Transportation Research Part E 154 (2021) 102447

Contents lists available at ScienceDirect

Transportation Research Part E

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

Core stable coalition selection in collaborative truckload transportation procurement

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, USA

ARTICLE INFO

Keywords:
Collaborative logistics
Cooperative game
Stable coalition selection
Column and row generation
Branch-and-cut-and-price

ABSTRACT

We study the problem of forming a core stable coalition that minimizes system-wide cost, given a set of candidate full truckload shipment lanes, which we model as players, and a characteristic function that corresponds to solving an NP-Hard cost minimizing lane covering problem. Since not every coalition has a non-empty core in this setting, we formulate a mixed integer linear program that can identify a core stable coalition with minimal system-wide cost along with a cost allocation in the core. We propose a solution method that embeds row generation, column generation, and an upper bounding heuristic into branch-and-bound, which can be considered as a form of branch-and-cut-and-price. We evaluate the performance of our solution method through extensive numerical experiments on randomly generated problem instances.

1. Introduction

Collaboration offers opportunities for individuals and businesses to achieve results that surpass those of standalone efforts. Selecting members, identifying the best operating structure, and sharing the benefits are crucial decisions in any collaborative setting. These decisions must be made with the goal of maximizing the gains of the members while ensuring the stability of the system and thus the persistence of its gains.

The literature on transferable utility games includes many proposed concepts; such as stable matchings, the core, stable sets, and coalition formation; with applications to a plethora of problem settings, that can address the aforementioned three decisions to varying degrees. In a *stable matching* (Gale and Shapley, 1962; Gale and Sotomayor, 1985), where the gain of each player from each possible matching is known beforehand, no pair (or set) of players can be mutually better off by forming their own matching outside the given matching structure. In a *core stable* coalition (Banerjee et al., 2001), where the set of players and the characteristic (i.e., gain) function defined on each subcoalition (including the grand coalition itself) is taken as input, there exists a budget-balanced gain sharing mechanism ensuring that no subcoalition is better off by breaking away from the coalition. That is, a core stable coalition has a *non-empty core*. Note that core stability focuses on internal stability of the coalition. A *stable set* (Anesi, 2010) is a coalition that is both internally and externally stable. Coalition formation (Ray and Vohra, 2015) commonly focuses on forming one or more stable coalitions given a set of candidate players. It may span all three of the decisions or assume a pre-determined gain sharing method. Researchers analyzed various forms of collaboration in logistics in light of game theory concepts. In this paper, we focus on core stability as it is the most commonly used concept in existing studies focusing on horizontal collaboration in logistics.

* Corresponding author at: Convoy Inc., Seattle, WA, USA.
E-mail address: gkuyzu@etu.edu.tr (G. Kuyzu).

https://doi.org/10.1016/j.tre.2021.102447
Received 5 November 2020; Received in revised form 4 July 2021; Accepted 8 August 2021

In logistics collaborations, gains are mostly in the form of cost savings, so the characteristic functions are usually cost functions and the gain allocation is replaced with cost savings when analyzing stability. Furthermore, the characteristic function rarely has a closed-form expression, and evaluating it amounts to solving a cost minimizing optimization problem in most cases. Although the body of literature applying game theory concepts to logistics collaborations is growing, we have run into far fewer papers on coalition formation than those on cost allocation.

In this paper, we study the problem of selecting a core stable coalition that minimizes system-wide cost, given a set of candidate full truckload (FTL) shipment lanes, which we model as players, and a characteristic function that corresponds to solving an NP-Hard cost minimizing lane covering problem. We take the perspective of a centralized planner such as a consulting company or an online platform that coordinates horizontal collaboration among shippers, i.e. companies that procure the services of FTL carriers through outsourcing. We model the lanes as the players because a shipper may decide to remove a subset of its lanes from the collaboration (Özener and Ergun, 2008), and a shipper is not likely to have multiple lanes on a collaborative route. Note that not every coalition has a non-empty core in this setting. Although the problem we are studying can be classified as a coalition formation problem, we prefer to call it as a coalition selection problem, since we focus on forming a core stable coalition with maximum cost savings by selecting lanes from a given set of candidate lanes. We define the selected coalition as the set of lanes that are included in the collaboration. The set of lanes included in the collaboration can take advantage of the increased buying power, inter-lane synergy, and the decision technology of the collaboration, while the excluded lanes cannot. We first formulate an integer program, which we call the coalition selection integer program (CSIP), that can identify a core stable coalition with minimal system-wide cost with a corresponding cost allocation in the core. We then develop a branch-and-price solution method that combines column generation, row generation, and upper bounding heuristic procedures for solving the CSIP. We perform extensive numerical experiments on randomly generated problem instances to assess the effectiveness of our solution procedures and the effect of problem parameters on coalition selection.

The main motivation for our study is collaborative truckload transportation procurement (CTTP), first studied by Ergun et al. (2007a,b), which is a form of horizontal logistics collaboration, where full-truckload (FTL) lanes from multiple shippers are combined into closed loop tours to make them attractive to carriers (or other LSPs). Since carriers and shippers want to avoid tours with many legs or very long tours, there are length and cardinality constraints on the tours that can be formed. Identifying a set of tours with minimum total cost, without any coalitional stability consideration, is equivalent to solving the constrained cardinality and length constrained lane covering problem (CLC-LCP), which is an NP-Hard combinatorial optimization problem (Ergun et al., 2007b). Since the identified collaborative tours are outsourced to carriers through a competitive market mechanism, vehicle depots are not of concern when optimizing the tours. Hence, the CLC-LCP can be regarded as an arc routing problem without a depot.

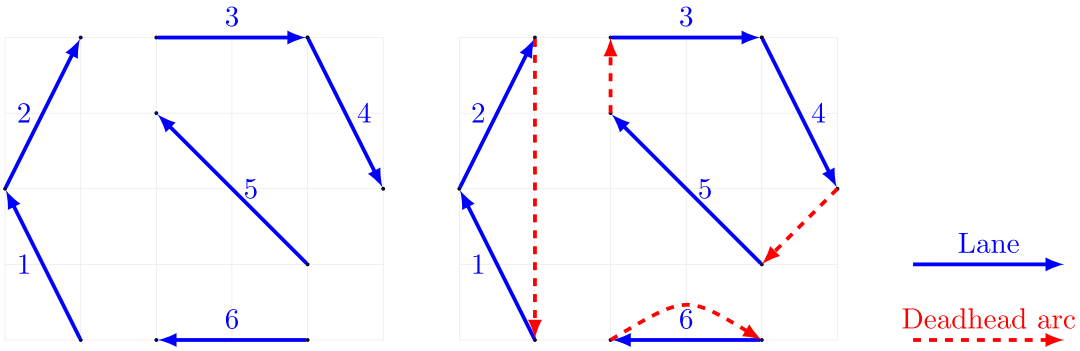
The CLC-LCP can be solved using a set covering (or partitioning) problem. It is well-known that any cooperative game with a set covering (or partitioning) type characteristic function has a non-empty core if and only if the corresponding set covering integer program has zero integrality gap (Bondareva, 1963; Shapley, 1967). The strong condition posed by this well-known result applies to our problem setting as well. As a result, a CTTP coalition consisting of an arbitrarily selected set of lanes paired with an CLC-LCP type characteristic function is probably not core stable. Therefore, minimum system-wide cost and core stability are not easy to achieve at the same time. Without core stability, the coalition may be susceptible to falling apart and ultimately leading to a higher system-wide cost than planned. We can achieve core stability by carefully selecting the lanes to be included in the CTTP lane pool. We take inspiration from maximum weight stable matching games (Wang et al., 2018b) and seek to select a core stable coalition with minimum total system-wide cost. Our goal can be interpreted as finding the right trade-off between core stability and cost. This trade-off has been referred to as the *price of stability* (Anshelevich et al., 2008) or the *cost of stability* (Bachrach et al., 2009) in the literature.

Previously published studies on CTTP either focused on solving the underlying LCPs (Kuyzu, 2017; Kuyzu, 2020) or cost allocation when the underlying LCP leads to an always non-empty core (Özener and Ergun, 2008; Hezarkhani et al., 2014). We contribute to the literature by extending the existing literature on CTTP by tackling the challenging problem of selecting a core stable coalition with minimum system-wide cost and thus with maximum system-wide savings. We also contribute to the relatively scarce literature on coalition formation in logistics and supply chain management. Our study can be adapted to different collaborative settings with different underlying cost functions.

In practice, our model can be used to determine the set of lanes that should be included in real-life collaborative logistics networks, such as the one orchestrated by TRI-VIZOR (Creemers et al., 2017). It can also be used to decide whether a new lane or set of lanes should be added to the collaborative logistics network. The model can also be used by logistics service providers in determining which contracted lanes they should keep in or remove from their networks. A very interesting example is the recently introduced concept of the universal trailer pool, which allows small FTL carriers to serve dedicated power-only (drop-and-hook) shipment lanes. Since the trailer routes must be closed-loop tours that may cover multiple FTL moves by different shippers with minimal empty driving, the underlying optimization problem can be formulated as an LCP variant. Accurately evaluating the set lanes to be covered by the trailer pool and computing the true cost of serving each individual lane are critical for the operation of the trailer pool, which can potentially be achieved by adopting the model we propose in this paper.

The contributions of this paper can be summarized as follows:

- We present necessary and sufficient conditions for the grand coalition of the CLC-LCG to be core stable.
- We propose a coalition selection model which aims to find a core stable coalition with maximum savings for the selected coalition along with a corresponding cost allocation in its core. It selects the grand coalition if its core is non-empty, otherwise it selects a smaller coalition that minimizes the total system-wide costs, equivalently maximizing the total savings.
- We develop an exact method that combines row and column generation, an upper bounding heuristic, and branch-and-bound to solve our coalition selection model.



(a) The representation of the lanes on a graph (b) The representation of the cycles on a graph

Fig. 1. A representative problem and solution to the CLC-LCP.

Table 1
Notation for the CLC-LCP

Sets	
L	Set of lanes
2^L	Set of all subsets (i.e. power set) of L
$C(L)$	Set of all feasible cycles over the lane set L, subset of 2^L
Parameters	
f_c	Cost of each feasible cycle $c \in C(L)$
b_{lc}	Binary parameter: 1 if a lane $l \in L$ is covered by a cycle $c \in C(L)$, 0 otherwise
Binary variables	
x_c	1 if a feasible cycle $c \in C(L)$ is selected, 0 otherwise

- We develop heuristics to generate rows and columns more efficiently. These heuristics can be applied to any problem that is formulated as a set covering/partitioning model.
- We test our theoretical results and computational models on randomly generated problem instances, and discuss broader implications of our research.

The rest of the paper is organized as follows. We discuss core stability as it relates to the problem setting in our focus in Section 2. We review the related literature in Section 3. We introduce the CSIP and our proposed solution method for it in Sections 4 and 5, respectively. We present the results of our computational experiments in Section 6, and we wrap up with conclusions in Section 7.

2. CLC-LCP and core stability

In the CLC-LCP, the objective is to find the tours minimizing the total cost such that every lane is covered by exactly one tour. The tours are restricted by cardinality and length constraints. In other words, there is an upper bound on both the number of lanes (cardinality) and the length of a tour. Mathematically, the CLC-LCP is defined by a directed graph $G = (N, A)$ over node set N, arc set A, lane set L ($L \subseteq A$), arc length d_a for each arc $a \in A$, cost per unit empty distance traveled γ_a^{empty} on each arc $a \in A$, cost per unit loaded distance traveled γ_l^{loaded} on each lane $l \in L$ with the objective of determining a set of simple cycles with minimum total cost such that:

1. each cycle covers at most K_{max} lanes,
2. the length of each cycle is at most T_{max} distance units.

We illustrate a CLC-LCP instance with six lanes in Fig. 1. As can be seen in Fig. 1a, each lane is defined by one origin node and one destination node on a graph. They also represent fully-loaded movements between these nodes. The underlying directed graph is assumed to be complete. The objective of the CLC-LCP is to find the set of cycles over lanes such that total cost is minimized and all lanes are covered. Let $K_{max} = 3$ lanes and $T_{max} = 10$ distance units. The grid consists of unit squares in terms of the distance unit used. Fig. 1b shows the cycles selected by the optimal solution for the given instance. The red dashed lines in Fig. 1b represent deadhead arcs or empty movements. They are used to complete the cycles, if necessary. There are three cycles in Fig. 1b: $\{1,2\}, \{3,4,5\}$, and $\{6\}$ with approximate costs of 8.48, 9.48, and 4, respectively assuming $\gamma_l^{loaded} = \gamma_l^{empty} = 1$ per unit distance for any lane l. We do not need to explicitly denote the arcs used for empty movement between lanes, as we can deduce them from the set of the lanes in each cycle. Similarly, we can also deduce the optimal sequencing of the lanes in each cycle and the resulting cycle cost. The selected cycles cover

all of the lanes, and satisfy K_{max} and T_{max} conditions.

The CLC-LCP can be solved to optimality by formulating a set covering or partitioning integer program over the set of all feasible cycles. To do this, we need to define sets, parameters and decision variables. Table 1 summarizes all the notation for sets, parameters and decision variable.

The resulting set partitioning integer program (SPIP) is as follows.

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

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

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

The objective of the SPIP is to minimize the total cost of the selected cycles. Constraints (2) assure that every lane is covered by only one cycle. Constraints (3) are integrality constraints.

We next discuss the subadditivity of the game in the following lemma. Let $F(L)$ be the optimal objective function value of the SPIP over the lane set L. The CLC-CLG is the cooperative game that can be defined in coalitional form by the pair (L, F) where L is the set of players and F is the characteristic function. Let w be the cost allocation vector such that w_l is the allocated cost to each lane ($l \in L$).

Lemma 1. *The CLC-CLG is subadditive.*

Proof. The subadditive property ensures that the total cost is less than or equal to the sum of the stand-alone costs of the players in the grand coalition. Let T and S be any two disjoint subsets of L and assume $T \cap S = \varnothing$. We have to show that the following inequality is valid for any two subsets.

$$F(T) + F(S) \geqslant F(T \cup S) \quad \forall T, S \subseteqq L \tag{4}$$

Note that any pair of disjoint subsets defines a partition for the union of these sets. Since the union of the optimal solutions of these sets is a feasible solution to the union of these sets, it also defines an upper bound on $F(T \cup S)$. In other words, let $x_c^{T^*}$ and $x_c^{S^*}$ be optimal solutions to the SPIP over T and S, respectively. Then we have $F(T) = \sum_{c \in C_{T^*}(L)} f_c x_c^{T^*}$ and $F(S) = \sum_{c \in C_{S^*}(L)} f_c x_c^{S^*}$. Since $x_c^{T^*}$ and $x_c^{S^*}$ define a feasible solution to the SPIP over $T \cup S$, we have $\sum_{c \in C_{T^*}(L)} f_c x_c^{T^*} + \sum_{c \in C_{S^*}(L)} f_c x_c^{S^*} \geqslant \sum_{c \in C_{T \cup S}(L)} f_c x_c$ which implies $F(T) + F(S) \geqslant F(T \cup S)$ for any two subsets. □ Since the game is subadditive, the grand coalition minimizes the total cost. On the other hand, if the grand coalition is not core stable, it will be susceptible to falling apart. If the grand coalition falls apart, the minimum cost given by the grand coalition cannot materialize. Furthermore, the CLC-LCG can be non-convex. Since the core of a convex game is always non-empty (Shapley, 1971), our results imply that the CLC-LCG is non-convex in general. We next derive the necessary and sufficient conditions for core stability in CLC-LCG, and show that the grand coalition is not always core stable in CLC-LCG.

The core of the CLC-LCG is defined by the budget balance and the stability conditions. Mathematically, w is in the core of the CLC-LCG if and only if it satisfies the following conditions:

$$\sum_{l \in L} w_l = F\left(L\right) \tag{5}$$

$$\sum_{l \in S} w_l \leqslant F\left(S\right) \quad \forall S \subseteqq L \tag{6}$$

Constraint (5) is *the budget balance (efficiency)* constraint (BBC) which ensures that the total allocated cost is equal to total minimized cost. The BBC does not allow any surplus or deficit. Constraints (6) are *the stability (rationality)* constraints (SCs) which assure that the total allocated cost does not exceed the minimized cost of a given subset of lanes.

We present the following lemmas to clarify the stability conditions for the core. Note that the following lemmas were previously shown to hold for the vehicle routing game (Göthe-Lundgren et al., 1996), the ridesharing game (Lu and Quadrifoglio, 2019), and the CLC-LCG (Öner and Kuyzu, 2020).

Lemma 2. *Inequalities (7) are equivalent to inequalities (6) for w to be in the core of the CLC-LCG.*

$$\sum_{l \in L} b_{lc} w_l \leqslant f_c \qquad \forall c \in C\left(L\right) \tag{7}$$

Proof. First, we will show that inequalities (6) imply inequalities (7). Suppose inequalities (6) are satisfied. Then, we have $\sum_{l \in S} w_l \leqslant F(S)$ for all $S \subseteqq L$. Let c be an arbitrary feasible cycle in $C(L)$. Recall that, as noted above, we denote each cycle in $C(L)$ as a set of lanes

without explicitly storing deadhead arcs, since the deadhead arcs (if any) can be deduced from the set of lanes in the cycle. Since every feasible cycle $c \in C(L)$ is composed of a set of lanes, i.e. $c \subseteqq L$, we have $C(L) \subset 2^L$ and $c \in 2^L$. Hence, if a cost allocation vector w (indexed by the lane set L) satisfies inequalities (6), then it automatically satisfies inequalities (7). Therefore, inequalities (6) imply inequalities (7).

Conversely, suppose inequalities (7) are satisfied. Any feasible solution to the SPIP over any lane set consists of a set of disjoint cycles, with its respective objective function value equal to the sum of the costs of these disjoint cycles. Consider an arbitrary lane set $S \subseteqq L$ and the optimal solution of the SPIP over S. By definition, $F(S)$ corresponds to the optimal objective function value of the SPIP over S. Let $C^*(S)$ denote the set of cycles selected by an optimal solution to the SPIP yielding $F(S)$. Then, $F(S) = \sum_{c \in C^*(S)} f_c$. Clearly, $C^*(S) \subseteqq 2^L$. If we take a linear combination of the subset of inequalities (7) corresponding to $C^*(S)$ with all multipliers equal to one, we obtain the inequality corresponding to S in inequalities (6). Mathematically, let ν_i be the multipliers corresponding to each element of $C^*(S)$ such that $\nu_i = 1$ for $i \in C^*(S)$, and n be equal to $|C^*(S)|$. Then we have:

$$\begin{aligned}
\nu_{c_1} \sum_{l \in L} b_{lc_1} w_l &\leqslant \quad \nu_{c_1} f_{c_1} : c_1 \in C^*(S) \\
\nu_{c_2} \sum_{l \in L} b_{lc_2} w_l &\leqslant \quad \nu_{c_2} f_{c_2} : c_2 \in C^*(S) \\
&\vdots \\
\nu_{c_n} \sum_{l \in L} b_{lc_n} w_l &\leqslant \quad \nu_{c_n} f_{c_n} : c_n \in C^*(S)
\end{aligned}$$

If we sum all inequalities over $C^*(S)$, we then have: $\sum_{c \in C^*(S)} \nu_c \sum_{l \in L} b_{lc} w_l \leqslant \sum_{c \in C^*(S)} \nu_c f_c$. Since all multipliers (ν_i) are equal to one, we can write $\sum_{c \in C^*(S)} \sum_{l \in L} b_{lc} w_l \leqslant \sum_{c \in C^*(S)} f_c$ which implies $\sum_{l \in S} w_l \leqslant F(S)$. Since we chose $S \subseteqq L$ arbitrarily, we have just shown that inequalities (7) imply inequalities (6) □

Lemma 3. *The core of the CLC-LCG is non-empty if and only if the LP relaxation of the SPIP yields integer solutions.*

Proof. Let the following problem (D-SPIP) be the dual of the LP relaxation of the SPIP. Let w be dual variable vector corresponding to the constraints (2) indexed by lane set L. The dual problem is defined as follows:

$$\text{D} - \text{SPIP} : \max \quad \sum_{l \in L} w_l \tag{8}$$

$$\text{s.t.} \quad \sum_{l \in L} b_{lc} w_l \quad \leqslant f_c \qquad \forall c \in C\left(L\right) \tag{9}$$

$$w_l \qquad \in \mathbb{R} \qquad \forall l \in L \tag{10}$$

Let $w_l^*, l \in L$ be an optimal solution of the D-SPIP and $Z(L)$ be the optimal objective value of the D-SPIP over the lanes set L. Clearly, $Z(L) = \sum_{l \in L} w_l^*$. Because of the weak duality theorem, we have $Z(L) \leqslant F(L)$. Since the D-SPIP is a maximization problem, $\sum_{l \in L} w_l \leqslant Z(L) \leqslant F(L)$ is valid for any given feasible allocation. Furthermore, $\sum_{l \in L} w_l^* = F(L)$ if and only if the SPIP has a zero integrality gap. Hence, we cannot find a cost allocation that satisfies both the BBC and the SC if the LP relaxation of the SPIP is not optimal for the SPIP. It follows that the core of the CLC-LCG is non-empty if and only if the LP relaxation of the SPIP yields integer solutions. □ Lemma 3 shows that strong conditions must be satisfied for the CLC-LCG to have a non-empty core. An implication of these conditions is that, for an arbitrarily selected lane set L, the core of CLC-LCG will likely be empty, and the grand coalition consisting of the lane set L will potentially be unstable. Motivated by these observations, we study the problem of selecting a core stable coalition with maximum system-wide savings for a given lane set L. We formulate a mixed integer linear program (MILP) for this purpose and develop methods for solving it. Our MILP is capable of identifying a core cost allocation if the core is non-empty for a given CLC-LCG instance. We make use of Lemma 4 in our formulation.

Lemma 4. *For a cost allocation vector in the core of the CLC-LCG, the cost of each cycle in the optimal SPIP solution is equal to the sum of the costs allocated to the set of lanes in that cycle. More formally, let w be a cost allocation vector in the core of the CLC-LCG, and $C^*(L)$ be the set of cycles in the optimal SPIP solution over the lane set L. Then,*

$$\sum_{l \in L} b_{lc} w_l \qquad = f_c \qquad \forall c \in C^*\left(L\right) \tag{11}$$

Proof. Let w^{core} be any core cost allocation vector indexed by lane set L, and L_c be the set of lanes covered by a feasible cycle c. Note that we can write the following equality for any feasible cycle c in $C(L)$: $\sum_{l \in L} b_{lc} w_l^{core} = \sum_{l \in L_c} w_l^{core}$. According to the BBC and SCs, we then have:

$$\sum_{l\in L} w_l^{core} = \sum_{c\in C^*(L)} f_c = F\left(L\right) \tag{12}$$

$$\sum_{l\in L_c} w_l^{core} \leqslant f_c \qquad \forall c \in C^*\left(L\right) \tag{13}$$

Suppose $\Delta > 0$ and there exists a cycle $\hat{c} \in C^*(L)$ such that:

$$\sum_{l\in L_{\hat{c}}} w_l^{core} = f_{\hat{c}} - \Delta < f_{\hat{c}} \tag{14}$$

$$\sum_{l\in L_c} w_l^{core} \leqslant f_c q \quad \forall c \in C^*\left(L\right)\setminus\left\{\hat{c}\right\} \tag{15}$$

If we add the inequalities defined in (14) and (15) together, since the cycles in $C^*(L)$ are disjoint, we get:

$$\sum_{l\in L} w_l^{core} \leqslant \sum_{c\in C^*(L)} f_c - \Delta = F\left(L\right) - \Delta < F\left(L\right) \tag{16}$$

which contradicts with (12). Hence, none of (13) can be satisfied as a strict inequality by a cost allocation in the core. Therefore, a cost allocation in the core must satisfy (13) with equality. □

3. Related Literature

Audy et al. (2010) reviewed the coordination mechanisms considered in the collaborative logistics literature, and noted that, in most cases, the logistics solution was planned first, and then the sharing was set based on the plan. They listed only a handful of works in which logistics planning and gain sharing was addressed simultaneously: most notably Agarwal and Ergun (2010) who design a mechanism with the purpose of motivating the members of a liner shipping alliance to act in the best interest of the alliance while maximizing their own profits. More recent papers (Cleophas et al., 2019; Basso et al., 2021; Wang et al., 2020a,b) indicate that the said pattern persists in the literature.

Although the assumption that the partners and the operational structure of the collaboration are determined independent of the cost allocation method remains to be common, some studies have relaxed that assumption. We are aware of only a handful of papers addressing coalition formation in collaborative logistics.

3.1. Forming only one coalition

The grand coalition may not be core stable for the characteristic function at hand. Some authors investigated mechanisms that can enable core stability of the grand coalition by incentivizing or penalizing players based on input parameters they set. Vanovermeire and Sörensen (2014) incorporated a mechanism to incentivize players to set their due dates such that the optimal cost of the grand coalition can be allocated to the players in a core stable way using Shapley value. Liu et al. (2018) proposed to simultaneously penalize players who leave the grand coalition and subsidize players who stay in the grand coalition to stabilize the grand coalition, and applied the concept to a class of machine scheduling games. Osicka et al. (2020) proposed incentives to ensure the carrier visits all relevant customers in a profitable tour problem setting, without enforcing any cooperative game-theoretic conditions. Akyol and De Koster (2018) formulated a satisfaction maximization game to jointly set the delivery time windows of neighboring cities, but did not enforce any cooperative game-theoretic conditions. In our study, we do not impose any penalties or incentives to stabilize the grand coalition, we focus on identifying a core stable coalition that would maximize system-wide savings, which can be smaller than the grand coalition.

Yang et al. (2020) worked on a cooperative vehicle routing problem with inter-depot transfers motivated by a real-life case involving three logistics providers. They solved a MIP to determine the coalition(s) without taking into account the core stability of the selected coalition and used Shapley value for cost allocation. They showed the conditions under which the selected coalition would have a non-empty core. Our approach differs from theirs in that we explicitly include constraints to ensure the core stability of the selected coalition.

Another approach we ran into is enumerating all the possible coalitions and selecting the best coalition from the enumerated set. Wang et al. (2017) follow such an approach for a collaborative vehicle routing problem with inter-depot transfers, where they solved a MIP for each enumerated coalition and allocated costs using an improved Shapley value model. Akyol and De Koster (2018) also enumerated all possible coalitions for determining time windows in urban freight and solved a nonlinear program maximizing total satisfaction for each enumerated coalition. We use column and row generation to enumerate only a small subset of possible coalitions without explicitly enumerating the entire set of possible coalitions.

3.2. Forming multiple coalitions

Forming multiple disjoint coalitions has been proposed by various researchers as a possible strategy when the grand coalition is not core stable. The resulting partition of the players is called a coalition structure (Aumann and Dreze, 1974). A coalition structure may allow forming multiple core stable coalitions. A harder condition to satisfy is *strong stability* (alternatively *strong rationality* or *strong equilibrium*), which ensures that players from different coalitions could not be better off by breaking away from their assigned coalitions and forming a separate coalition.

Elomri et al. (2012) studied the problem of identifying a set of disjoint core-stable coalitions that maximizes the savings of a set of firms purchasing products, according to respective EOQ policies, from the same supplier through joint replenishment. The authors defined core-stability with respect to a predefined standalone-cost-proportional allocation method. They formulated and solved a 0–1 fractional program over a set of enumerated core-stable coalitions.

Guajardo and Rönnqvist (2015) formulated mixed integer programs for determining a cost minimizing coalition structure along with a cost allocation satisfying core stability and/or strong stability, where the characteristic function amounted to solving a transportation problem. They assumed that a third party, such as a team of consultants suggested the set of candidate subcoalitions, which are not necessarily all core stable, and a cost minimizing way to implement the collaboration among the companies in each candidate subcoalition.

Jouida et al. (2017) proposed an iterative collaborative replenishment algorithm (RCA) for building the coalition structure consisting of core stable coalitions in a collaborative replenishment scenario with one supplier and its multiple customers where the characteristic function was given by minimized ordering and transportation cost. The RCA sequentially built each core-stable coalition in the structure using an enumeration scheme that eliminated some possible core-stable coalitions from consideration, assuming a volume-based gain-sharing method would be used.

Wang et al. (2018b) treated each ride match in a dynamic ridesharing platform as a coalition, and developed exact and greedy solution methods for identifying stable matchings with respect to a predetermined cost allocation formulas such as equal share of cost savings. The stability of the matchings were evaluated with respect to the concept of blocking coalitions.

Basso et al. (2020) studied collaborative job scheduling in the wine bottling process. The characteristic function was derived from a MIP minimizing total delay costs. They proposed a maximum entropy method for selecting the coalitional structure and a cost allocation simultaneously, given a set of enumerated core stable coalitions. They applied the proposed method with both exact and heuristic characteristic functions.

Jouida et al. (2020) proposed a three-phase collaborative warehousing algorithm that determined stand-alone costs, enumerated profitable coalitions, and heuristically selected a coalition structure in search of maximum total profit for a set of firms aiming to collaborate by sharing the capacities of their distribution networks. The algorithm was devised to achieve individual rationality with respect to a predetermined cost allocation method such as the egalitarian allocation, proportional allocation, and the Shapley value.

Basso et al. (2021) considered a coalition formation game with competing firms where the coalitional structure affected prices and production levels. They proposed multiple models to identify a coalition structure and an associated cost allocation, under stability and antitrust constraints, given a set of enumerated coalitions. The authors pointed out that collaboration would become very hard if the firms sought strong stability when forming the coalition structure.

We observe that determining the coalition structure for a given predetermined cost allocation formula (Elomri et al., 2012; Jouida et al., 2017; Wang et al., 2018a; Jouida et al., 2020) and simultaneously determining the coalition structure and the cost allocation (Guajardo and Rönnqvist, 2015; Basso et al., 2020; Basso et al., 2021) are both common and recently used approaches in the literature. We also observe that some studies considered strong stability (Guajardo and Rönnqvist, 2015; Basso et al., 2021), some considered only core stability (Jouida et al., 2017; Basso et al., 2020), while Jouida et al. (2020) considered less strict conditions. In LCGs, the characteristic function entails partitioning of the lane set into cycles, each of which is a (mini-) coalition itself. So, it is possible to interpret each LCG as a coalition structure game. If we interpreted the CLC-LCG as a coalition structure game, the SCs defined by inequality (11) would correspond to strong stability in the terminology of coalition structure games. Furthermore, the conditions satisfied by our proposed coalition selection approach could be considered a slight relaxation of the strong stability conditions of the coalition structure in the sense that they are not enforced for the lanes left outside the selected coalition.

A common feature we observe in the works we cite above is that the proposed methods predominantly took a set of enumerated coalitions as an input. The RCA of Jouida et al. (2017) integrated coalition enumeration and coalition structure construction. We note that these proposed methods were applied to test instances with few players, with the exception of Wang et al. (2018b) who assumed a predetermined cost allocation method. Since each lane is modeled as a player in the LCGs, a solution approach that could work efficiently on a high number of players/lanes is needed. In our solution approach, we generate a very small proportion of possible coalitions using large scale optimization methods, and perform computational experiments on test instances with a high number of lanes.

3.3. Lane covering problems and games

Previously published works on CTTP focused solely on either solving the LCP variants or developing cost allocation methods through cooperative games that have the LCP variants as characteristic functions. Ergun et al. (2007b) were the first to study CTTP and introduce the lane covering problems (LCPs) to the literature. They showed that the (base) LCP, without any constraints on the tours to be formed, can be solved in polynomial time either as a bi-partite matching problem or a flow circulation problem. They also showed that imposing cardinality and/or length constraints on the tours result in NP-Hard LCP variants, and developed heuristics for tackling

Table 2
Additional notation for the CLC-LCG.

Parameters	
d_l	Length of lane $l \in L$
g_l	Stand-alone (out-of-coalition) cost of lane $l \in L$
λ_l	Lower bound (multiplier) of the cost allocated to lane $l \in L$ if included in the coalition
θ_l	Minimum proportional savings for lane $l \in L$ if included in the coalition
Binary variables	
u_l	1 if lane $l \in L$ is not included in the coalition, 0 otherwise
Continuous variables	
w_l	Allocated cost to lane $l \in L$ if included in the collaboration

very large scale cardinality constrained LCP instances. The same authors (Ergun et al., 2007a) studied the time constrained LCP, which included length and time window constraints on the tours, and developed heuristics for finding good solutions to very large scale problem instances. Later, Kuyzu (2017, 2020) developed branch-and-price and hybrid genetic algorithm approaches, respectively, for solving the LCP with cardinality, length, and/or partner constraints.

Özener and Ergun (2008) studied the cost allocation problem arising in CTTP by formulating it as a cooperative game with the polynomially solvable the LCP as its characteristic function, namely the lane covering game (LCG). They analyzed the LCG using LP duality and showed that its core was always non-empty. Hezarkhani et al. (2014) analyzed the theoretical properties of dual based methods for the LCG to characterize the resulting core cost allocations. Recently, Öner and Kuyzu (2020) studied the cardinality and length constrained LCG (CLC-LCG), showed that each CLC-LCG must satisfy strong conditions to have a non-empty core, and designed nucleolus-based cost allocation methods for the CLC-LCP involving constraint generation procedures.

Coalition formation in CTTP settings where coalitions may have empty cores has not been studied before. We study the problem of selecting and forming a core stable coalition optimally from a given set of FTL lanes when the underlying characteristic function amounts to solving a CLC-LCP.

4. Coalition selection model for the CLC-LCG

We have shown in Section 2 that strong conditions exist on the presence of a core stable solution for the whole of an arbitrarily selected lane set L. These conditions imply that the core of the game is likely to be empty for the grand coalition. In other words, the grand coalition may not be core stable. As stated above, even though the grand coalition minimizes the total cost, the cost advantage of the grand coalition does not hold if it falls apart. An empty core increases the chances of the grand coalition falling apart. Including the wrong set of lanes in the coalition may cause a subset of lanes to break away from the coalition and form a separate coalition, therefore threatening the continuity of the coalition. We propose to use a MILP to select the right set of lanes for the coalition.

In this section, we present our MILP that selects the lanes to be included in the coalition such that the selected lanes form a core stable coalition and achieve maximum possible savings among stable coalitions. Our model integrates lane selection decisions, cost allocation decisions, and core stability conditions into one phase. The model allows us to incorporate additional desirable properties into the cost allocation decisions as well. In particular, the SCs allow allocating some lanes zero or even negative cost while leaving some lanes without any positive savings. We place a lower bound and an upper bound on each allocated cost to prevent such cases. The lower bound prevents free riders, while the upper bound guarantees a certain percentage of savings for each lane in the coalition. The objective of our coalition selection model is to find only one core stable coalition over L such that all conditions defined above are satisfied.

In order to formulate our MILP, we need to define more parameters and decision variables. We provide the updated or additional notation that is needed for the new formulation in Table 2.

Then, our coalition selection integer program (CSIP) is as follows.

$$\text{CSIP}: \min \quad \sum_{c \in C(L)} f_c x_c + \sum_{l \in L} g_l u_l \tag{17}$$

$$\text{s.t.} \quad \sum_{c \in C(L)} b_{lc} x_c = \left(1 - u_l\right) \quad \forall l \in L \tag{18}$$

$$\sum_{l \in L} w_l - \sum_{c \in C(L)} f_c x_c \geqslant 0 \tag{19}$$

$$\sum_{l \in L} b_{lc} w_l \leqslant f_c \quad \forall c \in C\left(L\right) \tag{20}$$

$$w_l \leqslant (1-\theta_l) g_l (1-u_l) \quad \forall l \in L \tag{21}$$

$$w_l \geqslant \lambda_l d_l (1-u_l) \quad \forall l \in L \tag{22}$$

$$u_l \in \{0,1\} \quad \forall l \in L \tag{23}$$

$$x_c \in \{0,1\} \quad \forall c \in C(L) \tag{24}$$

$$w_l \geqslant 0 \quad \forall l \in L \tag{25}$$

The objective (17) of the CSIP minimizes the total system-wide cost, which is equal to the sum of the cost of the selected coalition and the standalone costs of the lanes left outside the coalition. Constraints (18) ensure that every lane included in the coalition is covered by exactly one cycle. Constraints (19) and (20) are the BBC and the SCs, respectively, making use of Lemma 2. Constraints (21) ensure that every lane in the coalition obtains specific percentage savings. These constraints also satisfy the individual rationality condition. Without these constraints, a lane could be allocated its standalone cost and have zero cost savings. Constraints (22) provide a lower bound for allocated costs. Without constraints (22), the cost allocated to a lane may be zero. We defined constraints (21) and (22) to avoid these undesirable situations, which are possible in the standard core stability conditions. These two constraints also allocate zero cost to the lanes excluded from the collaboration, and thus ensure that excluded lanes do not cause violation of constraints (19) and (20). Constraints (24)–(25) are integrality and non-negativity constraints.

Note that constraints (20) will force constraint (19) to be satisfied with equality. Let $\widehat{L}$ be a set of lanes covered in any feasible solution and $C(\widehat{L})$ be a set of cycles which cover all lanes in $\widehat{L}$. Then, we can write constraints (19) as follows: $\sum_{l\in L} w_l - \sum_{c\in C(\widehat{L})} f_c \geqslant 0$. According to constraints (20), we have $\sum_{l\in L} b_{lc} w_l \leqslant f_c \quad \forall c \in C(\widehat{L})$. If we sum constraints (20) over $C(\widehat{L})$, we obtain $\sum_{c\in C(\widehat{L})} \sum_{l\in L} b_{lc} w_l \leqslant \sum_{c\in C(\widehat{L})} f_c$. Since $w_l = 0 \ \forall l \in L \backslash \widehat{L}$, we can write $\sum_{l\in L} w_l \leqslant \sum_{c\in C(\widehat{L})} f_c$. $\sum_{l\in L} w_l - \sum_{c\in C(\widehat{L})} f_c \geqslant 0$ and $\sum_{l\in L} w_l - \sum_{c\in C(\widehat{L})} f_c \leqslant 0$ imply $\sum_{l\in L} w_l - \sum_{c\in C(\widehat{L})} f_c = 0$. Writing constraint (19) as an inequality causes the corresponding dual variable to take on non-negative values only, which helps us reduce the computational effort in our solution approach.

The CSIP implicitly assumes that the lanes not included in the coalition remain as singletons. As mentioned in the introduction, our model can be used in collaborative logistics networks or digital platforms in which accepted members take advantage of a large pool of lanes and centralized optimization technology. The network/platform has a severely asymmetric advantage over the participants in terms of access to information and technological capabilities. Players left out of the coalition may not be aware of other candidates or may be aware of a limited number of candidates. In addition, lanes submitted for consideration are likely to be "bad" lanes which cannot be combined with the respective shippers' remaining lanes. Therefore, it is reasonable to assume that the lanes excluded from the coalition will remain as singletons.

We present the details of our approach for solving the CSIP in the next section.

5. A solution procedure for the CSIP

As the number of lanes in the CSIP increases, the set of feasible cycles grows exponentially. As a result, the size of the CSIP also grows exponentially since it includes a row and a column for each feasible cycle. Hence, identifying an optimal solution to the CSIP becomes much harder as the number of lanes grows. We refer the reader to Section 6.3 for a set of experiments showing the computational difficulty of solving the CSIP.

Identifying high quality integer feasible solutions to the CSIP efficiently also turns out to be a challenging task. Therefore, we focus on finding a core stable coalition by solving the CSIP over a restricted set of cycles. We first solve the LP relaxation of the CSIP, which we denote the CSLP, using a row and column generation method we developed. We then solve the CSIP over the set of cycles corresponding to the rows and columns generated in the process. We will refer to the resulting integer program as the R-CSIP in the rest of the paper.

We obtain the CSLP by replacing constraints (23) and (24) of the CSIP with (26) and (27).

$$1 \geqslant u_l \geqslant 0 \quad \forall l \in L \tag{26}$$

$$x_c \geqslant 0 \quad \forall c \in C(L) \tag{27}$$

In our solution procedure, we work with the restricted CSLP (R-CSLP), which initially has a small subset of the rows and the columns of the CSLP. There are a limited number of u_l variables in the formulation. Thus, we include all of the columns corresponding to the u_l variables in the R-CSLP from the beginning. We gradually add the columns corresponding to the cycle selection variables x_c, $c \in C(L)$. Since there is an SC (20) for each feasible cycle, we also add the corresponding rows gradually. At the beginning of the algorithm, we only generate the feasible cycles which include up to two lanes. Hence, the R-CSLP includes only the cycle selection

variables and SCs corresponding to such cycles in the beginning. We generate cycle selection variables and SCs as needed. We generate cycle selection variables by finding columns with negative reduced cost. We generate SCs by searching for violated ones.

Since the cycle selection variables do not appear in the SCs, we can generate the rows and the columns in any order. Our algorithm first tries to identify cycles which violate the stability conditions (rows). If it fails to find such cycles, it tries to find cycles with negative reduced cost (columns). If it can find a new row or a new column, it tries to find a new row or a new column again. In row generation, only one cycle which has the highest stability violation is added to the R-CSLP at each step. As a speed-up, the column corresponding to the cycle with the highest stability violation is also added to the R-CSLP. Similar to row generation, column generation seeks to identify the cycle which has the most negative reduced cost. Likewise, the SC corresponding to any generated column is added to the R-CSLP. The algorithm stops when no new row or no new column can be generated.

5.1. Column generation procedure

We first explain our column generation procedure, before explaining our row generation procedure, because our column generation procedure provides the foundation for our row generation procedure.

Let $C(L)^{gen}$ be the set of cycles included in the R-CSLP. The column generation procedure aims to identify feasible cycles with a negative reduced cost. If it can find such a cycle, it is added to $C(L)^{gen}$, and the corresponding column and SC row are added to the R-CSLP. We now define the reduced cost of a cycle mathematically. Let β_l and δ be the dual variables corresponding to constraints (18), and (19) of the CSIP, respectively. Then, the reduced cost of any cycle c is given by:

$$\overline{f_c} = f_c - \sum_{l \in L} b_{lc}\beta_l + \delta f_c \tag{28}$$

Let b_{ac}^{dh} indicate if a cycle c contains an arc $a \in A$ as a deadhead arc or not. We can replace (28) with the following Eq. (29), which we will use when searching for negative reduced cost columns:

$$\overline{f_c} = \sum_{l \in L} \left((1+\delta)\gamma_l^{loaded} d_l - \beta_l \right) b_{lc} + \sum_{a \in A} \left(1+\delta \right) \gamma_a^{empty} d_a b_{ac}^{dh} \tag{29}$$

Next, we formulate an integer program to search for negative reduced cost columns. We first define decision variables corresponding to b_{lc}, and b_{ac}^{dh}. Let binary decision variables r_l be equal to 1 if the lane $l \in L$ is selected as loaded truckload movement, 0 otherwise. Let binary decision variables t_a be equal to 1 if arc $a \in A$ is selected for empty truckload movement, 0 otherwise. The resulting column generation pricing subproblem (CGPSP) is given as follows:

$$\text{CGPSP}: \min \quad \sum_{l \in L} \left(\left(1+\delta \right) \gamma_l^{loaded} d_l - \alpha_l \right) r_l + \sum_{a \in A} \left(1+\delta \right) \gamma_a^{empty} d_a t_a \tag{30}$$

$$\text{s.t.} \quad \sum_{(m,n) \in L} r_{(m,n)} + \sum_{(m,n) \in A} t_{(m,n)} - \sum_{(n,m) \in L} r_{(n,m)} - \sum_{(n,m) \in A} t_{(n,m)} = 0 \quad \forall n \in N \tag{31}$$

$$\sum_{(m,n) \in L} r_{(m,n)} + \sum_{(m,n) \in A} t_{(m,n)} \leqslant 1 \quad \forall n \in N \tag{32}$$

$$\sum_{(n,m) \in A} t_{(n,m)} + \sum_{(k,n) \in A} t_{(k,n)} \leqslant 1 \quad \forall n \in N \tag{33}$$

$$\sum_{l \in L} r_l \leqslant K_{max} \tag{34}$$

$$\sum_{l \in L} d_l r_l + \sum_{a \in A} d_a t_a \leqslant T_{max} \tag{35}$$

$$r_l \in \{0,1\} \quad \forall l \in L \tag{36}$$

$$t_a \in \{0,1\} \quad \forall a \in A \tag{37}$$

The CGPSP aims to find the cycle(s) with the least reduced cost. Constraints (31) are flow-balance constraints. They ensure that the number of entering arcs is equal to the number of leaving arcs. Constraints (32) guarantee that every generated cycle is a simple cycle. Constraints (33) assure that two consecutive deadhead arcs are not selected in the solution. Constraints (34)–(35) are the cardinality, and length constraints, respectively. Constraints (36)–(37) are the integrality constraints.

A negative optimal objective value for the CGPSP means that the current solution to the R-CSLP can be improved by adding a column. The CGPSP can generate more than one simple cycle. In this regard, it can be viewed as a relaxation of the true CGPSP, which can be shown to be NP-Hard by a reduction from the asymmetric traveling salesman problem. If the optimal solution of the CGPSP includes more than one simple cycle, the one with most negative reduced cost is added to the R-CSLP. If the objective of the CGPSP is positive, it means that we cannot find any cycle with a negative reduced cost. Consequently, we stop the column generation procedure.

5.1.1. Column generation heuristics

Since the CGPSP is an integer program, generating columns solely by solving the CGSP may add up to excessive running time. Hence, we develop four heuristics to generate the columns with negative reduced cost instead of using the CGPSP at each step. If all of the heuristics fail to identify a new column, we solve the CGPSP as an integer program using a commercial solver.

The first column generation heuristic, *Add*, searches for new columns by adding a lane into the cycles in the basis of the R-CSLP at the last position. Note that we represent each cycle as a sequence of lanes. Hence, the last position represents the end of the corresponding sequence of lanes. After adding each lane into each possible cycle, the one with most negative reduced column is added to the model if exists.

The second column generation heuristic, *Join*, tries to find new columns by joining two disjoint columns in the basis of the R-CSLP. In order to join these columns, we remove the deadhead arcs at the last position, and then add two new deadhead arcs to join the columns. In this case, we join two columns with non-positive reduced cost.

The third column generation heuristic, *Cross-Join*, seeks to identify new columns by joining a column in the basis and a column out of the basis. Unlike *Join*, *Cross-Join* tries to join a column with non-positive reduced cost and a column with non-negative reduced cost.

The fourth column generation heuristic, *Mega-Add*, aims to obtain new columns with negative reduced cost by adding each lane into the columns in the R-CSLP. Unlike *Add*, *Mega-Add* tries all possible positions in the columns to find new columns. Because of that, we use this heuristic as the last alternative since it may take much more time than the others.

The heuristics are tried in the order given above. 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 one cannot find the column with negative reduced cost, we try the next one. For every heuristic, the column with the most negative reduced cost is added to the R-CSLP. When all heuristics fail to identify a new column with a negative reduced cost, we solve the CGPSP to optimality using a commercial solver.

After generating a cycle in the cycle generation procedure, both the variable and the stability condition corresponding to the cycle are added to the R-CSLP.

5.2. Row generation procedure

The R-CSLP has a limited number of the SCs, i.e. (20), in the beginning. We add these constraints to the R-CSLP, as needed. Given optimal allocated costs $\widetilde{w}_l$ for the R-CSLP, we first look for cycles which violate the SCs. If we can find at least one such cycle, we add the SC and the column of the cycle with the highest stability violation to $C(L)^{gen}$ and the R-CSLP. The stability violation of cycle c is calculated as follows:

$$\sum_{l \in L_c} \widetilde{w}_l - f_c \tag{38}$$

We adapt the column generation heuristics to search for rows efficiently. We aim to find a cycle corresponding to a violated stability constraint. Similar to the column generation procedure, we solve the row generation subproblem (RGSP) to optimality using a commercial solver if the row generation heuristics cannot identify a violated stability constraint. The RGSP is an integer program with the objective of maximizing stability violation, which we write in minimization form, subject to the same constraints as the CGPSP:

$$\text{RGSP}: \min \quad \sum_{l \in L} \left(\gamma_l^{loaded} d_l - \widetilde{w}_l \right) r_l + \sum_{a \in A} \gamma_a^{empty} d_a t_a \tag{39}$$

$$\text{s.t.} \quad (31) - (37)$$

If the optimal objective of the RGSP is non-negative, the current allocation vector satisfies all of the SCs. Like the CGPSP, if the optimal solution is composed of multiple cycles, we add the cycle with the highest stability violation to $C(L)^{gen}$ and the R-CSLP.

Similar to the column generation procedure, we add both the stability constraint and the variable to the R-CSLP corresponding to the cycle when we generate a row from the heuristics or the RGSP. Once a new row is added to the RMLP, we try to generate new rows until the RGSP solution has a positive objective value.

If the objective of the RGSP is non-negative, then we switch to the column generation procedure. If both the objective of the RGSP and the CGPSP are non-negative, the row and column generation procedures satisfy the stopping condition. In the rest of the paper, we refer to our row and column generation procedure as the RCG. The pseudocode of the RCG is given in Algorithm 1.

Algorithm 1. Row and Column Generation

$C(L)^{gen} \leftarrow$set of cycles with at most 2 lanes
repeat
  $c_{new} \leftarrow \emptyset$
  Solve the R-CSLP over current $C(L)$
  $C_b(L) \leftarrow$ Set of cycles in the basis of the R-CSLP
  $C_{nb}(L) \leftarrow$ Set of cycles out of the basis of the R-CSLP
  $c_{new} \leftarrow RowGenerationProcedure(C_b(L), C_{nb}(L), L)$
  if $c_{new} = \emptyset$ **then**
    $c_{new} \leftarrow ColumnGenerationProcedure(C_b(L), C_{nb}(L), L)$
  if $c_{new} \neq \emptyset$ **then**
    $C(L)^{gen} \leftarrow C(L)^{gen} \cup \{c_{new}\}$
    Add c_{new}'s column and SC to the R-CSLP
until c_{new} = null

5.3. Ratio based heuristic

As stated earlier, we also develop a heuristic algorithm to search for a good solution to the CSIP. The heuristic, which we call the *ratio based heuristic (RBH)*, builds a core stable coalition by selecting cycles greedily and randomly from a set of candidate cycles. The RBH starts with the cycles generated by the RCG. The RBH also helps to increase the efficiency of the branch-and-RCG procedure, which we describe in the next section, by providing upper bounds.

The RBH is based on the *ratio* of a cycle, which is calculated by dividing the total fully-loaded distance of the cycle by the cost of the cycle. We expect the final solution to contain high ratio cycles. The RBH is a greedy randomized selection heuristic that utilizes the ratio of each cycle as its selection probability. The ratio r_c of a cycle $c \in C(L)$ is calculated as:

$$r_c = \frac{\sum_{l \in L_c} d_l}{f_c} \tag{40}$$

The RBH starts with a set of candidate cycles $\widetilde{C}(L)$ and an empty set of selected cycles $\widehat{C}(L)$. Then, the RBH keeps evaluating the cycles in $\widetilde{C}(L)$ until either it is empty or a predetermined time limit has been reached. The evaluation of cycles is done as follows. The algorithm takes the cycle with the highest ratio in $\widetilde{C}(L)$, and generates a random number between zero and one. If the random number is greater than the ratio of the cycle (r_c), then the algorithm moves on to the cycle with the next highest ratio among the remaining candidate cycles. If the random number is less than or equal to the ratio of the cycle, then it temporarily adds that cycle to $\widehat{C}(L)$, and check whether the resulting $\widehat{C}(L)$ forms a feasible solution to the CSIP. If $\widehat{C}(L)$ is found to be infeasible for the CSIP, then the last added cycle is removed from $\widehat{C}(L)$ and the next cycle is tried.

If the feasibility conditions are satisfied, then the CSIP objective value corresponding to $\widehat{C}(L)$ is compared with the incumbent solution. Note that if the selected cycles yield a feasible solution, it means that they satisfy constraints (18)–(22) in the CSIP. If the current solution is better than the incumbent solution, then the incumbent solution is updated. After all cycles are evaluated, the ratio of the cycles in $\widehat{C}(L)$ is decreased by 0.2 to diversify the solutions found by preventing the same cycles to be selected repeatedly. Since a cycle whose ratio is less than 0.5 cannot be in the optimal solution, the ratio of each such cycle is recalculated. After decreasing the ratios, all procedures described above are followed until the stopping conditions are reached. The pseudocode of the RBH is given in Algorithm 2. Recall that each cycle is a set of lanes.

Table 3
Instances used in the experiments.

Instance	$\|N\|$	$\|L\|$	CPR
1–3	100	100	0.5
4–6	100	100	0.8
7–9	100	150	0.5
10–12	100	150	0.8
13–15	100	200	0.5
16–18	100	200	0.8
19–21	150	150	0.5
22–24	150	150	0.8
25–27	150	200	0.5
28–30	150	200	0.8

Algorithm 2. Ratio Based Heuristic

for all $c \in \widetilde{C}(L)$ **do**
  $r_c \leftarrow \frac{\sum_{l \in L_c} d_l}{f_c}$
while Time limit has not been reached **do**
  $\widehat{C}(L) \leftarrow \varnothing$
  Sort $\widetilde{C}(L)$ in descending order of r_c
  for all $c \in \widetilde{C}(L)$ **do**
    if $\widehat{L} \cap c = \varnothing$ **then**
      generate *rndNumber* $\in [0,1]$
      if *rndNumber*$\leqslant r_c$ **then**
        if $\widehat{C}(L) \cup \{c\}$ does not violate any CSIP constraints **then**
          $\widehat{C}(L) \leftarrow \widehat{C}(L) \cup \{c\}$
          $\widehat{L} \leftarrow \widehat{L} \cup c$
          totalCost $\leftarrow \sum_{c \in \widehat{C}(L)} f_c + \sum_{l \in L \setminus \widehat{L}} g_l$
          for $\hat{c} \in \widehat{C}(L)$ **do**
            $r_{\hat{c}} \leftarrow r_{\hat{c}} - 0.2$
            if $r_{\hat{c}} \leqslant 0.5$ **then**
              $r_{\hat{c}} \leftarrow \frac{\sum_{l \in L_c} d_l}{f_c}$
          if *bestObjective* $<$ *totalCost* **then**
            bestObjective $\leftarrow$ *totalCost*

5.4. The branch-and-RCG algorithm

The RCG is not sufficient to solve the CSIP to optimality. It can only solve its LP relaxation —the CSLP. We embed the RCG into the branch-and-bound procedure commonly used for solving integer programs. We refer to the resulting solution method as the branch-and-RCG algorithm (BnRCG). The BnRCG is capable of solving the R-CSIP to provable optimality.

Table 4
Results obtained by full cycle enumeration and solving the SPIP with CPLEX defaults.

Instances	LCP Gap	Process Time	$\lvert\widehat{L}\rvert$	$\lvert\widehat{C}(L)\rvert$	$\lvert C(L)\rvert$
1	1.0	4.7	96	36	86,399
2	0.4	12.2	96	34	301,241
3	2.4	9.3	98	33	166,085
4	0.3	5.4	97	37	85,104
5	0.1	16.2	99	31	332,606
6	0.1	12.1	98	32	272,819
7	0.6	43.1	144	57	1,072,968
8	1.1	30.3	142	54	542,419
9	0.3	12.2	147	56	256,662
10	0.7	24.2	146	55	490,895
11	0.2	31.7	143	50	622,558
12	0.3	47.8	149	53	742,763
13	0.8	42.7	196	73	608,663
14	1.0	32.9	193	71	467,756
15	0.4	62.3	196	67	865,389
16	0.6	309.8	191	65	3,760,943
17	–	–	–	–	–
18	0.0	200.5	197	67	3,689,912
19	0.6	12.3	146	53	149,223
20	1.7	19.6	146	51	384,438
21	0.9	14.6	145	51	172,477
22	0.2	27.6	148	52	593,938
23	0.0	1,194.4	149	45	21,927,644
24	0.2	119.0	148	46	1,910,757
25	0.2	72.1	199	65	1,074,052
26	0.2	508.9	195	63	7,132,911
27	2.3	408.8	194	61	5,406,714
28	1.7	16.4	194	73	223,634
29	0.2	42.4	197	67	672,057
30	0.2	114.7	199	68	1,362,533

Table 5
Core stability of the grand coalitions.

Instances	SPIP	LP-SPIP	Difference	MMSV	Modified MMSV
1	113,216.6	113,206.4	10.22	0.26	6.15
2*	106,937.8	106,937.8	0.00*	0.00*	14.48
3	107,163.5	107,160.0	3.43	0.10	7.36
4	117,249.9	117,248.7	1.27	0.03	13.25
5	98,631.3	98,626.6	4.67	0.15	5.15
6*	97,201.1	97,201.1	0.00*	0.00*	8.89
7*	177,618.7	177,618.7	0.00*	0.00*	47.39
8	180,574.1	180,562.3	11.85	0.19	54.59
9*	179,099.7	179,099.7	0.00*	0.00*	11.91
10*	184,304.5	184,304.5	0.00*	0.00*	23.95
11	168,826.9	168,820.3	6.61	0.12	35.02
12	166,998.6	166,996.2	2.45	0.05	6.59
13	228,890.3	228,874.7	15.53	0.20	13.56
14	232,172.5	232,165.4	7.16	0.09	10.04
15	212,532.3	212,528.7	3.58	0.05	4.71
16	219,269.2	219,268.8	0.42	0.01	22.80
17	–	–	–	–	–
18*	199,002.3	199,002.3	0.00*	0.00*	10.81
19	172,937.8	172,897.4	40.34	0.71	6.32
20*	162,226.2	162,226.2	0.00*	0.00*	20.40
21	167,708.1	167,698.5	9.57	0.17	8.74
22	169,634.0	169,621.7	12.37	0.22	26.70
23	113,639.7	113,637.6	2.12	0.05	20.99
24	145,207.5	145,193.6	13.91	0.30	8.98
25	207,039.7	207,026.1	13.62	0.21	4.21
26	186,005.2	185,994.9	10.30	0.16	10.82
27	200,760.1	200,737.4	22.70	0.33	9.67
28	245,212.6	245,208.9	3.73	0.05	11.74
29*	215,196.4	215,196.4	0.00*	0.00*	4.58
30	208,262.3	208,258.1	4.13	0.06	7.13

*: Non-Empty Core.

The BnRCG identifies a lower bound to the optimal CSIP objective value by solving an LP relaxation using the RCG at each node of the branch-and-bound tree. If the LP relaxation turns out to be infeasible, then the corresponding node is pruned. If the RCG stops with a feasible lower bound, we first check the integrality of the binary decision variables. If all of the binary decision variables are integer-valued, then the current solution yields an upper bound to the optimal objective value. We will have identified an optimal solution if this happens at the root node. At other nodes, we either set the current solution as the incumbent or prune it, depending on whether the current solution improves the best known upper bound or not.

If the solution found by the RCG has at least one fractional-valued binary decision variable at any node, then the BnRCG proceeds to branching. The variables u_l have higher priority than x_c in branching because a high number of cycles might be eliminated when a variable u_l is set to one. This strategy could reduce the size of the solution space. When all u_l variables are integer-valued, we branch on one of the x_c variables. We branch on the variable with the value of which is closest to one. We use a depth-first strategy to search the tree.

We use two stopping conditions in the BnRCG. The first one is to run the algorithm until there is no active node. In this case, the algorithm yields an optimal solution to the CSIP. Because of the complexity of the problem, the computation effort required by the BnRCG to find an exact solution grows exponentially as the problem size grows. Hence, we apply a time limit on the running time of the algorithm as a second stopping condition.

Before running the BnRCG, we run the RBH to find a good upper bound to increase the efficiency of the algorithm since a good initial upper bound might reduce the size of the search tree. In addition, we run the RBH periodically after a specific number of iterations. We add all of the cycles generated by the RCG to a common list throughout the search tree. Hence, the set of generated cycles gradually grows and the RBH can try different cycles to find better bounds as the number of nodes in the search tree increases.

6. Computational experiments

We tested the mathematical model and the proposed solution procedure on a set of Euclidean instances. The test instances were generated using the same procedure as Ergun et al. (2007a,b); Kuyzu (2017); Öner and Kuyzu (2020). The test instances contain lanes between semi-clustered points and distributed over a $1,800 \times 1,800$ miles square region. The instances mimic the structure of supply chains by dividing the nodes into three classes representing suppliers, plants/warehouses, and customers. The instances have 100, 150, and 200 lanes ($|L|$) and 100 or 150 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 desired number of lanes is randomly chosen from among all possible arcs, providing that each

Table 6
LCP gaps and process times for solving the CSIP by brute force and using the BnRCG.

	BFM			BnRCG	
Instance	CPLEX Gap	LCP Gap	Process Time	LCP Gap	Process Time
1	0.0	1.0	89.6	2.8	*TL*
2	–	–	*TL*	1.1	*TL*
3	1.9	4.3	*TL*	2.5	*TL*
4	24.8	33.4	*TL*	1.4	*TL*
5	–	–	*TL*	2.8	*TL*
6	–	–	*TL*	9.7	*TL*
7	–	–	*TL*	3.7	*TL*
8	–	–	*TL*	2.8	*TL*
9	31.2	45.8	*TL*	6.5	*TL*
10	–	–	*TL*	7.5	*TL*
11	–	–	*TL*	5.1	*TL*
12	–	–	*TL*	1.7	*TL*
13	–	–	*TL*	1.8	*TL*
14	–	–	*TL*	2.3	*TL*
15	–	–	*TL*	3.0	*TL*
16	–	–	*TL*	18.4	*TL*
17	–	–	*TL*	11.0	*TL*
18	–	–	*TL*	8.2	*TL*
19	33.1	50.5	*TL*	29.9	*TL*
20	–	–	*TL*	4.0	*TL*
21	34.8	54.8	*TL*	1.5	*TL*
22	–	–	*TL*	5.1	*TL*
23	–	–	*TL*	5.9	*TL*
24	–	–	*TL*	1.4	*TL*
25	–	–	*TL*	1.1	*TL*
26	–	–	*TL*	34.9	*TL*
27	–	–	*TL*	38.7	*TL*
28	–	–	*TL*	3.6	*TL*
29	–	–	*TL*	8.1	*TL*
30	–	–	*TL*	39.6	*TL*

point has at least one lane incident to it and there are no lanes with an origin and a destination in the same cluster. The lanes must be either from suppliers to plants/warehouses, between plants/warehouses, or from plants/warehouses to customers. Moreover, each supplier must have least one outgoing lane, each plant/warehouse must have at least one incoming and one outgoing lane, and each customer must have at least one incoming lane. We used three different random seeds for every combination to generate the test instances. In total, we generated 30 different instances to evaluate our solution approaches. Table 3 provides detailed information about the instances used in the experiments.

In the experiments, the total length of each cycle (T_{max}) was allowed to be at most 3850 miles, and each cycle could include at most four lanes (K_{max}). We set the minimum proportional savings (θ_l) to 0.05, the lower bound of the cost (λ_l) to 0.2, the fully-loaded cost coefficient (γ_l^{loaded}) to 1.0 and the standalone cost (g_l) to $1.8d_l$ for all lanes and the repositioning cost coefficient (γ_a^{empty}) to 0.8 for all arcs. We imposed a time limit of two minutes on the RBH and a time limit of two hours on the BnRCG.

We implemented all of the mathematical models and our proposed algorithms using the Java programming language. We ran 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. IBM ILOG CPLEX 12.9 solver and Concert technology were utilized to solve the linear and mixed integer models.

6.1. Core stability of the grand coalition

We ran a set of experiments to check the core stability of the grand coalitions in the instances we randomly generated. Lemma 3 implies that the grand coalition L will be core stable if and only if the integrality gap of the SPIP corresponding to $F(L)$ is zero. So, for each instance, we solved the SPIP and the LP relaxation of the SPIP (LP-SPIP) over the respective grand coalition *without any time limit*, and compared the optimal objective function values. If both the SPIP and the LP-SPIP yielded the same objective function value, the grand coalition was core stable, i.e. the CLC-LCG instance had a non-empty core. We adopted a brute force approach for this purpose. We fully enumerated $C(L)$ and solved both the SPIP and the LP-SPIP over $C(L)$. Recall the CLC-LCP instance with six lanes in Fig. 1. When enumerating cycles, we first take each lane as a cycle. The rest of the cycles are generated recursively using these cycles. We do not allow duplicate cycles. A duplicate prevention strategy we use is making sure that the first lane of each each cycle is the lowest numbered lane in that cycle. For example, we generate the ordered lane tuple $(1,2)$ but not $(2,1)$. Furthermore, a subset of lanes can be visited in a different order, i.e. $(1,2,3)$ or $(1,3,2)$. In this case, we keep the one with the lowest cost in $C(L)$ since only such a cycle can be in the optimal solution, and we represent it as the (unordered) set $\{1,2,3\}$. We also do not allow the nodes to be visited more than

Table 7
Lane and cycle counts when solving the CSIP by brute force and using the BnRCG.

Instance	BFM			BnRCG			
	$\lvert\widehat{L}\rvert$	$\lvert\widehat{C}(L)\rvert$	$\lvert C(L)\rvert$	$\lvert\widehat{L}\rvert$	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)	$\lvert\widehat{C}(L)\rvert$	$\lvert C(L)^{gen}\rvert$
1	95	36	67,339	84	84.0	31	3,804
2	–	–	208,794	84	84.0	27	4,407
3	90	31	116,999	95	95.0	32	4,014
4	34	13	65,495	83	83.0	29	3,342
5	–	–	245,484	79	79.0	26	4,622
6	–	–	227,590	69	69.0	23	5,129
7	–	–	785,535	91	60.7	33	8,499
8	–	–	396,972	103	68.7	36	8,002
9	19	8	166,291	95	63.3	37	6,488
10	–	–	384,761	93	62.0	34	6,985
11	–	–	495,528	94	62.7	34	8,348
12	–	–	587,169	128	85.3	46	6,644
13	–	–	394,904	158	79.0	60	12,014
14	–	–	346,714	160	80.0	62	11,684
15	–	–	613,232	153	76.5	54	13,549
16	–	–	2,800,185	86	43.0	29	14,867
17	–	–	16,786,177	104	52.0	31	20,766
18	–	–	2,664,581	128	64.0	45	16,565
19	18	7	109,908	50	33.3	20	6,728
20	–	–	257,462	113	75.3	37	7,876
21	8	3	129,343	136	90.7	48	6,755
22	–	–	364,934	103	68.7	37	7,466
23	–	–	12,744,589	97	64.7	27	12,162
24	–	–	1,347,310	126	84.0	41	9,809
25	–	–	871,924	183	91.5	61	15,300
26	–	–	4,768,825	45	22.5	18	17,421
27	–	–	3,432,777	37	18.5	13	16,233
28	–	–	162,649	165	82.5	65	7,344
29	–	–	526,432	138	69.0	50	12,587
30	–	–	1,066,320	51	25.5	17	13,962

once, since such cycles can be split without any harm. According to this process, we can generate 16 cycles for this instance under K_{max} and T_{max} contraints: $\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{1,2\},\{1,6\},\{2,3\},\{2,5\},\{3,4\},\{3,5\},\{4,5\},\{4,6\},\{5,6\}$, and $\{3,4,5\}$. Each cycle is an element of $C(L)$ and corresponds to an x_c variable.

We double-checked our results by solving an LP that seeks a budget-balanced cost allocation minimizing the maximum violation of the SCs. The said LP is similar to the pre-nucleolus concept (Guajardo and Jörnsten, 2015), except that its constraints are defined over $C(L)$ instead of 2^L and it does not include the individual rationality constraints. We use the same notation in Tables 1 and 2. In addition, we define a new decision variable ϵ to denote the maximum SC violation over all cycles in $C(L)$, equivalently all SCs in (7). The resulting min-max stability violation LP (MMSV) is given by Eqs. (41)–(45).

$$\text{MMSV}: \min \quad \varepsilon \tag{41}$$

$$\text{s.t.} \quad \sum_{l\in L} w_l = F\left(L\right) \tag{42}$$

$$\sum_{l\in L} b_{lc} w_l \leqslant f_c + \varepsilon \quad \forall c \in C\left(L\right) \tag{43}$$

$$w_l \in \mathbb{R} \quad \forall l \in L \tag{44}$$

$$\varepsilon \in \mathbb{R} \tag{45}$$

The objective (41) is to minimize the maximum amount of stability violation. Constraints (42) ensure that the total allocated cost is equal to the total cost from the SPIP. Constraints (43) ensure that ε is greater than or equal to the violation amount of any coalition. Constraints (44) and (45) define the domains of the decision variables.

Unlike the CSIP, the MMSV does not consider any lower and upper bounds on the allocation. In order to analyze the impact of the bounds we use, we added the following constraints to the MMSV. We will refer to this new model as the modified MMSV.

Table 8
Total process time (seconds) for row and column generation with and without heuristics

Instance	Without Heuristics	With Heuristics	% Improvement
1	162.7	106.1	−34.8
2	182.3	156.8	−14.0
3	172.0	141.6	−17.7
4	184.0	167.4	−9.0
5	245.2	186.4	−24.0
6	297.9	242.4	−18.6
7	588.5	562.4	−4.4
8	522.2	419.9	−19.6
9	325.5	274.3	−15.7
10	375.1	291.7	−22.2
11	504.0	425.6	−15.6
12	959.5	386.3	−59.7
13	780.3	688.6	−11.8
14	593.3	568.2	−4.2
15	868.0	679.4	−21.7
16	882.8	802.7	−9.1
17	1,169.2	1,018.8	−12.9
18	1,081.5	1,016.8	−6.0
19	809.5	551.2	−31.9
20	1,916.3	1,297.4	−32.3
21	1,189.7	895.9	−24.7
22	1,249.9	833.1	−33.3
23	1,255.4	1,117.7	−11.0
24	1,228.1	1,064.3	−13.3
25	1,626.7	1,335.7	−17.9
26	2,141.7	1,634.3	−23.7
27	2,726.5	1,810.5	−33.6
28	1,915.7	1,465.6	−23.5
29	1,877.8	1,386.0	−26.2
30	2,057.5	1,534.5	−25.4
Average	996.3	768.7	−20.6

$$w_l \leqslant (1-\theta_l) g_l \qquad \forall l \in L \tag{46}$$

$$w_l \geqslant \lambda_l d_l \qquad \forall l \in L \tag{47}$$

Constraints (46) guarantee a percentage savings for each lane. They also ensure individual rationality. Constraints (47) define lower bounds on allocated cost for each lane.

We solved the SPIP, the LP-SPIP, the MMSV, and the modified MMSV on the 30 instances we randomly generated. Table 4 summarizes our results. We report the LCP Gap in the second column. The LCP Gap shows the percentage gap between the objective function values of the (unconstrained) LCP and the SPIP, as a percentage of the optimal LCP objective, which provides a lower bound for the optimal objective values of the SPIP, the LP-SPIP, and the CSIP. We obtained the highest LCP Gap for instance 27 which is 2.3. The commercial solver yielded optimal solutions for all of the instances within the time limit. We could not obtain any integer-feasible solution for instance 17 since our workstation ran out of memory.

In Table 5, we present the optimal objective values of both the SPIP and the LP-SPIP, the difference between the two objective values, as well as the optimal objective values of the MMSV and the modified MMSV. We obtained the same objective value from the SPIP as the LP-SPIP for eight of the 30 instances. As expected, the MMSV yielded a cost allocation in the core for these instances. The CLC-LCG had an empty core for the remaining 22 instances. Furthermore, the modified MMSV did not yield any cost allocation in the core for any of the instances. We observe that the stability violation in the MMSV tended to decrease when the proportion of points placed in clusters (CPR) increases. Our results show that grand coalitions with empty cores is pervasive CLC-LCG, and additional restrictions on the allocated costs gravely deteriorates the core stability of the grand coalition. Careful selection of the lanes to be included in the coalition is needed. The CSIP addresses this need by integrating lane selection, cost optimization, and cost allocation decisions with core stability in mind.

6.2. Discussion on the potential impact of solving the SPIP heuristically

The proof of Lemma 1 holds iff the SPIP is solved optimally. If the SPIP is solved heuristically, then the resulting game is not necessarily subadditive and the grand coalition does not necessarily minimize total cost. However, it has no impact on the validity and the value of our study. The proofs and the implications of Lemmas 2–4 still hold, which form the basis of our analysis and computational experiments.

Table 9
Comparative results on the CSIP by solving the BnRCG with and without the RBH

Instance	BnRCG			BnRCG + RBH		
	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)	$\lvert\widehat{C}(L)\rvert$	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)	$\lvert\widehat{C}(L)\rvert$
1	2.4	86.0	31	2.8	84.0	31
2	25.2	23.0	9	1.1	84.0	27
3	2.5	95.0	32	2.5	95.0	32
4	1.4	83.0	30	1.4	83.0	29
5	4.5	76.0	24	2.8	79.0	26
6	1.6	85.0	30	9.7	69.0	23
7	2.4	67.3	36	3.7	60.7	33
8	2.8	69.3	35	2.8	68.7	36
9	6.3	64.0	38	6.5	63.3	37
10	5.2	67.3	36	7.5	62.0	34
11	5.9	59.3	33	5.1	62.7	34
12	1.2	86.7	48	1.7	85.3	46
13	1.9	78.5	60	1.8	79.0	60
14	2.3	79.5	61	2.3	80.0	62
15	3.0	76.5	54	3.0	76.5	54
16	–	–	–	18.4	43.0	29
17	13.4	47.0	28	11.0	52.0	31
18	6.4	68.0	47	8.2	64.0	45
19	42.0	17.3	11	29.9	33.3	20
20	4.0	75.3	37	4.0	75.3	37
21	1.6	88.0	48	1.5	90.7	48
22	1.0	80.7	43	5.1	68.7	37
23	7.6	61.3	27	5.9	64.7	27
24	1.4	84.0	41	1.4	84.0	41
25	2.3	87.5	60	1.1	91.5	61
26	4.5	70.0	43	34.9	22.5	18
27	–	–	–	38.7	18.5	13
28	3.6	82.0	65	3.6	82.5	65
29	6.1	73.5	52	8.1	69.0	50
30	–	–	–	39.6	25.5	17

Thanks to Lemma 2, checking the stability constraints for the set of feasible cycles is sufficient, eliminating the need to check them for larger subsets of L. Since a feasible cycle is expected to have a few lanes, its optimal cost (i.e. exact-SPIP solution) can be determined with little computational effort. Thus, we can still take advantage of Lemma 2 in the case of a heuristic-SPIP game. In order to discuss the impact of a heuristic-SPIP solution on Lemma 3, let $F^h(S)$ be the best solution identified by some arbitrary SPIP heuristic h on the lane set $S \subseteq L$. Clearly, $F(S) \leqslant F^h(S)$, which implies that $\sum_{l\in L} w_l^* \leqslant Z(L) \leqslant F^h(L)$. Since $F^h(S)$ may be strictly greater than $F(S)$, we may have $\sum_{l\in L} w_l^* \leqslant Z(L) < F^h(L)$, which means that we may not achieve budget balance in a heuristic-SPIP game, even when the core of its exact-SPIP counterpart is non-empty. Thus, using a heuristic to solve the SPIP makes it harder to identify a cost allocation in the core. Furthermore, we can expect core stability to get harder to achieve as the coalition size increases. Lemma 4 and its proof would hold directly without any modifications in a heuristic-SPIP game.

In light of the above, we expect that our experimental results would be impacted as follows if the SPIP were solved heuristically. In Table 5, we could potentially see higher differences between the LP-SPIP and the SPIP objective values, empty cores for at least some of the instances with non-empty cores, and higher objective values for the MMSV and the modified MMSV. We would not see any changes in the results presented in Tables 6–12, since the models used in the relevant experiments do not use pre-computed SPIP solutions.

6.3. Solving the CSIP by brute force and using the BnRCG

We conducted two sets of experiments to assess the computational effort involved in solving the CSIP. In the first, we tried to solve the CSIP by the brute force method (BFM), i.e. over the fully enumerated set of feasible cycles, with a *time limit of two hours*. In the second, we ran the BnRCG on the instances. We did not apply a time limit at the root node of the BnRCG tree, but we applied a time limit of *two minutes* on the RBH and *two hours* on the remaining nodes of the BnRCG tree.

We present the results we obtained from the BFM and the BnRCG for the CSIP in Tables 6 and 7, respectively. Table 6 provides information about the LCP gap and the process time for both the BFM and the BnRCG. In the second and fifth columns, we report the respective LCP Gap which represents the percentage between the objective value of the LCP and the objective value of the CSIP obtained by the two methods. Since we can solve the base (unconstrained) LCP in polynomial time, we used the optimal LCP objective value as a lower bound in our comparison for both the BFM and the BnRCG. The BnRCG yielded better LCP gaps than the BFM in all but one of the instances. Note that a trivial feasible CSIP solution can be obtained by excluding all lanes from the coalition, which we call it

Table 10
Impact of λ_l on the BnRCG solution to the CSIP ($\theta_l = 0.05$).

	$\lambda_l = 0.0$		$\lambda_l = 0.2$		$\lambda_l = 0.5$		$\lambda_l = 1.0$	
Instance	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$(%)	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)
1	3.0	83.0	2.8	84.0	3.9	78.0	16.2	56.0
2	1.3	83.0	1.1	84.0	6.6	58.0	12.8	46.0
3	2.5	95.0	2.5	95.0	7.8	74.0	16.4	63.0
4	1.4	83.0	1.4	83.0	1.1	84.0	12.2	57.0
5	3.0	80.0	2.8	79.0	3.0	80.0	8.6	73.0
6	9.7	69.0	9.7	69.0	1.8	85.0	17.7	55.0
7	2.3	68.0	3.7	60.7	2.5	66.7	11.3	46.0
8	2.8	69.3	2.8	68.7	2.6	70.7	9.3	52.0
9	14.7	50.7	6.5	63.3	5.7	65.3	12.6	58.0
10	4.8	68.7	7.5	62.0	11.4	55.3	13.4	56.0
11	5.8	68.7	5.1	62.7	4.3	55.3	11.0	50.7
12	1.2	86.7	1.7	85.3	3.2	78.7	14.1	62.7
13	1.8	79.0	1.8	79.0	19.1	39.5	18.4	47.0
14	2.2	80.5	2.3	80.0	4.9	70.0	15.6	48.0
15	3.0	76.5	3.0	76.5	3.7	74.0	17.5	48.0
16	5.6	67.5	18.4	43.0	7.8	64.0	18.8	44.5
17	14.3	45.5	11.0	52.0	12.6	48.0	17.8	40.0
18	7.7	65.0	8.2	64.0	27.9	35.5	18.6	50.5
19	31.6	34.0	29.9	33.3	25.1	44.0	17.2	59.3
20	4.0	75.3	4.0	75.3	5.4	73.3	16.8	50.7
21	6.8	69.3	1.5	90.7	2.8	80.7	13.9	60.7
22	1.0	80.7	5.1	68.7	10.6	56.7	11.3	54.7
23	5.9	64.7	5.9	64.7	5.7	66.0	16.3	50.7
24	1.4	84.0	1.4	84.0	7.3	68.0	16.0	57.3
25	1.9	88.5	1.1	91.5	39.5	23.5	30.4	38.5
26	37.7	19.5	34.9	22.5	32.0	26.5	24.5	39.5
27	37.6	20.5	38.7	18.5	37.1	20.5	26.2	38.5
28	6.3	75.0	3.6	82.5	40.7	25.5	23.0	50.0
29	42.8	22.0	8.1	69.0	43.8	22.0	30.8	38.5
30	41.1	23.0	39.6	25.5	41.0	23.0	33.4	35.0
Average	10.2	65.9	8.9	67.2	14.0	57.1	17.4	50.9

the worst feasible solution (WFS). The BFM yielded optimal solution for only one of the instances within the two-hour time limit while it obtained the WFS for 24 instances. While the BFM exceeded the time limit for 29 of the 30 instances, the BnRCG reached the two-hour time limit for all of the instances. We denote the process time of such instances as *TL* in the table.

Table 7 shows the number of generated cycles and the number of covered lanes and cycles for the BFM and the BnRCG. Since the BFM yielded the WFS for 24 instances, there were no covered lanes and cycles for these instances. Conversely, the BnRCG could find solutions which cover 67.2% of lanes on average (not shown in the table). In other words, the BnRCG had a higher proportion of lanes included in the coalition (seen in the sixth column as $\lvert\widehat{L}\rvert/\lvert L\rvert(\%)$). As expected, when the number of lanes increased, the number of cycles grew exponentially for the BFM. In the last column, we report the number of cycles generated during the run of the BnRCG, which involves column and row generation procedures. We observe that only a small fraction of the set of feasible cycles was generated by the BnRCG. Overall, we conclude that the BnRCG provided better solutions in much less computation time than the BFM. Also, the BnRCG selected a higher number of lanes and cycles for the coalition than the BFM.

6.4. Impact of row and column generation heuristics

In order to analyze the impact of the heuristics we use in row and column generation on process times, we conducted two sets of experiments in which we solve the LP relaxation of the CSIP (namely the CSLP) with and without the heuristics. In the first set of experiments, we only used the integer models (the RGSP and the CGPSP) to generate rows and columns. In the second set of experiments, we used the heuristics (see Section 5.1.1) in addition to the integer models.

In Table 8, we present the total process time (in seconds) spent in row and column generation without and with the heuristics in the second and the third columns, respectively. The last column of the table shows the percentage improvement (%Improvement) compared to the integer models on total process time. As expected, using only integer models took more time for all of the instances. The heuristics reduced the total process time by %20.6 on average. We obtained a minimum improvement rate of %4.2 which was on instance 14. The results demonstrate that the proposed heuristics significantly reduce the time spent in row and column generation.

Table 11
Impact of cost savings guarantee θ_l on the CSIP ($\lambda_l = 0.2$)

	$\theta_l = 0\%$		$\theta_l = 5\%$		$\theta_l = 10\%$		$\theta_l = 20\%$	
Instance	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$(%)	LCP Gap	$\lvert\widehat{L}\rvert/\lvert L\rvert$ (%)
1	1.0*	95.0*	2.8	84.0	2.4	84.0	13.0	60.0
2	0.4*	96.0*	1.1	84.0	6.8	57.0	10.4	51.0
3	2.4*	96.0*	2.5	95.0	4.2	84.0	9.4	74.0
4	0.4*	96.0*	1.4	83.0	2.7	76.0	8.4	63.0
5	0.1*	97.0*	2.8	79.0	1.6	84.0	4.7	74.0
6	0.1*	98.0*	9.7	69.0	1.0	87.0	6.9	74.0
7	0.6*	96.0*	3.7	60.7	8.1	47.3	9.6	50.0
8	1.1*	95.3*	2.8	68.7	5.5	59.3	7.5	54.0
9	0.3*	98.0*	6.5	63.3	5.7	65.3	5.1	67.3
10	0.7*	97.3*	7.5	62.0	11.8	55.3	16.6	46.0
11	0.3	89.3	5.1	62.7	2.9	68.7	8.7	56.0
12	0.3	98.7	1.7	85.3	3.2	79.3	8.3	69.3
13	0.8	94.0	1.8	79.0	10.5	54.0	16.8	42.5
14	2.9	78.0	2.3	80.0	6.6	64.5	9.2	62.0
15	0.4	94.5	3.0	76.5	9.9	60.5	18.3	47.0
16	0.6	95.0	18.4	43.0	9.4	59.5	13.8	49.5
17	0.1*	98.0*	11.0	52.0	22.1	33.0	24.1	30.0
18	0.1*	99.0*	8.2	64.0	30.0	34.0	14.0	52.0
19	0.7	93.3	29.9	33.3	3.7	79.3	28.9	34.0
20	1.7	97.3	4.0	75.3	4.0	76.0	6.6	67.3
21	2.3	83.3	1.5	90.7	2.1	86.0	7.4	68.7
22	18.9	46.0	5.1	68.7	15.2	50.7	6.0	66.7
23	0.1*	98.7*	5.9	64.7	4.9	66.0	24.0	34.0
24	38.2	24.0	1.4	84.0	2.2	81.3	9.9	64.0
25	40.7	21.5	1.1	91.5	1.1	92.0	41.5	20.5
26	37.3	18.5	34.9	22.5	34.9	23.0	35.5	22.0
27	38.6	18.0	38.7	18.5	37.2	20.5	36.6	21.0
28	1.7	96.5	3.6	82.5	7.0	72.5	38.7	29.5
29	0.2*	99.0*	8.1	69.0	5.8	72.5	8.1	70.0
30	1.4	86.0	39.6	25.5	38.5	26.5	7.1	73.0
Average	6.5	83.1	8.9	67.2	10.0	63.3	15.2	53.1

*: Solved before reaching the time limit

6.5. Impact of the RBH

Table 9 summarizes comparative results to the CSIP by using the BnRCG with/without the RBH to analyze the impact of the RBH. Due to the size of the instances, the BnRCG without the RBH did not yield any core stable solution for three instances with 200 lanes within the time limit. Conversely, the proposed solution method included 18.5% of lanes in the coalition in the worst case. Using the BnRCG with the RBH yielded better solutions for large instances in terms of LCP Gap which is 8.9 on average. We can conclude that the RBH helps to improve solution quality of the BnRCG to large instances by providing upper bounds.

6.6. Impact of λ_l and θ_l

We evaluated the impact of the parameters λ_l and θ_l on the final solutions returned by the BnRCG, by running the BnRCG with different values of the λ_l and θ_l. Recall that the lower bound λ_l helps us prevent very low, including zero and negative, costs to be allocated to the lanes, while θ_l is used to set upper bounds on the allocated costs to guarantee a minimum cost savings percentage. Note that λ_l and θ_l are design parameters that would be determined by the relevant decision maker(s).

In order to analyze the impact of λ_l and θ_l on coalition selection, we ran the BnRCG on our test instances with four different values of each parameter while keeping the other one fixed. More specifically, we varied the value of λ_l between $\{0.0, 0.2, 0.5, 1.0\}$ with $\theta_l = 5\%$, and then we varied the value of θ_l between $\{0\%, 5\%, 10\%, 20\%\}$ with $\lambda_l = 0.2$. We compared the results in terms of LCP Gap and $\lvert\widehat{L}\rvert/\lvert L\rvert$ (%).

We present the results of our experiments for measuring the impact of λ_l on the BnRCG solution to the CSIP in Table 10. When $\lambda_l = 0.0$, the LCP Gap values for 14 of 30 the instances increase, compared to the LCP Gap values when $\lambda = 0.2$. We attribute the increased LCP Gap value to decreased speed of convergence as a result of relaxed constraints due to lower θ_l value and the time limit. With $\lambda_l = 0.0$, we obtained the same solutions as with $\lambda_l = 0.2$ for 9 of the 30 instances. When we set λ_l equal to 0.5 or 1.0, LCP Gap increased, and fewer lanes were included in the coalition, as expected since the constraints of the CSIP became tighter. On average, we obtained

Table 12
Impact of θ_l and λ_l on the optimal value of the modified MMSV

	θ_l (with $\lambda_l = 0.2$)				λ_l (with $\theta_l = 5\%$)			
Instance	0%	5%	10%	20%	0.0	0.2	0.5	1.0
1	0.29	6.15	21.14	109.16	6.15	6.15	29.37	183.87
2	0.00	14.48	69.18	325.78	14.48	14.48	38.29	288.49
3	0.11	7.36	21.23	74.54	7.36	7.36	12.26	96.80
4	0.03	13.25	49.51	151.56	13.25	13.25	42.30	215.06
5	0.15	5.15	17.17	65.64	5.15	5.15	12.52	87.02
6	0.00	8.89	28.56	87.73	8.89	8.89	8.89	150.16
7	0.00	47.39	147.23	481.94	47.39	47.39	95.29	382.52
8	0.22	54.59	152.53	485.09	54.59	54.59	90.25	366.52
9	0.00	11.91	36.34	120.35	11.91	11.91	18.66	131.94
10	0.00	23.95	63.87	229.78	23.95	23.95	61.29	287.29
11	0.13	35.02	108.67	305.00	35.02	35.02	68.72	334.02
12	0.05	6.59	20.03	80.86	6.59	6.59	12.96	109.94
13	0.22	13.56	42.22	157.34	13.56	13.56	34.02	215.10
14	0.10	10.04	28.68	103.28	10.04	10.04	18.50	192.02
15	0.05	4.71	19.00	86.96	4.71	4.71	5.43	128.44
16	0.01	22.80	69.93	216.31	22.80	22.80	46.31	208.96
17	–	–	–	–	–	–	–	–
18	0.00	10.81	37.30	117.17	10.81	10.81	40.67	180.75
19	0.76	6.32	19.47	78.99	6.32	6.32	15.23	117.90
20	0.00	20.40	66.14	198.53	20.40	20.40	66.27	279.74
21	0.19	8.74	29.16	109.42	8.74	8.74	15.42	169.15
22	0.24	26.70	69.49	191.73	26.70	26.70	74.75	307.70
23	0.05	20.99	57.27	184.41	20.99	20.99	26.49	188.71
24	0.31	8.98	23.36	73.55	8.98	8.98	16.10	132.26
25	0.21	4.21	10.23	38.60	4.21	4.21	4.21	73.29
26	0.17	10.82	33.81	119.70	10.82	10.82	14.71	150.29
27	0.36	9.67	27.08	100.53	9.67	9.67	15.59	121.55
28	0.05	11.74	28.97	84.36	11.74	11.74	31.17	171.66
29	0.00	4.58	14.98	65.14	4.58	4.58	19.34	156.06
30	0.06	7.13	22.65	81.40	7.13	7.13	22.47	130.03
Average	0.13	15.07	46.04	156.03	15.07	15.07	33.02	191.63

the highest LCP Gap and the lowest $\left|\widehat{L}\right|/\left|L\right|$ (%) when $\lambda_l = 1.0$. Note that the BnRCG reached two hours time limit for all λ_l values. We can conclude that when the bounds become tighter, $\left|\widehat{L}\right|/\left|L\right|$ (%) tends to decrease in general.

Table 11 contains the results of our experiments for evaluating the impact of θ_l on the BnRCG solution to the CSIP. When $\theta_l = 0\%$, there was no cost savings guarantee for the lanes in the coalition, i.e. constraints on the savings were relaxed. Hence, more lanes could be selected for the coalition. Conversely, when θ_l increased, the related constraints became tighter. As expected, the number of lanes included in the coalition increased when there was no saving guarantee. Since there were more lanes in the coalition, LCP Gap was lowest when $\theta_l = 0\%$. When we gave the lanes a higher cost savings guarantee, LCP Gap became higher. On average, we were able to include 83.1%, 67.2%, 63.3%, and 53.1% of the lanes in the coalition for θ_l values of 0%, 5%, 10%, and 20%, respectively.

We also conducted experiments to analyze the impact of λ_l and θ_l on the core stability of the grand coalitions of our test instances. We solved the modified MMSV introduced in Section 6.1 under the same set of values of λ_l and θ_l on the SPIP solutions discussed in the same section. Recall that the optimal solution of the modified MMSV yields the minimum SC violation possible under budget-balance, lower bound, and upper bound constraints on the allocated costs. Table 12 shows the optimal objective value of the modified MMSV. We clearly see that increasing either θ_l or λ_l lead to higher optimal objective values for the modified MMSV, which means that the grand coalition became more unstable. We can interpret λ_l as an indication of the level of egalitarianism expected from the allocated costs. We can interpret θ_l as an indication of cost savings expectations. Our results imply that if the cost savings or egalitarianism expectations are high, the grand coalition faces an increased risk of attrition and eventual collapse. By taking such expectations into account from the beginning, the CSIP enables a higher level of stability and foreseeability in the coalition formation process.

7. Conclusions

In this paper, we presented our research on coalition selection in CTTP. We modeled a cooperative game, namely the CLC-LCG, and showed the necessary and sufficient conditions for the grand coalition to be core stable. These conditions imply that an arbitrarily selected grand coalition is likely to have a non-empty core, which threatens the persistence of the collaboration and its benefits. Our computational experiments on randomly generated instances confirmed that grand coalitions with non-empty cores are pervasive in the CLC-LCG. We observe that finding a cost allocation in the core that also satisfies the lower and upper bounds was even harder. Our

proposed model, the CSIP, was built on the idea of achieving a core stable coalition by including the right lanes in the coalition. The CSIP selects a core stable coalition with minimum total cost by making lane selection, cycle cover, and cost allocation decisions in an integrated manner. We developed the BnRCG to solve the CSIP, since solving the CSIP is a challenging task, and demonstrated its computational efficiency on randomly generated CLC-LCG instances.

The CSIP and the solution approach we propose can be adopted to cost allocation games where the cost of a coalition is given by the solution of a set covering/partitioning problem. In particular, adopting our approach to closely related settings like the vehicle routing game and the ridesharing game would be fairly straightforward. The CLC-LCG shares important features with these settings. Specifically, (i) each feasible solution consists of a set of disjoint routes/(mini-) coalitions, and (ii) the characteristic function value for each coalition is equal to the sum of the costs of the routes/(mini-) coalitions making up the optimal feasible solution for the coalition. Column generation methods for set partitioning/covering formulations of the vehicle routing problem have been extensively studied in the literature. Row generation methods for the vehicle routing game have also been studied. Lu and Quadrifoglio (2019) developed row generation methods for the ridesharing game. Since the generated rows and columns of the CSIP correspond to the set of feasible routes, any row or column generation method can be used for solving the CSIP, albeit with different objective functions. The constraints stemming from the depot, subtour elimination in particular, in the vehicle routing setting is likely to cause a more computationally challenging column/row generation subproblem. The computational tractability of the ridesharing game is much less studied, but we expect it to be less computationally challenging than the vehicle routing setting due to the lack of a depot.

Our coalition selection idea can be generalized to cooperative games in other settings where the grand coalition is not always core stable. A good example is collaborative hub networks involving many shippers and/or carriers. Likewise, collaborative supply chain distribution networks constitute another potential application area. Including the right facilities, customers, and/or origin-destination pairs with benefit maximization and sharing in mind would allow longevity of such collaborative settings. Note that the CSIP in effect eliminates parts of the grand coalition that do not fit well with the rest. Hence, although probably not obvious, our approach can be applied in non-collaborative settings to determine which parts fit well/badly with the overall business of a company. It can be used to determine the parts to be outsourced, cut, or subsidized. An interesting example would be service network design, which has recently been attracting an increasing level of attention from researchers. So far, we have mainly focused on forming a coalition from scratch. Our approach can also be used for evaluation of potential new collaborators, or potential new business in more traditional settings, as well. Additional constraints may be needed to keep the existing players in the coalition and content when evaluating expansion candidates.

Several potential research directions can be derived from the work we presented in this paper. We have assumed that only one coalition will be formed, and the lanes left out of the coalition will incur a high cost. We can further reduce the system-wide cost by partitioning the lanes of the grand coalition into multiple coalitions. On the other hand, considering the formation of multiple coalitions adds another layer of complexity to our models. In the near future, we plan to study the problem of identifying optimal or near-optimal coalitional structure for constrained LCGs as an extension of this paper. Aside from the LCGs, coalition selection models for non-CTTP applications with different characteristic functions, some of which we discussed above, would be worth investigating. Another possible future research direction is the characterization of the instances that lead to larger coalitions, which would require a larger number of problem instances with richer features and could be carried out using data science and machine learning methods.

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.

Acknowledgments

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

References

Agarwal, R., Ergun, Ö., 2010. Network design and allocation mechanisms for carrier alliances in liner shipping. Oper. Res. 58, 1726–1742.

Akyol, D.E., De Koster, R.B., 2018. Determining time windows in urban freight transport: A city cooperative approach. Transp. Res. Part E: Logist. Transp. Rev. 118, 34–50.

Anesi, V., 2010. Noncooperative foundations of stable sets in voting games. Games Econ. Behav. 70, 488–493.

Anshelevich, E., Dasgupta, A., Kleinberg, J., Tardos, E., Wexler, T., Roughgarden, T., 2008. The price of stability for network design with fair cost allocation. SIAM J. Comput. 38, 1602–1623.

Audy, J.F., D'Amours, S., Lehoux, N., Rönnqvist, M., 2010. Generic Mechanisms for Coordinating Operations and Sharing Financial Benefits in Collaborative Logistics. In: Camarinha-Matos, L., Boucher, X., Afsarmanesh, H. (Eds.), Collaborative Networks for a Sustainable World. Springer Boston, IFIP Advances in Information and Communication Technology, vol. 336, pp. 537–544.

Aumann, P.R.J., Dreze, P.J.H., 1974. Cooperative games with coalition structures. Int. J. Game Theory 3 (217–237), 00667.

Bachrach, Y., Elkind, E., Meir, R., Pasechnik, D., Zuckerman, M., Rothe, J., Rosenschein, J.S., 2009. Algorithmic Game Theory: Second International Symposium, SAGT 2009, Paphos, Cyprus, October 18–20, 2009. Proceedings. Springer, Berlin Heidelberg, Berlin, Heidelberg. chapter The Cost of Stability in Coalitional Games. pp. 122–134.

Banerjee, S., Konishi, H., Sönmez, T., 2001. Core in a simple coalition formation game. Soc. Choice Welfare 18, 135–153.

Basso, F., Basso, L.J., Rönnqvist, M., Weintraub, A., 2021. Coalition formation in collaborative production and transportation with competing firms. Eur. J. Oper. Res. 289, 569–581. https://doi.org/10.1016/j.ejor.2020.07.039. URL: https://www.sciencedirect.com/science/article/pii/S0377221720306652.

Basso, F., Guajardo, M., Varas, M., 2020. Collaborative job scheduling in the wine bottling process. Omega 91, 102021.

Bondareva, O.N., 1963. Some applications of linear programming methods to the theory of cooperative games. Problemy kibernetiki 10, 119–139.
Cleophas, C., Cottrill, C., Ehmke, J.F., Tierney, K., 2019. Collaborative urban transportation: Recent advances in theory and practice. Eur. J. Oper. Res. 273, 801–816.
Creemers, S., Woumans, G., Boute, R., Beliën, J., 2017. Tri-Vizor Uses an Efficient Algorithm to Identify Collaborative Shipping Opportunities. Interfaces 47, 244–259. https://doi.org/10.1287/inte.2016.0878. URL: http://pubsonline.informs.org/doi/10.1287/inte.2016.0878.
Elomri, A., Ghaffari, A., Jemai, Z., Dallery, Y., 2012. Coalition formation and cost allocation for joint replenishment systems. Prod. Oper. Manage. 21, 1015–1027.
Ergun, O., Kuyzu, G., Savelsbergh, M., 2007a. Reducing truckload transportation costs through collaboration. Transp. Sci. 41 (206–221), 00083.
Ergun, O., Kuyzu, G., Savelsbergh, M., 2007b. Shipper collaboration. Comput. Oper. Res. 34, 1551–1560.
Gale, D., Shapley, L.S., 1962. College admissions and the stability of marriage. Am. Math. Monthly 69, 9–15.
Gale, D., Sotomayor, M., 1985. Some remarks on the stable matching problem. Discrete Appl. Math. 11, 223–232.
Göthe-Lundgren, M., Jörnsten, K., Värbrand, P., 1996. On the nucleolus of the basic vehicle routing game. Math. Programm. 72, 83–100.
Guajardo, M., Jörnsten, K., 2015. Common mistakes in computing the nucleolus. Eur. J. Oper. Res. 241, 931–935. https://doi.org/10.1016/j.ejor.2014.10.037.00015. URL: http://linkinghub.elsevier.com/retrieve/pii/S0377221714008595.
Guajardo, M., Rönnqvist, M., 2015. Operations research models for coalition structure in collaborative logistics. Eur. J. Oper. Res. 240 (147–159), 00002.
Hezarkhani, B., Slikker, M., Van Woensel, T., 2014. On characterization of the core of lane covering games via dual solutions. Oper. Res. Lett. 42, 505–508.
Jouida, S.B., Guajardo, M., Klibi, W., Krichen, S., 2020. Profit maximizing coalitions with shared capacities in distribution networks. Eur. J. Oper. Res. 288, 480–495.
Jouida, S.B., Krichen, S., Klibi, W., 2017. Coalition-formation problem for sourcing contract design in supply networks. Eur. J. Oper. Res. 257, 539–558.
Kuyzu, G., 2017. Lane covering with partner bounds in collaborative truckload transportation procurement. Comput. Oper. Res. 77, 32–43.
Kuyzu, G., 2020. A hyrid genetic algorithm approach for solving partner constrained lane covering problems. Dokuz Eylul Univ. Faculty Eng. J. Sci. Eng. 22, 401–416. https://doi.org/10.21205/deufmd.2020226509.
Liu, L., Qi, X., Xu, Z., 2018. Simultaneous Penalization and Subsidization for Stabilizing Grand Cooperation. Oper. Res. https://doi.org/10.1287/opre.2018.1723. URL: https://pubsonline.informs.org/doi/abs/10.1287/opre.2018.1723.
Lu, W., Quadrifoglio, L., 2019. Fair cost allocation for ridesharing services – modeling, mathematical programming and an algorithm to find the nucleolus. Transp. Res. Part B: Methodol. 121, 41–55. https://doi.org/10.1016/j.trb.2019.01.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S019126151730944X.
Öner, N., Kuyzu, G., 2020. Nucleolus based cost allocation methods for a class of constrained lane covering games. Unpublished manuscript.
Osicka, O., Guajardo, M., van Oost, T., 2020. Cooperative game-theoretic features of cost sharing in location-routing. Int. Trans. Oper. Res. 27, 2157–2183. https://doi.org/10.1111/itor.12698. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12698.
Özener, O., Ergun, O., 2008. Allocating Costs in a Collaborative Transportation Procurement Network. Transp. Sci. 42, 146–165.
Ray, D., Vohra, R., 2015. Coalition formation. Handbook of game theory with economic applications, vol. 4. Elsevier, pp. 239–326.
Shapley, L.S., 1967. On balanced sets and cores. Naval Res. Logist. Quart. 14, 453–460.
Shapley, L.S., 1971. Cores of convex games. Int. J. Game Theory 1, 11–26.
Vanovermeire, C., Sörensen, K., 2014. Measuring and rewarding flexibility in collaborative distribution, including two-partner coalitions. Eur. J. Oper. Res. 239 (157–165), 00000.
Wang, J., Yu, Y., Tang, J., 2018a. Compensation and profit distribution for cooperative green pickup and delivery problem. Transp. Res. Part B: Methodol. 113, 54–69.
Wang, X., Agatz, N., Erera, A., 2018b. Stable matching for dynamic ride-sharing systems. Transp. Sci. 52, 850–867.
Wang, Y., Ma, X., Li, Z., Liu, Y., Xu, M., Wang, Y., 2017. Profit distribution in collaborative multiple centers vehicle routing problem. J. Clean. Prod. 144, 203–219.
Wang, Y., Yuan, Y., Guan, X., Xu, M., Wang, L., Wang, H., Liu, Y., 2020a. Collaborative two-echelon multicenter vehicle routing optimization based on state–space–time network representation. J. Clean. Prod. 258, 120590.
Wang, Y., Zhang, S., Guan, X., Peng, S., Wang, H., Liu, Y., Xu, M., 2020b. Collaborative multi-depot logistics network design with time window assignment. Expert Syst. Appl. 140, 112910.
Yang, F., Dai, Y., Ma, Z.J., 2020. A cooperative rich vehicle routing problem in the last-mile logistics industry in rural areas. Transp. Res. Part E: Logist. Transp. Rev. 141, 102024.