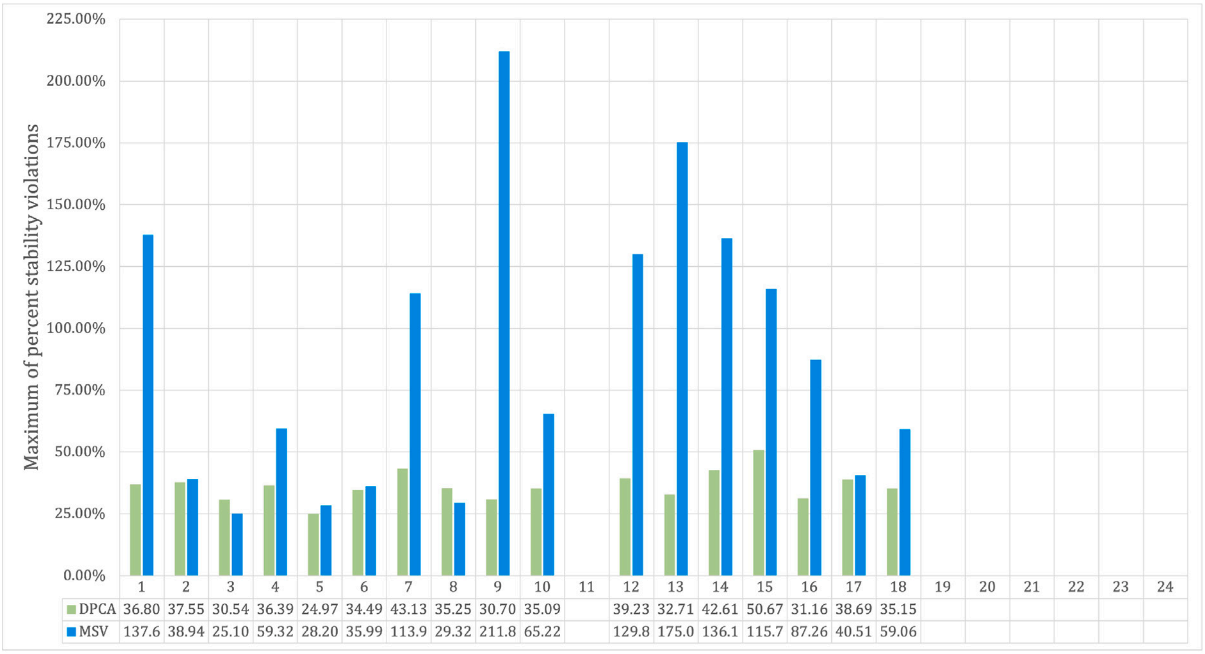
We test our proposed allocation methods, the alternative cost allocation methods and proposed solution approaches on a set of Euclidean problem instances, which were generated randomly using a method described in the previously published works on the LCPs (Ergun et al., 2007a, 2007b; Kuyzu, 2017). The generated instances involve lanes between semi-clustered points scattered over a 1800×1800 miles square region; and a supply chain structure in which the set of the points are partitioned into three classes representing suppliers, plants/warehouses, and customers.

The instances have 100 or 200 points (nodes $|N|$), with a fraction of points in clusters (CPR) equal to 0.5 or 0.8, and with an average number of 10 points per cluster. The density of each instance is regulated with the lane-to-point ratio (ratio of the number of lanes to the number of points) with values one and two. Three different random number seeds for each combination are used to generate the instances. We use a total of 24 instances in our experiments. Detailed information about them can be seen in Table 2. The instances we used can be accessed at http://dx.doi.org/10.17632/8zx5wf2jg2.

In the experiments, we allow at most four lanes in each of the cycles. We require the total length of each to be at most 3850 miles, which is determined such that every cycle can be completed in one week (i.e. 550 miles per day times 7 days per week).

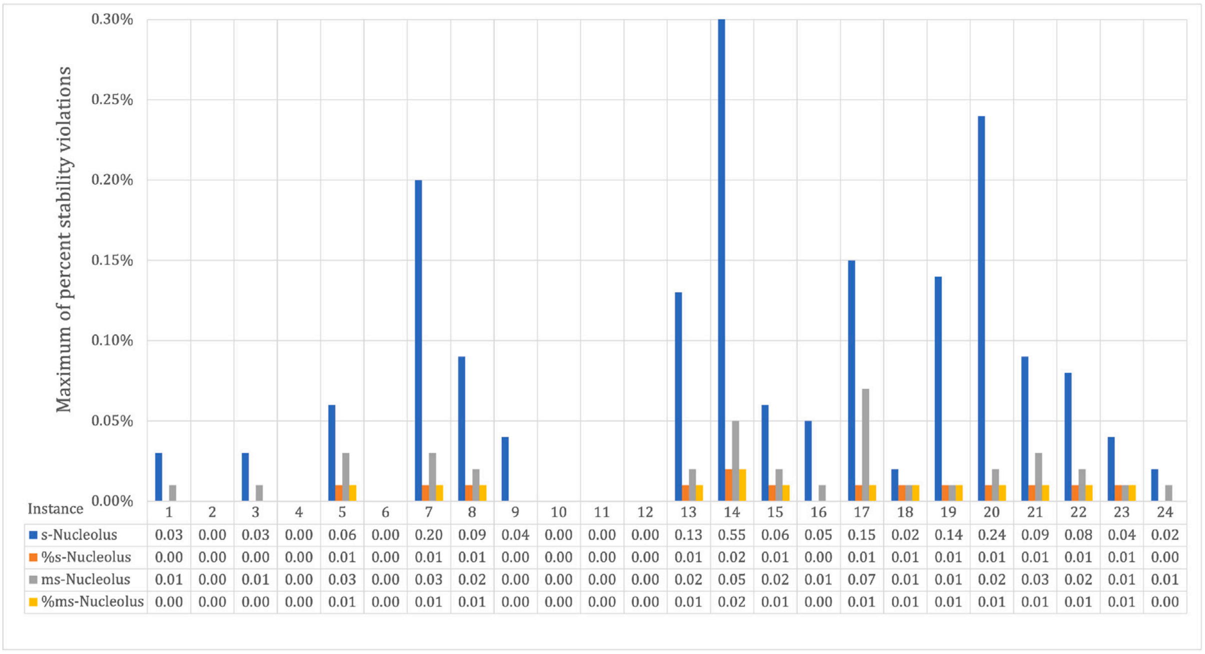
We set the repositioning cost coefficient to $\rho_a = 0.8$ and the movement cost coefficient to $\beta_l = 1.0$ for all of the arcs and the lanes, respectively. We set the standalone cost of each lane to $g_l = 1.8 f_l$. We require each coalition lane to pay at least the difference between its loaded movement cost and empty movement cost, i.e. we set $\pi_l = 0.2$ for all of the lanes.

We have implemented all of the cost allocation methods, mathematical models and our solution procedures using Java programming



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
🟩 DPCA	36.80	37.55	30.54	36.39	24.97	34.49	43.13	35.25	30.70	35.09		39.23	32.71	42.61	50.67	31.16	38.69	35.15						
🟦 MSV	137.6	38.94	25.10	59.32	28.20	35.99	113.9	29.32	211.8	65.22		129.8	175.0	136.1	115.7	87.26	40.51	59.06						

(a) The maximum of percent stability violations for distance proportional cost allocation (DPCA) and the modified Shapley value (MSV)



Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
🟦 s-Nucleolus	0.03	0.00	0.03	0.00	0.06	0.00	0.20	0.09	0.04	0.00	0.00	0.00	0.13	0.55	0.06	0.05	0.15	0.02	0.14	0.24	0.09	0.08	0.04	0.02
🟧 %s-Nucleolus	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.02	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
▨ ms-Nucleolus	0.01	0.00	0.01	0.00	0.03	0.00	0.03	0.02	0.00	0.00	0.00	0.00	0.02	0.05	0.02	0.01	0.07	0.01	0.01	0.02	0.03	0.02	0.01	0.01
🟨 %ms-Nucleolus	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.02	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00

(b) The maximum of percent stability violations for nucleolus based methods

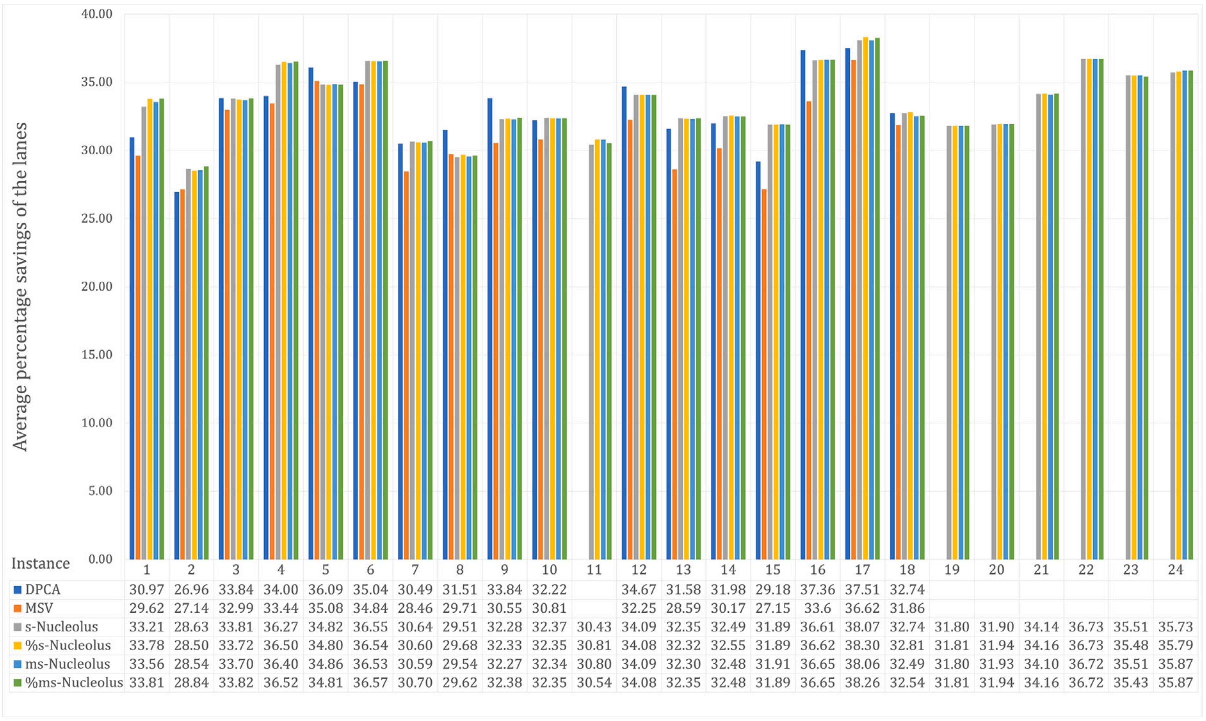
Fig. 5. The maximum of percent stability violations for cost allocation methods.

language. We have run all of the experiments on a workstation with dual 2.00 GHz Intel Xeon E5-2665 processors, 128 GB RAM and Windows Server 2012 Operating System. We have utilized IBM ILOG CPLEX 12.8 solver and Concert technology to solve the linear and integer programs.

10.1. The CGH and the RGA versus the brute force method

Before applying any of the cost allocation methods, we need to solve the SPP. A naive way of solving the SPP is first enumerating the entire set of feasible cycles and then solving the SPP exactly. The resulting solution and the set of cycles can be used in computing the s-nucleolus or any of its variants. We will refer to this approach as the brute force method. We expect the set of feasible cycles to grow exponentially as the problem size grows. Hence, we do not expect the brute force method to be applicable in large-scale problem instances. In this section, we demonstrate how quickly the brute force method runs into scaling issues.

Since we observed that the brute force method did not scale well, we propose running the CGH first to obtain high-quality solutions to the SPP and then compute the s-nucleolus or its variants using the RGA, taking the SPP solution obtained by the CGH as an input. Namely,



Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
DPCA	30.97	26.96	33.84	34.00	36.09	35.04	30.49	31.51	33.84	32.22		34.67	31.58	31.98	29.18	37.36	37.51	32.74						
MSV	29.62	27.14	32.99	33.44	35.08	34.84	28.46	29.71	30.55	30.81		32.25	28.59	30.17	27.15	33.6	36.62	31.86						
s-Nucleolus	33.21	28.63	33.81	36.27	34.82	36.55	30.64	29.51	32.28	32.37	30.43	34.09	32.35	32.49	31.89	36.61	38.07	32.74	31.80	31.90	34.14	36.73	35.51	35.73
%s-Nucleolus	33.78	28.50	33.72	36.50	34.80	36.54	30.60	29.68	32.33	32.35	30.81	34.08	32.32	32.55	31.89	36.62	38.30	32.81	31.81	31.94	34.16	36.73	35.48	35.79
ms-Nucleolus	33.56	28.54	33.70	36.40	34.86	36.53	30.59	29.54	32.27	32.34	30.80	34.09	32.30	32.48	31.91	36.65	38.06	32.49	31.80	31.93	34.10	36.72	35.51	35.87
%ms-Nucleolus	33.81	28.84	33.82	36.52	34.81	36.57	30.70	29.62	32.38	32.35	30.54	34.08	32.35	32.48	31.89	36.65	38.26	32.54	31.81	31.94	34.16	36.72	35.43	35.87

Fig. 6. Average cost savings of the lanes for cost allocations methods.

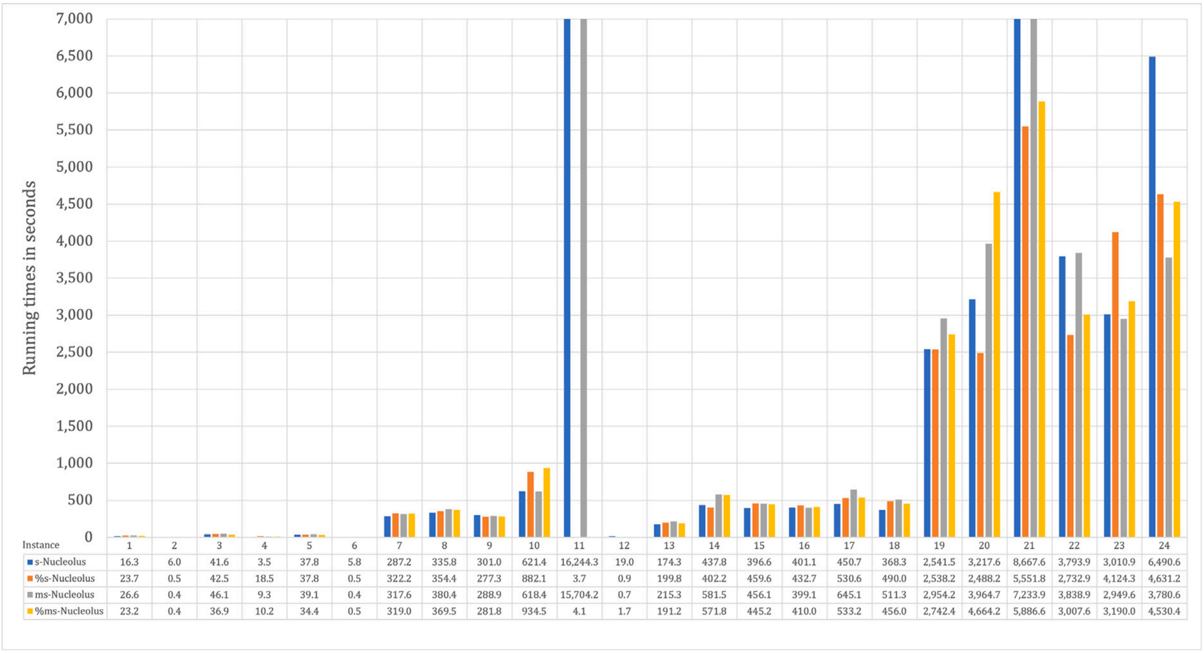
we propose to combine the CGH with the RGA for computing the s-nucleolus and its variants, as discussed above. In this section, we compare the performance of the combination of the CGH and the RGA against the performance of the brute force method.

10.1.1. *The CGH versus the brute force method in solving the SPP*

Table 3 provides a summary of the results we obtain when solving the SPP by the brute force method and the CGH. We observe that most of the instances with 100 and 200 lanes (i.e. 1–18) can be solved to optimality fairly quickly by the brute force method. However, the brute force method cannot find any feasible solutions for the instances with 400 lanes and one instance with 200 lanes, which are 7 instances in total, because it runs out of memory when enumerating cycles despite the massive amount of RAM available on our workstation. The instances with 400 lanes require more computational effort than the others to generate all feasible cycles. On the other hand, the CGH can find good solutions in reasonable time for all of the instances. In the fourth column, we present the difference between the brute force method and the CGH in terms of the objective value of the SPP. The CGH yields high-quality solutions for the 17 instances for which we can find the optimal solution. We also present the objective function values of the brute force and the CGH for each instance in Fig. 2. We give the difference between the number of cycles generated in the methods ($|C(L)^{SPP}|$) in columns five and six. Note that $C(L)^{SPP} = C(L)$ for the brute force method, and $C(L)^{SPP} = C(L)^{col}$ for the CGH. As expected, the difference between the number of cycles generated by the brute force method and the CGH is huge. While the maximum number of generated cycles is nearly 5.5 million for the brute force (can be seen in Fig. 3(a)), the maximum number is around 61.6 thousand for the CGH (can be seen in Fig. 3(b)). The times reported in the eighth and ninth columns show the total time spent by the brute force method and the CGH. The process time of the brute force method includes enumerating all of the feasible cycles in seconds and the time spent on solving the SPP. The process time of the CGH includes generating all of the feasible cycles with at most two lanes in seconds, and the time spent on solving the SPP. While the brute force method terminates more quickly with a solution on 15 instances, the CGH terminates more quickly on nine instances. We also present in the table the number of cycles selected for covering the lanes ($|\hat{C}(L)|$) in final solutions of both methods. The results imply that the average number of lanes in the selected cycles is more than two for both methods. In the last two columns, we give the number of lanes covered by a cycle having more than one lane ($|\hat{L}|$). We observe that the proportion of lanes that are covered by a cycle having more than one lane (calculated as $|\hat{L}|/|L|$, not reported in the table) in the SPP solution exceeds 94% in all of the instances for both of the two methods. Overall, we can conclude that we can solve the SPP efficiently by generating a small portion of the set of feasible cycles.

10.1.2. *The CGH and the RGA versus the brute force method in computing the s-nucleolus*

We compute the s-nucleolus using both of the CGH-RGA combination and the brute force method. We present our results in Table 4, which includes the number of violated stability constraints, total process time, the maximum of percent stability violations and the average of unit allocated cost for both the brute force method and the RGA. The number of violated stability constraints is zero for three of 24 instances for both methods. In line with Proposition 2, the linear relaxation of the SPP yields an integer feasible solution for these three instances. We show that, when the CGH yields optimal solution for the SPP, the RGA also gives optimal solution for the s-nucleolus, as expected. When the CGH does not yield and optimal solution to the SPP, the number of violated stability constraints of the RGA is a little higher than that of the brute force method. But the maximum difference between the brute force method and the RGA in terms of the number of violated stability constraints is just 23 cycles for instance 8. The higher objective value of the CGH solution leads to this difference. Since the s-nucleolus has a budget-balance constraint which means the total allocated cost will be higher than the brute force method. Note that the brute force method cannot identify the exact s-nucleolus for instance 10 due to the 12-h time limit we impose. Consequently, the RGA obtains a better solution for that instance. The process time to calculate the s-nucleolus in the RGA is shorter than the brute force method for 21 instances. The brute force method terminates in less time for just three instances. Similar to the number of violated stability constraints, the maximum of percent



Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
■ s-Nucleolus	16.3	6.0	41.6	3.5	37.8	5.8	287.2	335.8	301.0	621.4	16,244.3	19.0	174.3	437.8	396.6	401.1	450.7	368.3	2,541.5	3,217.6	8,667.6	3,793.9	3,010.9	6,490.6
■ %s-Nucleolus	23.7	0.5	42.5	18.5	37.8	0.5	322.2	354.4	277.3	882.1	3.7	0.9	199.8	402.2	459.6	432.7	530.6	490.0	2,538.2	2,488.2	5,551.8	2,732.9	4,124.3	4,631.2
■ ms-Nucleolus	26.6	0.4	46.1	9.3	39.1	0.4	317.6	380.4	288.9	618.4	15,704.2	0.7	215.3	581.5	456.1	399.1	645.1	511.3	2,954.2	3,964.7	7,233.9	3,838.9	2,949.6	3,780.6
■ %ms-Nucleolus	23.2	0.4	36.9	10.2	34.4	0.5	319.0	369.5	281.8	934.5	4.1	1.7	191.2	571.8	445.2	410.0	533.2	456.0	2,742.4	4,664.2	5,886.6	3,007.6	3,190.0	4,530.4

Fig. 7. The running times of the nucleolus based cost allocation methods.

stability violations of the CGH is higher than that of the brute force method when the CGH cannot find and optimal solution to the SPP. We observe that the cost allocation obtained by the RGA has a higher unit allocated cost value on only three instances. We can conclude that our solution procedure combining the CGH and the RGA performs well in terms of the number of violated stability constraints and process time, while the brute force method cannot find any solution to the s-nucleolus for 7 instances and has the highest process time for the process time for 21 instances.

10.1.3. *The CGH and the RGA versus the brute force method in terms of process time*

Our objective is to calculate the s-nucleolus for the SPP. In order to do that, we need to solve both the SPP and the s-nucleolus. We present the total process time for solving both the SPP and the s-nucleolus in Table 5. While the brute force method has the lowest process time for 8 instances, the CGH and the RGA have the lowest process time for 16 instances in overall. For the large instance, the brute force method needs the more computational time and effort than the CGH and the RGA. We can conclude that our proposed solution approach combining the CGH and the RGA yields good solutions in terms of the total process time for large instances. Therefore, we apply this approach for the s-nucleolus and its variants in the computational experiments discussed in the remainder of this paper.

10.2. *Comparison of cost allocation methods*

After obtaining a solution to the SPP, we apply each of the cost allocation methods s-nucleolus, ms-nucleolus, %s-nucleolus, %ms-nucleolus, distance proportional cost allocation, and the modified Shapley value for comparison. We use the CGH-RGA combination to compute the s-nucleolus and its variants. For distance proportional cost allocation and the modified Shapley value, we first obtain a solution to the SPP using the brute force method, and then apply the respective one of the two, in order to be able to run our stability assessment procedure efficiently.

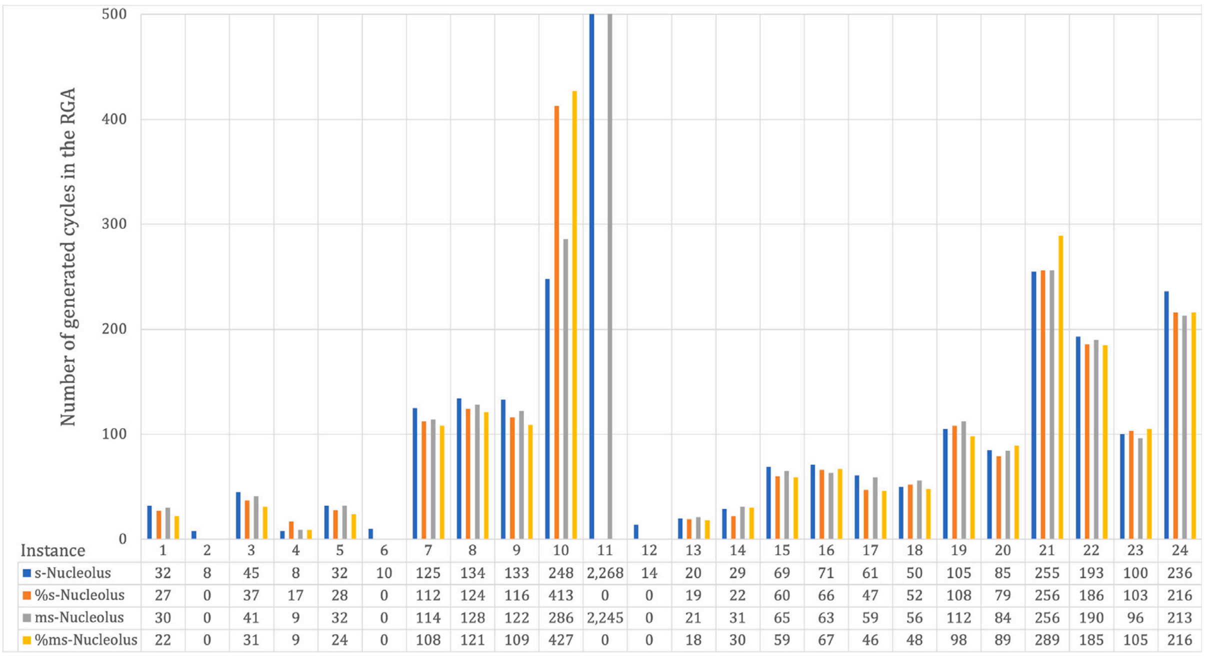
We report the number of violated stability constraints, and the average and the maximum of percent stability violations. Additionally, we report statistics on the minimum, the average and the maximum of unit allocated cost for lanes, i.e. cost per loaded unit distance to be paid by the shippers, and the average of the percentage cost savings for lanes.

10.2.1. *Stability violations*

We assess the stability of the grand coalition using the procedure described in Section 9. We run our stability assessment procedure on the costs allocated by each of the above methods. Table 6 contains a summary of our results. We also present the number of violated stability constraints and the maximum of percent stability violations in Figs. 4 and 5. We note that the maximum of percent stability violations is as high as 50.67 percent for distance proportional cost allocation, 211.81 percent for the modified Shapley value, 0.55 percent for the s-Nucleolus, 0.02 for the %s-Nucleolus, 0.07 for the ms-nucleolus, and 0.02 for the %ms-nucleolus, respectively. The number of violated stability constraints is quite high for distance proportional cost allocation and the modified Shapley value. The %s-nucleolus has the lowest number of violated stability constraints. The average of percent stability violations is very low in all of our proposed cost allocation methods. We observe that the %s-nucleolus, the ms-nucleolus and the %ms-nucleolus have low maximum of percent stability violations while the s-nucleolus has high maximum of percent stability violations. We can conclude that our proposed cost allocation methods yield better cost allocations in terms of the number of violated stability constraints, the average and maximum of percent stability violations.

10.2.2. *Unit allocated cost*

An important metric for the shippers is the cost they will be charged per loaded mile (or kilometer) by the carriers. In Table 7, we present the minimum, the average and the maximum of the unit allocated cost of the lanes included in the coalition given by the cost allocation approaches we test for the grand coalition. We observe that the modified Shapley value has the highest maximum unit allocated cost for all instances since it does not have any upper bound on the allocated cost. In fact, the cost allocated by the modified Shapley value exceeds the standalone cost for some of the lanes. Our proposed cost allocation methods perform similarly in terms of minimum, average and maximum unit allocated cost on average. The minimum unit allocated cost value of distance proportional cost allocation is higher than the others, indicating a more balanced cost allocation.



Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
🟦 s-Nucleolus	32	8	45	8	32	10	125	134	133	248	2,268	14	20	29	69	71	61	50	105	85	255	193	100	236
🟧 %s-Nucleolus	27	0	37	17	28	0	112	124	116	413	0	0	19	22	60	66	47	52	108	79	256	186	103	216
⬜ ms-Nucleolus	30	0	41	9	32	0	114	128	122	286	2,245	0	21	31	65	63	59	56	112	84	256	190	96	213
🟨 %ms-Nucleolus	22	0	31	9	24	0	108	121	109	427	0	0	18	30	59	67	46	48	98	89	289	185	105	216

Fig. 8. The number of generated cycles in the RGA.

Table 7
Unit allocated cost of the lanes for cost allocation methods.

Instance	DPCA			MSV			s-Nucleolus			%s-Nucleolus			ms-Nucleolus			%ms-Nucleolus		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	1.01	1.24	1.80	0.93	1.27	4.28	0.22	1.20	1.80	0.24	1.19	1.80	0.22	1.20	1.80	0.24	1.19	1.80
2	1.00	1.31	1.80	0.93	1.31	2.21	0.23	1.28	1.80	0.24	1.29	1.80	0.24	1.29	1.80	0.24	1.28	1.80
3	1.02	1.19	1.80	0.87	1.21	2.06	0.22	1.19	1.80	0.23	1.19	1.80	0.23	1.19	1.80	0.21	1.19	1.80
4	1.01	1.19	1.80	0.87	1.20	2.87	0.22	1.15	1.80	0.23	1.14	1.80	0.23	1.14	1.80	0.25	1.14	1.80
5	1.01	1.15	1.80	0.88	1.17	2.01	0.23	1.17	1.80	0.23	1.17	1.80	0.23	1.17	1.80	0.23	1.17	1.80
6	1.00	1.17	1.80	0.92	1.17	1.80	0.36	1.14	1.80	0.39	1.14	1.80	0.35	1.14	1.80	0.40	1.14	1.80
7	1.00	1.25	1.80	0.87	1.29	3.85	0.23	1.25	1.80	0.22	1.25	1.80	0.23	1.25	1.80	0.22	1.25	1.80
8	1.00	1.23	1.80	0.95	1.27	2.33	0.21	1.27	1.80	0.21	1.27	1.80	0.21	1.27	1.80	0.21	1.27	1.80
9	1.00	1.19	1.80	0.86	1.25	5.61	0.46	1.22	1.80	0.45	1.22	1.80	0.41	1.22	1.80	0.45	1.22	1.80
10	1.00	1.22	1.80	0.89	1.25	2.97	0.26	1.22	1.80	0.26	1.22	1.80	0.26	1.22	1.80	0.26	1.22	1.80
11	–	–	–	–	–	–	0.23	1.25	1.80	0.23	1.25	1.80	0.23	1.25	1.80	0.23	1.25	1.80
12	1.00	1.18	1.80	0.79	1.22	2.68	0.22	1.19	1.80	0.22	1.19	1.80	0.22	1.19	1.80	0.20	1.19	1.80
13	1.00	1.23	1.80	0.90	1.29	4.95	0.21	1.22	1.80	0.21	1.22	1.80	0.21	1.22	1.80	0.21	1.22	1.80
14	1.01	1.22	1.80	0.76	1.26	4.25	0.21	1.22	1.81	0.21	1.21	1.80	0.21	1.22	1.80	0.21	1.22	1.80
15	1.01	1.27	1.80	0.83	1.31	3.51	0.22	1.23	1.80	0.22	1.23	1.80	0.22	1.23	1.80	0.22	1.23	1.80
16	1.00	1.13	1.80	0.83	1.20	3.29	0.26	1.14	1.80	0.30	1.14	1.80	0.26	1.14	1.80	0.26	1.14	1.80
17	1.00	1.12	1.80	0.90	1.14	2.11	0.21	1.11	1.80	0.21	1.11	1.80	0.22	1.12	1.80	0.22	1.11	1.80
18	1.01	1.21	1.80	0.83	1.23	2.66	0.21	1.21	1.80	0.21	1.21	1.80	0.21	1.22	1.80	0.21	1.21	1.80
19	–	–	–	–	–	–	0.21	1.23	1.80	0.21	1.23	1.80	0.21	1.23	1.80	0.21	1.23	1.80
20	–	–	–	–	–	–	0.22	1.23	1.80	0.23	1.23	1.80	0.23	1.23	1.80	0.23	1.23	1.80
21	–	–	–	–	–	–	0.23	1.19	1.80	0.23	1.19	1.80	0.23	1.19	1.80	0.23	1.19	1.80
22	–	–	–	–	–	–	0.22	1.14	1.80	0.22	1.14	1.80	0.22	1.14	1.80	0.22	1.14	1.80
23	–	–	–	–	–	–	0.20	1.16	1.80	0.20	1.16	1.80	0.20	1.16	1.80	0.20	1.16	1.80
24	–	–	–	–	–	–	0.23	1.16	1.80	0.22	1.16	1.80	0.22	1.15	1.80	0.22	1.15	1.80
Average	**1.00**	**1.21**	**1.80**	**0.87**	**1.24**	**3.14**	**0.24**	**1.20**	**1.80**	**0.24**	**1.20**	**1.80**	**0.24**	**1.20**	**1.80**	**0.24**	**1.20**	**1.80**

DPCA: Distance Proportional Cost Allocation, MSV: Modified Shapley Value.

10.2.3. Cost savings

We provide the average percentage cost savings of lanes for all cost allocation methods in Table 8 and Fig. 6 for the grand coalition. While the %s-nucleolus and the %ms-nucleolus have the highest percentage savings on average, the s-Nucleolus and the ms-nucleolus have the second highest percentage savings on average. We observe that the modified Shapley value has the lowest percentage cost savings for the lanes. We can conclude that using the percentage stability violation instead of the stability violation amount in the nucleolus-based methods yields better cost allocations in terms of average percentage cost savings of the lanes. Note that we cannot find any feasible solution for seven instances due to running out of memory when generating all feasible cycles. Because of that, there are 17 solutions for distance proportional cost allocation, the modified Shapley value and the brute force method.

Table 8
Average percentage cost savings of the lanes for cost allocation methods.

Instances	DPCA	MSV	BF	s-Nucleolus	%s-Nucleolus	ms-Nucleolus	%ms-Nucleolus
1	30.97	29.62	33.36	33.21	33.78	33.56	33.81
2	26.96	27.14	28.46	28.63	28.50	28.54	28.84
3	33.84	32.99	33.76	33.81	33.72	33.70	33.82
4	34.00	33.44	36.43	36.27	36.50	36.40	36.52
5	36.09	35.08	34.78	34.82	34.80	34.86	34.81
6	35.04	34.84	36.53	36.55	36.54	36.53	36.57
7	30.49	28.46	30.60	30.64	30.60	30.59	30.70
8	31.51	29.71	29.86	29.51	29.68	29.54	29.62
9	33.84	30.55	32.34	32.28	32.33	32.27	32.38
10	32.22	30.81	32.47	32.37	32.35	32.34	32.35
11	–	–	–	30.43	30.81	30.80	30.54
12	34.67	32.25	34.06	34.09	34.08	34.09	34.08
13	31.58	28.59	32.31	32.35	32.32	32.30	32.35
14	31.98	30.17	32.48	32.49	32.55	32.48	32.48
15	29.18	27.15	32.25	31.89	31.89	31.91	31.89
16	37.36	33.6	36.61	36.61	36.62	36.65	36.65
17	37.51	36.62	37.99	38.07	38.30	38.06	38.26
18	32.74	31.86	32.55	32.74	32.81	32.49	32.54
19	–	–	–	31.80	31.81	31.80	31.81
20	–	–	–	31.90	31.94	31.93	31.94
21	–	–	–	34.14	34.16	34.10	34.16
22	–	–	–	36.73	36.73	36.72	36.72
23	–	–	–	35.51	35.48	35.51	35.43
24	–	–	–	35.73	35.79	35.87	35.87
Average	**32.94**	**31.35**	**33.34**	**33.44**	**33.50**	**33.46**	**33.51**

DPCA: Distance Proportional Cost Allocation
MSV: Modified Shapley Value
BF: Brute Force.

Table 9
Running times of cost allocation methods in seconds.

Instances	DPCA	MSV	BF	s-Nucleolus	%s-Nucleolus	ms-Nucleolus	%ms-Nucleolus
1	0.10	0.09	6.67	16.27	23.70	26.59	23.17
2	0.15	0.14	7.12	5.95	0.47	0.42	0.43
3	0.10	0.08	43.80	41.55	42.53	46.05	36.86
4	0.10	0.09	3.88	3.52	18.52	9.30	10.22
5	0.14	0.13	44.20	37.77	37.80	39.09	34.42
6	0.17	0.15	6.62	5.81	0.46	0.44	0.46
7	0.22	0.22	2,413.66	287.23	322.22	317.57	318.96
8	0.27	0.25	5,834.70	335.83	354.40	380.40	369.47
9	0.52	0.30	2,083.07	300.99	277.26	288.87	281.80
10	1.17	2.03	58,453.21	621.44	882.08	618.40	934.52
11	–	–	–	16,244.26	3.67	15,704.15	4.09
12	1.10	1.12	193.01	18.99	0.86	0.68	1.73
13	0.42	0.48	1,901.59	174.30	199.77	215.29	191.17
14	0.24	0.23	294.71	437.75	402.18	581.45	571.81
15	1.7	1.59	16,606.13	396.55	459.57	456.05	445.16
16	0.4	0.43	5,630.07	401.14	432.68	399.12	410.01
17	0.27	0.26	1,120.44	450.67	530.58	645.10	533.16
18	0.42	0.39	206.88	368.31	490.03	511.32	456.00
19	–	–	–	2,541.47	2538.18	2,954.20	2742.36
20	–	–	–	3,217.64	2488.20	3,964.66	4664.21
21	–	–	–	8,667.56	5551.84	7,233.93	5886.63
22	–	–	–	3,793.87	2732.90	3,838.94	3007.60
23	–	–	–	3,010.86	4124.31	2,949.55	3189.96
24	–	–	–	6,490.60	4631.16	3,780.62	4530.38
Average	**0.44**	**0.47**	**5,579.40**	**1,994.60**	**1106.06**	**1,873.42**	**1193.52**

DPCA: Distance Proportional Cost Allocation
MSV: Modified Shapley Value
BF: Brute Force.

Table 10
The number of cycles generated $C(L)^{row}$ in the RGA for nucleolus-based models.

Instances	s-Nucleolus	%s-Nucleolus	ms-Nucleolus	%ms-Nucleolus
1	32	27	30	22
2	8	0	0	0
3	45	37	41	31
4	8	17	9	9
5	32	28	32	24
6	10	0	0	0
7	125	112	114	108
8	134	124	128	121
9	133	116	122	109
10	248	413	286	427
11	2268	0	2245	0
12	14	0	0	0
13	20	19	21	18
14	29	22	31	30
15	69	60	65	59
16	71	66	63	67
17	61	47	59	46
18	50	52	56	48
19	105	108	112	98
20	85	79	84	89
21	255	256	256	289
22	193	186	190	185
23	100	103	96	105
24	236	216	213	216
Average	**180**	**87**	**177**	**88**

10.2.4. Running time

We report the running times of the cost allocation methods in Table 9 and in Fig. 7 for the grand coalition. We cut the y-axis at 7000 to make the other values more visible. However, all values in Table 9 can be seen at the bottom of Fig. 7. Note that the reported numbers correspond to only the running times of the cost allocation methods. Distance proportional cost allocation and the modified Shapley value terminate well under a second. Except for the brute force method, we employ our proposed nucleolus-based cost allocation methods without any time limit over the limited set of feasible cycles after solving the SPP by the CGH. On the other hand, the cost allocation phase of the brute force method runs under a 12-h time limit over the set of all feasible cycles it generates when solving the SPP. We observe that the nucleolus-based methods require much more computational effort than distance proportional cost allocation and the modified Shapley value as expected. We also observe that the proposed RGA yields better solution for the nucleolus based models in terms of running time. The %s-nucleolus has the lowest running time among the others while the s-nucleolus has the highest running time. Their running times on 100-lane and 200-lane instances are generally less than those on 400-lane instances. The same conclusion for the average percentage savings of the lanes is also valid for the running time. We can conclude that using the percentage stability violation instead of the stability violation amount in the nucleolus gives better cost allocation solutions in terms of the running time. However, using minimum savings requirements and lower bounds on allocated costs leads to increase the running time with both the percentage stability violation and the stability violation amount. Overall, we can find better solutions with the help of the CGH and the RGA for all of the 24 instances in reasonable time.

10.2.5. Number of cycles generated

We present the number of cycles generated in the RGA for all nucleolus based models in Table 10 and Fig. 8. Note that we cut the y-axis at 500 in Fig. 8 to make it more readable. However, all values corresponding to the y-axis can be seen at the bottom of the figure. We show that we can generate fewer rows (cycles) in the RGA for percent stability violation than violation amount. The %s-nucleolus has the lowest number of generated cycles while the %ms-nucleolus has the second-lowest number on average. The results show that using percent stability violation decreases the number of generated cycles which means that it needs less computational effort.

11. Conclusions

In this paper, we study nucleolus-based cost allocation methods for a class of constrained LCGs in which constraints arising from practical considerations are imposed on the cycles which can be used in the cover. We discuss the necessary and sufficient conditions for such constrained lane covering games to have a non-empty core. These conditions imply that an arbitrarily selected set of lanes may have an empty core. We develop nucleolus-based cost allocation methods to identify a cost allocation with as little stability violation as possible without incurring a budget deficit. We incorporate minimum savings and minimum payment requirements into our cost allocation methods. We also modify the objective function in the nucleolus to lexicographically minimize the maximum percent stability violation instead of the maximum absolute stability violation. We show that using percent stability violation instead of stability violation amount yields better solutions in terms of running time. Additionally, we propose column and row generation procedures to solve the mathematical models.

Our computational experiments demonstrate that the cost allocations obtained by the proposed methods violate the stability condition in smaller amount and lower percentage. We show that the alternative cost allocation methods (distance proportional cost allocation and the modified Shapley value) violate the stability condition for most of the cycles and have a high percentage of stability violations. In addition, our experiments show that the solution procedures based on column and row generation yield good solutions with much less computational effort than the brute force method.

CRediT authorship contribution statement

Nihat Öner: Software, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft. **Gültekin Kuyzu:** Conceptualization, Methodology, Formal analysis, Resources, Supervision, Writing – review & editing.

Acknowledgment

The research presented in this paper was partially supported by the Scientific and Technological Research Council of Turkey (TUBITAK) through grant 112M227.

References

Audy, J.-F., D'Amours, S., & Rousseau, L.-M. (2010). Cost allocation in the establishment of a collaborative transportation agreement—an application in the furniture industry. *Journal of the Operational Research Society*, *62*(6), 960–970.

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

Chardaire, P. (2001). The core and nucleolus of games: A note on a paper by Göthe-Lundgren others. *Mathematical Programming*, *90*(1), 147–151.

De Vos, B., & Raa, B. (2018). Stability analysis of cost allocation methods for inventory routing. *IFAC-PapersOnLine*, *51*(11), 1682–1688.

Defryn, C., Sörensen, K., & Cornelissens, T. (2016). The selective vehicle routing problem in a collaborative environment. *European Journal of Operational Research*, *250*(2), 400–411.

Ergun, O., Kuyzu, G., & Savelsbergh, M. (2007a). Reducing truckload transportation costs through collaboration. *Transportation Science*, *41*(2), 206–221, 00083.

Ergun, O., Kuyzu, G., & Savelsbergh, M. (2007b). Shipper collaboration. *Computers & Operations Research*, *34*(6), 1551–1560.

Erhun, F., & Keskinocak, P. (2011). Collaborative supply chain management. In K. G. Kempf, P. Keskinocak, & R. Uzsoy (Eds.), *International series in operations research & management science*: *Vol. 151, Planning production and inventories in the extended enterprise* (pp. 233–268). Springer US.

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

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

Guajardo, M., & Jörnsten, K. (2015). Common mistakes in computing the nucleolus. *European Journal of Operational Research*, *241*(3), 931–935, 00015.

Hadas, Y., Gnecco, G., & Sanguineti, M. (2017). An approach to transportation network analysis via transferable utility games. *Transportation Research, Part B (Methodological)*, *105*, 120–143.

Hezarkhani, B., Slikker, M., & Van Woensel, T. (2014). On characterization of the core of lane covering games via dual solutions. *Operations Research Letters*, *42*(8), 505–508.

Jain, K., & Mahdian, M. (2007). Cost sharing. In N. Nisan, T. Roughgarden, E. Tardos, & V. V. Vazirani (Eds.), *Algorithmic game theory* (pp. 385–410). Cambridge University Press.

Kimms, A., & Kozeletskyi, I. (2016). Core-based cost allocation in the cooperative traveling salesman problem. *European Journal of Operational Research*, *248*(3), 910–916.

Kuyzu, G. (2017). Lane covering with partner bounds in collaborative truckload transportation procurement. *Computers & Operations Research*, *77*, 32–43.

Lozano, S., Moreno, P., Adenso-Díaz, B., & Algaba, E. (2013). Cooperative game theory approach to allocating benefits of horizontal cooperation. *European Journal of Operational Research*, *229*(2), 444–452.

Lu, W., & Quadrifoglio, L. (2019). Fair cost allocation for ridesharing services – modeling, mathematical programming and an algorithm to find the nucleolus. *Transportation Research, Part B (Methodological)*, *121*, 41–55.

Özener, O., & Ergun, O. (2008). Allocating costs in a collaborative transportation procurement network. *Transportation Science*, *42*(2), 146–165.

Özener, O. O., Ergun, O., & Savelsbergh, M. (2013). Allocating cost of service to customers in inventory routing. *Operations Research*, *61*(1), 112–125.

Pál, M., & Tardos, E. (2003). Group strategy proof mechanisms via primal-dual algorithms. In *Foundations of computer science, 2003. Proceedings. 44th annual IEEE symposium on* (pp. 584–593). IEEE.

Rasulkhani, S., & Chow, J. Y. (2019). Route-cost-assignment with joint user and operator behavior as a many-to-one stable matching assignment game. *Transportation Research, Part B (Methodological)*, *124*, 60–81.

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

Shapley, L. S. (1953). A value for n-person games. In H. W. Kuhn, & A. W. Tucker (Eds.), *Annals of mathematical studies*: *Vol. 28*, *Contributions to the theory of games II* (pp. 307–317). Princeton University Press.

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

Standing, C., Standing, S., & Biermann, S. (2019). The implications of the sharing economy for transport. *Transport Reviews*, *39*(2), 226–242.

Tae, H., Kim, B.-I., & Park, J. (2020). Finding the nucleolus of the vehicle routing game with time windows. *Applied Mathematical Modelling*, *80*, 334–344.

Vanovermeire, C., & Sörensen, K. (2014a). Integration of the cost allocation in the optimization of collaborative bundling. *Transportation Research Part E: Logistics and Transportation Review*, *72*, 125–143, 00000.

Vanovermeire, C., & Sörensen, K. (2014b). Measuring and rewarding flexibility in collaborative distribution, including two-partner coalitions. *European Journal of Operational Research*, *239*(1), 157–165, 00000.

Wang, X., Agatz, N., & Erera, A. (2017). Stable matching for dynamic ride-sharing systems. *Transportation Science*, *52*(4), 850–867.

Wang, H., & Yang, H. (2019). Ridesourcing systems: A framework and review. *Transportation Research, Part B (Methodological)*, *129*, 122–155.

Wang, J., Yu, Y., & Tang, J. (2018). Compensation and profit distribution for cooperative green pickup and delivery problem. *Transportation Research, Part B (Methodological)*, *113*, 54–69.

Xu, H., Chen, Z.-L., Rajagopal, S., & Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, *37*(3), 347–364.

Young, H. P. (1985). *Cost allocation: methods, principles, applications*. North Holland Publishing Co..