Spot collaborative shipping *sans* orchestrator using Blockchain

Kameshwaran Sampath*, Sai Koti Reddy Danda*, Ken Kumar†
Krishnasuri Narayanam*, Pankaj Dayama* and Suryanarayana Sankagiri‡
*IBM Research, India
{kameshwaran.s, d.saikotireddy, knaraya3, pankajdayama}@in.ibm.com
†Oracle IDC, India
ken.kumar@oracle.com
‡University of Illinois, USA
ss19@illinois.edu

***Abstract*—Collaborative shipping refers to multiple independent shippers consolidating their freight flows to share the same set of carriers for goods transportation, in order to reduce cost. In this paper, we design a collaborative shipping marketplace for spot shipments, where the shipping demands are bundled dynamically without premeditated planning by the shippers. Collaborative shipping requires a neutral third party as an orchestrator, who identifies the bundling opportunities, matches with the carrier, and allocates the cost among the shippers. We propose a Blockchain based application that enables decentralized and transparent participation of shippers and carriers in choosing the matching of bundled demand with supply, without an orchestrator. A multi-disciplinary approach with techniques from multi criteria optimization, group decision making, cooperative game theory, and electoral voting are used to develop the distributed collaborative shipping application using Blockchain.**

***Keywords*-Logistics marketplace; horizontal collaboration; collaborative decision making; multiobjective optimization; blockchain;**

I. Introduction

The trucking industry is a significant contributor to the commercial cargo movement. The industry is highly fragmented - in the US, 97% of carriers own less than 20 trucks and 90% own six or fewer trucks [1]. Due to the fragmentation, carriers and shippers have high search costs in supply-demand matching. Further, trucks are underutilized (*less-than-truckload*) and often run empty return miles. Trucking industry is also one of the significant contributors to greenhouse gas emissions. This paper focuses on Blockchain powered collaborative shipping marketplace that consolidates freights resulting in higher utilization of trucks and lower shipping costs.

Online freight marketplaces can be differentiated based on whether they cater to *spot* market or *contract* based long term partnerships. Spot market is for shippers with small scale or occasional or irregular transportation needs that are outside the scope of a long term contractual arrangement with carriers. In this paper, we consider collaborative shipping in *spot* markets. Compatible demands of shippers are consolidated and shipped together thereby reducing overall cost and increasing the utilization of the trucks. This is similar to UberPool where passengers heading the same direction are *bundled* by Uber to ride together and each pay cheaper fare than their respective individual rides.

TRI-VIZOR[1] is a cross supply chain orchestrator that proactively designs, prepares, and operates collaborative shipping among shippers. While its operations are popularly dubbed as *carpooling for cargo*, TRI-VIZOR identifies and enables collaboration among shippers for long term shipping (unlike UberPool that caters to spot demands). Our focus is on spot collaborative market for truck cargo, that enables transparent bundling and matching, without the need for an orchestrator like TRI-VIZOR.

Our main contributions are: i) design a collaborative shipping marketplace for spot shipments market comprising of bundling, cost allocation, and centralized multi objective matching (Section II); ii) propose a collaborative decision making technique that enables decentralized and transparent participation of shippers and carriers in choosing the matching of bundled demand with supply (Section III); and iii) implement the above decentralized marketplace using Blockchain (Sections IV and V).

II. Spot Collaborative Shipping Marketplace

Spot freight shipments are known few hours to few days in advance, with a determined window for pickup of shipments. Thus, the bundling opportunities can be identified in *batch* mode, at predetermined time slots (twice or thrice in a day). Several types of bundling opportunities can be identified: *onward* (origin $\rightarrow$ destination), *back-haul* (origin $\rightarrow$ destination $\rightarrow$ origin), *round-trips* (back to origin via multiple destinations), and more complex trips [2]. In this paper, we consider only the onward bundling (origin $\rightarrow$ destination), from a given origin to all the possible destinations. Also,

† The author contributed during his association with IBM Research
‡ The author contributed during his internship with IBM Research

[1]http://www.trivizor.com/

Table I
NOTATION

L	Set of all Locations	o^*_*, d^*_*	$\in L$ Origin, destination	α_i	Demand of shipper i
F	Set of all consignment types	D^*_*	$\subseteq L$ Set of destinations	γ_j	Supply of carrier j
G	$\subseteq F \times F$ Compatible consignments	w^*_*	$\in \mathbb{Z}^+$ Weight	β_k	Bundle of demands
h^*_*	$\in F$ Consignment type	v^*_*	$\in \mathbb{Z}^+$ Volume	$\eta(\cdot)$	$\in \mathbb{R}^+$ Market cost
H^*_*	$\subseteq F$ Consignment types	$\underline{t}^*_*, \overline{t}^*_*$	$\in \mathbb{Z}^+$ Start time, end time	$\zeta_j(\dot{})$	$\in \mathbb{R}^+$ Shipping cost of carrier j

Table II
DEMAND, SUPPLY, AND BUNDLE

		Supply γ_j:		Bundle β_k:	
Demand α_i		o^γ_j	Origin	Q_k	Set of demands $\{\alpha_i\}$
(o^α_i, d^α_i)	(Origin, Destination)	D^γ_j	Set of potential destinations	(o^β_k, d^β_k)	$o^\beta_k = o^\alpha_i$, $d^\beta_k = d^\alpha_i$, $\forall \alpha_i \in Q_k$
h^α_i	Consignment type	H^γ_j	Set of acceptable consignment types	H^α_k	$\cup_{\alpha_i \in Q_k} \{h^\alpha_i\}$
w^α_i	Weight	w^γ_j	Maximum weight	w^β_k	$\sum_{\alpha_i \in Q_k} w^\alpha_i$
v^α_i	Volume	v^γ_j	Maximum volume	v^β_k	$\sum_{\alpha_i \in Q_k} v^\alpha_i$
$[\underline{t}^\alpha_i, \overline{t}^\alpha_i]$	Load pickup time range	$\underline{t}^\gamma_j$	Start time availability	$[\underline{t}^\beta_k, \overline{t}^\beta_k]$	$\cap_{\alpha_i \in Q_k} [\underline{t}^\alpha_i, \overline{t}^\alpha_i] \neq \emptyset$
		ζ^γ_j	Shipment cost function	$\eta(\beta_k)$	$\sum_{\alpha_i \in Q_k} \eta(\alpha_i)$

we assume only the bundling of demand (shipments), which can be matched with a single truck (bundled shipments are not split among multiple trucks).

A. Notation

Table I lists the basic notation followed in the paper. The collaborative shipping marketplace caters to a set of locations L and consignment types F. Compatible consignment types are given by $G \subseteq F \times F$, to define the set of allowable consignment types that can be bundled and shipped together. The three basic entities that we will deal with are 1) demand α_i of shipper i, 2) supply γ_j of carrier j, and 3) bundle β_k of demands. The attributes shared by the above are consignment types (h^*_*, H^*_*), origin (o^*_*), destination (d^*_*, D^*_*), weight (w^*_*), volume (v^*_*), start time $(\underline{t}^*_*)$, and end time $(\overline{t}^*_*)$. The superscript in the attributes denotes the demand, supply, and bundle (α, γ and β respectively) and the subscript denotes the indices (i, j, k). Weight, volume, and time take integer values, as multiples of some predefined base units of respective measures.

There are two types of cost functions: 1) market cost function $\eta(\cdot)$ and 2) shipping cost function $\zeta_j(\cdot)$. Market cost function $\eta(\cdot)$ is assumed to be exogenously available, which is used to determine the typical cost of shipping without bundling for any demand. For example, shipper i would incur cost $\eta(\alpha_i)$ elsewhere for the shipment. Note that the *market* in the phrase *market cost function* denotes the large external market (probably with higher search and transaction costs) outside of the collaborative shipping marketplace, that serves as the best alternative for the shippers and carriers to find business. Henceforth, we will use *market* to refer to the external market and *marketplace* to refer to the proposed collaborative shipping marketplace. In order to gain through collaborative shipping, the final shipment cost for α_i should be less than that of the market cost. The shipping cost function $\zeta_j(\cdot)$ is endogenously given by carrier j as part of the supply γ_j.

B. Demand and Supply

Table II shows the notation of demand, supply, and bundles. For the sake of brevity, we will assume that shipper i (carrier j) submits only one demand α_i (supply γ_j), so that i (j) can denote either the shipper i (carrier j) or the demand α_i (supply γ_j), depending on the context. The demand is for a *less-than-truckload* requirement, implying that the shipper is willing to bundle the demand with other compatible demands, and the supply γ_j denotes a single truck.

The load pickup time range $[\underline{t}^\alpha_i, \overline{t}^\alpha_i]$ is the interval during which the shipment has to be picked-up. The supply start time availability $\underline{t}^\gamma_j$ indicates the earliest time by which the truck would be ready for shipping. There is no end time specified and if the carrier obtains a business elsewhere, can withdraw the supply order from the marketplace.

The shipment cost function $\zeta_j(\cdot)$, quoted by carrier j, is used to determine the cost of shipment. It can be composed of cost schedules defined as a function of the consignment type(s), distance, weight, volume, and size of the bundle. Without loss of generality, both the supply and market cost functions take as input the demand α_i or a set of demand profiles $\{\alpha_i\}$ or a bundle of demands β_k (to be defined next).

C. Bundling

As we consider only onward bundling of shipments, identification of bundling opportunities is separable with respect to the origin location. Let $\mathrm{A}_l = \{\alpha_i : o^\alpha_i = l\}$ be the set of demands originating from location l. The objective is to identify onward bundling opportunities $\mathrm{B}_l = \{\beta_k\}$ from A_l, where β_k is a set of *compatible* demands that can be bundled together. Any pair of demands $\alpha_{i'}$ and $\alpha_{i''}$ are compatible if the following conditions are satisfied: (i) same origin - destination: $(o^\alpha_{i'}, d^\alpha_{i'}) = (o^\alpha_{i''}, d^\alpha_{i''})$; (ii) consignment types $(h^\alpha_{i'}, h^\alpha_{i''}) \in G$ are *compatible* and can be shipped together; (iii) overlapping pick-up time range:$[\underline{t}^\alpha_{i'}, \bar{t}^\alpha_{i'}] \cap [\underline{t}^\alpha_{i''}, \bar{t}^\alpha_{i''}] \neq \emptyset$.

Let β_k denote a set of demands that are mutually compatible with respect to the above conditions. The attributes of bundle β_k are given in Table II. Figure 1 presents an algorithm to generate all possible bundles B_l from A_l for origin l, that satisfy the above compatibility conditions.

```
1: input:
       $\mathrm{A}_l = \{\alpha_i\}$
2: initialize:
       $k \leftarrow 0$
       $\mathrm{B}_l \leftarrow \emptyset$
3: for each $\alpha_i \in \mathrm{A}_l$ do
4:     $k \leftarrow k + 1$
5:     $\beta_k \leftarrow \{\alpha_i\}$
6:     $X \leftarrow \{\beta_k\}$
7:     for each $\beta_{k'} \in \mathrm{B}_l$ do
8:         if isCompatible($\beta_{k'}, \alpha_i$) then
9:             $k \leftarrow k + 1$
10:            $\beta_k \leftarrow$ addToBundle($\beta_{k'}, \alpha_i$)
11:            $X \leftarrow X \cup \beta_k$
12:        end if
13:    end for
14:    $\mathrm{B}_l \leftarrow \mathrm{B}_l \cup X$
15: end for
16: output:
       $\mathrm{B}_l = \{\beta_k\}$
```

Figure 1. Algorithm to generate all possible bundles from a set of demands

The algorithm iterates over each of the items and creates all possible bundles with the current set of items. The output B_l contains singleton bundles of demand profiles and hence every demand α_i belongs to at least one bundle in B_l. As the bundling and matching problem is separable with respect to the origin location l, remainder of the discussion in the paper pertains to a single origin. We will skip the index l wherever required for brevity and the set of demands ($\{\alpha_i\}$), supply ($\{\gamma_j\}$), and bundles ($\{\beta_k\}$) refer to the same origin location.

D. Cost Allocation

Let bundle β_k, with compatible demand profiles $Q_k = \{\alpha_i\}$, is matched with supply γ_j. The shipment cost for the bundle, as given by the supply profile γ_j, is $\zeta_j(\beta_k)$. The problem of apportioning the shipment cost among the shippers $i \in \beta_k$ is called as the *cost allocation* or *gain sharing* problem. Let $\phi : (k, j) \rightarrow (c^{kj}_i)_{i\in\beta_k}$ be the cost allocation mechanism that apportions the total shipment cost among the shippers, where c^{kj}_i is cost of shipping α_i in bundle β_k using supply γ_j. Cost allocation $\phi(k, j)$ should satisfy the following properties: $\sum_i c^{kj}_i = \zeta_j(\beta_k)$ and $c^{kj}_i \geq 0, \forall i$.

Cost allocation or gain sharing is a well studied problem in *cooperative game theory* [3]. To be consistent with the notion of *characteristic* cost function in cooperative games, we introduce the following notation. Let $N = \{i : \alpha_i \in Q_k\}$ be the set of shippers in the bundle β_k and $S \subseteq N$. Define the *characteristic* cost function C for β_k as follows:

$$C_N = \zeta_j(\beta_k) \tag{1}$$

$$C_S = \zeta_j(S) \tag{2}$$

$$C_i = \zeta_j(\alpha_i) \tag{3}$$

$$C_{-i} = C_{N\setminus i} \tag{4}$$

$$C_\emptyset = 0 \tag{5}$$

A cost allocation mechanism ϕ essentially uses the above characteristic function to determine the allocation $(c^{kj}_i)_{i\in\beta_k}$. Few of the desirable properties of ϕ are: *efficiency*: $\sum_i c^{kj}_i = C_N\ (= \zeta_j(\beta_k))$; *individual rationality*: $c^{kj}_i \leq C_i\ (= \zeta_j(\alpha_i))$; and *stability*: $\sum_{i\in S} c^{kj}_i \leq C_S, \forall S$

Stability is a stronger property that implies individual rationality. Shapely value based cost allocation that satisfy stability tend to be computationally intensive as the property is ensured at all subsets. In collaborative transport settings, mechanisms like *equal charge method*, *alternative cost avoided method*, *cost gap method*, and *equal profit method* are computationally manageable as they satisfy only efficiency and individual rationality, using C_N, C_i, C_{-i} [4]. While any of these methods can be used for cost allocation using the characteristic function C, following regularization is required.

In cooperative games, C_i in the characteristic cost function denotes the cost of shipper i not choosing to collaborate with other shippers, which is the market cost $\eta(\alpha_i)$. However, we have defined it above to be equivalent to the shipping cost of the singleton bundle ($C_i = \zeta_j(\alpha_i)$) that consists of only the demand α_i. Our definition uses the shipping cost in order to ensure the monotonous property of the characteristic function $C(S) \leq C(S')\forall S \subset S'$. As the market cost is obtained exogenously and shipping cost is defined endogenously, it is possible to have $\eta(\alpha_i) < \zeta_j(\alpha_i)$ for some demand α_i and supply γ_j (especially if the truck has very large capacity compared to that of the shipment). Hence, individual rationality should be satisfied with respect to the market cost $c^{kj}_i \leq \eta(\alpha_i)$, otherwise there is no incentive for the shipper to participate in collaborative shipping. For shipping cost functions with $\eta(\alpha_i) < \zeta_j(\alpha_i)$, individual rationality with respect to shipping cost does not necessarily imply that of with market cost. It is thus imperative to determine cost allocation $\phi(k, j) = (c^{kj}_i)$ prior to matching as only those (k, j) that satisfy individual rationality for all demand profiles with respect to market cost are eligible for matching. The above problem is due to the existence of two cost functions, while only one can be used to define the characteristic cost function with monotonous property.

E. Matching

Let Γ_l denote the set of all supply originating from location l. The objective is to match the *bundled* demand from B_l to supply Γ_l. The *demand bundle supply* (DBS) graph $(\mathrm{B}_l, \Gamma_l, E)$ is a bipartite graph generated from $\mathrm{B}_l = \{\beta_k\}$ and $\Gamma_l = \{\gamma_j\}$ as nodes, and with edges E. Edge $(k, j) \in E$ implies that the bundle β_k can be matched with the supply γ_j, if the following matching conditions are satisfied:

$$d_k^\beta \in D_j^\gamma \tag{6}$$
$$H_k^\beta \subseteq H_j^\gamma \tag{7}$$
$$w_k^\beta \leq w_j^\gamma \tag{8}$$
$$v_k^\beta \leq v_j^\gamma \tag{9}$$
$$\underline{t}_j^\gamma \in [\underline{t}_k^\beta, \overline{t}_k^\beta] \tag{10}$$
$$\zeta_j(\beta_k) < \sum_{i \in Q_k} \eta(\alpha_i) \tag{11}$$
$$c_i^{kj} < \eta(\alpha_i), \forall i \tag{12}$$

Conditions (6) - (10) ensure that constraints imposed with respect to destination, compatibility of consignment types, weight, volume, and pickup time are met, respectively. The market cost $\eta(\beta_k)$ of the bundle is the sum of market cost of individual demand profiles ($\sum_i \eta(\alpha_i)$), without leveraging the economies of scope and scale. Condition (11) ensures positive value from bundling and conditions (12) ensure individual rationality.

A matching $\omega \subseteq E$ that matches bundles with supply can be determined by formulating the matching problem as the following binary (0-1) integer programming problem. Let $x_{kj} \in \{0, 1\}$ denote the binary decision variable for $(k, j) \in E$. Matching $\omega = \{(k, j) : x_{kj} = 1\}$ is feasible if the following constraints are satisfied by (x_{kj}):

$$\sum_{(k,j) \in E} x_{kj} \leq 1, \forall j \tag{13}$$
$$\sum_{(k,j) \in E} x_{kj} \leq 1, \forall k \tag{14}$$
$$\sum_{k:\alpha_i \in Q_k} \sum_{(k,j) \in E} x_{kj} \leq 1, \forall i \tag{15}$$

Constraints (13) ensure that supply j is at most matched with one bundle, (14) ensure that demand bundle k is at most matched to one supply, and (15) are multiple choice constraints to ensure that demand i belongs to at most one matched bundle. The marketplace can *maximize* any of the following objectives to determine a matching.

$$z_{\text{bval}} = \sum_{(k,j) \in E} (\eta(\beta_k) - \zeta_j(\beta_k)) x_{kj} \tag{16}$$
$$z_{\text{util}} = \min_{(k,j) \in E} 1 - \frac{v_k^\beta}{v_j^\gamma} x_{kj} \tag{17}$$
$$z_{\text{ndem}} = \sum_{(k,j) \in E} |Q_k| x_{kj} \tag{18}$$

The objective function z_{bval} maximizes the value derived from matching the bundle β_k with γ_j, which is expressed as the difference in the market cost $\eta(\beta_k)$ (without bundling) and the actual shipping cost $\zeta_j(\beta_k)$ (after bundling). Under-utilization of truck j with respect to (k, j) can be measured as $1 - \frac{v_k^\beta}{v_j^\gamma}$. Maximizing z_{util} (17) finds a matching that maximizes the minimal under-utilization of trucks (*maxmin* objective). The third objective z_{ndem} (18) maximizes the total number of demands shipped. Objective functions z_{bval}, z_{util}, and z_{ndem} are egalitarian measures of total bundle value, utilization, and the number of shipped demands of the matching solution.

The marketplace can define many such objectives taking into account business value, fairness, and incentives. For example, prioritizing trucks that are unmatched for longer duration, and distributing (or restricting) the business share across certain carrier organizations, can be modeled as objectives (or included as constraints). Marketplace can work with a set of objectives, and dynamically choose the relative importance of the objectives depending on the business needs. Let us assume that the marketplace works with the three objectives (16) - (18). The simplest and a feasible way is to combine these multiple objectives into a single objective function using weights or utilities [5]. The *weighted multi-objective matching* (WMOM) problem is given by:

$$\begin{aligned}
&\text{WMOM:} \\
&\qquad \textit{max } \mu_1 z_{\text{bval}} + \mu_2 z_{\text{util}} + \mu_3 z_{\text{ndem}} \qquad (19) \\
&\text{subject to} \\
&\qquad (13) - (15) \\
&\qquad x_{kj} = \{0, 1\}, \forall (k, j) \in E
\end{aligned}$$

The weight vector $\mu = (\mu_1, \mu_2, \mu_3)$ is non-negative and capture the relative importance of the objective as required, by the marketplace. The above linear binary integer program can be solved using standard optimization solvers and the matching $\omega_\mu = \{(k, j) : x_{kj} = 1\}$ derived from its solution dictates which bundle is matched to which truck. Note that the auxiliary linear decision variable and its constraints to model the *maxmin* z_{util} are omitted for brevity and the resulting mixed integer program can be solved by off-the-shelf optimization solvers.

F. Trust and Transparency Issues

Given that all the crucial elements are orchestrated by the marketplace, there are obvious trust and transparency issues. The marketplace can choose the matching by adding business constraints or systematically influence the objectives to favor certain business partners. As the shippers and carriers have no visibility and control over how the matching is done, such trust issues are common in orchestrated collaboration. For example, cab aggregators are being commonly accused of prioritizing business to cabs financed by partners. To alleviate the trust and transparency issues, we propose a Blockchain based collaborative shipping marketplace, where the partners have stake in choosing the matching solution.

III. COLLABORATIVE SHIPPING *sans* ORCHESTRATOR

In order to build a Blockchain based system, we first propose a *collaborative decision making mechanism*, to replace the centralized matching by the orchestrator. Figure 2(a) outlines the steps in determining the matching in a centralized fashion by an orchestrator. Given the demand and supply profiles ($\{\alpha_i\}$ and $\{\gamma_j\}$), step (A1) is the construction of the DBS (*demand bundle supply*) graph $(\{\beta_k\}, \{\gamma_j\}, E)$. The construction of the DBS graph is achieved through 1) determining the demand bundles $\{\beta_k\}$ (Figure 1), 2) cost allocation $\phi(k, j) = (c_i^{kj})$ for all (k, j) pairs, and 3) determining the edges E using the matching feasibility conditions (6) - (12). In step (A2), the integer programming formulation WMOM (19) is solved with preference vector μ to obtain the final matching ω_μ.

The equivalent collaborative decision making process without orchestrator is outlined in Figure 2(b). The construction of the DBS graph (step (B1)) does not require collaboration from stake-holders. It is a deterministic process that can be implemented as a smart contract in Blockchain (to be explained in Section V-B). Steps (B2) - (B5) illustrate how the *centralized matching* by the orchestrator is replaced with a *collaborative matching* by the stake holders.

The three egalitarian objectives z_{bval}, z_{util}, z_{ndem} ((16) - (18)) make the matching non-trivial and a multiobjective problem. The objectives generally are not aligned and can be conflicting while optimized individually. There are three ways to solve multiobjective problems [6]: 1) scalarization technique that combines all the three objectives using the weights provided by the decision maker, 2) interactive technique that alternates between finding solutions and soliciting feedback from the decision maker, and 3) Pareto method that first generates all the *efficient* (or *non-dominated*) solutions, followed by the decision maker choosing the best suited solution. A solution is efficient or non-dominated if it cannot be improved with respect to any objective function, without reducing the value of at least one of the remaining objectives.

In the centralized matching, orchestrator uses the scalarization technique with weights μ to solve the WMOM (19) problem. Scalarization assumes that the weights μ,

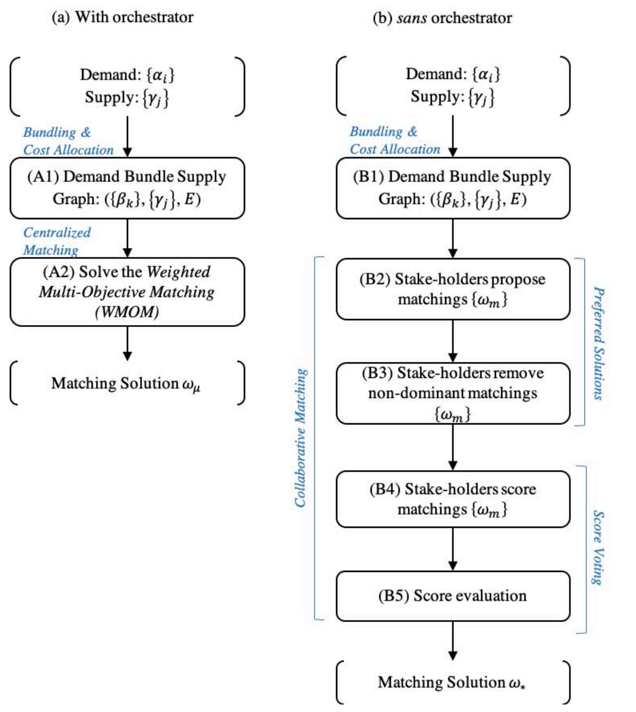
(a) With orchestrator
(b) *sans* orchestrator
Demand: $\{\alpha_i\}$
Supply: $\{\gamma_j\}$
Bundling & Cost Allocation
(A1) Demand Bundle Supply Graph: $(\{\beta_k\}, \{\gamma_j\}, E)$
Centralized Matching
(A2) Solve the *Weighted Multi-Objective Matching (WMOM)*
Matching Solution ω_μ
(B1) Demand Bundle Supply Graph: $(\{\beta_k\}, \{\gamma_j\}, E)$
(B2) Stake-holders propose matchings $\{\omega_m\}$
(B3) Stake-holders remove non-dominant matchings $\{\omega_m\}$
(B4) Stake-holders score matchings $\{\omega_m\}$
(B5) Score evaluation
Matching Solution ω_*
Collaborative Matching
Preferred Solutions
Score Voting

Figure 2. Collaborative shipping with and without orchestrator

which capture the preferences (relative importance of the objectives) of the decision maker, are known and fixed *a priori*. It is a useful and commonly used technique for centralized decision making, but there are few shortcomings that pose serious challenges with multiple stake-holders. A small change in weights may result in large changes in the solution [7]. For example, few percent change in the weights could alter the solution drastically and majority of the stake-holders might find the new solution favorable. Thus by fixing the weights, stake-holders will not be able to explore and suggest the alternatives. It is also possible for the optimal solution to be not acceptable, as inappropriate setting of the weights might exclude aspects of the problem which were unknown prior to optimization [8]. Also, fixing the weights *a priori* precludes the possibility of examining several candidate solutions and choosing the most preferred solution. Due to the above shortcomings of scalarization technique with respect to multiple stake-holders, we propose the use of Pareto method, which is *a posteriori* technique, consisting of two steps: 1) enumerate all or a *preferred* subset of the efficient solutions, and 2) choose the *best* of the efficient candidate solutions.

Using the Pareto method for collaboratively choosing the matching poses new challenges, as shown in Table III. The objectives ((16) - (18)) with matching constraints ((13) - (15)) is a *multiobjective combinatorial optimization* problem. Computational complexity of enumerating all the efficient solutions is usually hard as there are exponential

Table III
APPROXIMATING THE PARETO METHOD FOR COLLABORATIVE MATCHING

Pareto Method	Challenges for collaborative matching	Approximate resolution
1) Enumerate efficient solutions	Exponential number of solutions	Stake-holders propose preferred solutions
2) Choose the best	Multiple decision makers	Score voting

number of such non-dominated solutions [9]. However, it can be resolved by enumerating only the preferred solutions rather than all the solutions. Stake-holders propose multiple preferred efficient solutions (step (B2) in Figure 2) to create a pool of candidate solutions to be examined. Multiple efficient solutions can be generated by solving a series of scalarization problems with different combinations of weights or by ϵ-constraint techniques [9].

Step (B3) is a validation phase, where solutions submitted in step (B2) can be proved as inefficient (dominated) by other stake-holders. This step is essential to disincentivize stake-holders from submitting poor (dominated) solutions in step (B2). As a decentralized and untrusted system, it is imperative to mitigate unintended and malicious behavior of the stake-holders populating the candidate pool with dominated solutions.

The second step of the Pareto method is to choose the best solution from the candidate pool, which is again challenging with multiple decision makers. We resolve this by *score or range voting* [2,3,4]. Score voting is an electoral system where stake-holders vote each candidate solution with a score (say 0-9 Likert scale) and the solution with highest score (added or averaged) is chosen as the final solution. Ties can be broken by choosing a highest score solution randomly. Steps (B4) and (B5) in Figure 2 capture the application of score voting electoral system to choose the best solution.

It is worth noting that the collaborative matching in steps (B2) - (B5) enable the stake-holders to align their personal objectives with the global egalitarian objectives ((16) - (18)). Step (B2) allow stake-holders to submit multiple preferred efficient solutions. Once the candidate pool is validated in (B3), stake-holders score all the solutions, including those submitted by others. The solution with the highest score captures the majority preference for a solution. All the above steps are transparent with participation from all the stake-holders. In the remainder of the paper, we briefly introduce the Blockchain system and illustrate the implementation the collaborative shipping marketplace using Blockchain.

[2]http://rangevoting.org/
[3]https://electology.org/score-voting
[4]https://electology.org/score-voting

IV. BLOCKCHAIN

Blockchain system is a network of *peers* or *nodes*, where each of the peers hosts a global, distributed, append-only database. The database operations are admitted based on the *consensus* of the peers, to ensure that a single global state of truth is maintained across all peers. Thus Blockhain systems are *logically* centralized but *organizationally* decentralized, without a centralized administrator. In addition to hosting the database, the peers can execute arbitrary, programmable transaction logic, in the form of *smart contracts*. The peers execute a *consensus* protocol to validate transactions, group them into blocks, and build a hash chain over the blocks.

Blockchain became popular with Bitcoin[5] and now seen as a technology enabler for trusted exchanges in the digital domain (beyond cryptocurrencies). Bitcoin runs on *public* or *permissionless* Blockchain where anyone can participate without a specific identity. Public Blockchains typically involve a native cryptocurrency and often use computational consensus protocols with economic incentives. Our focus is on *permissioned* Blockchain, which connects a set of known, identified participants. A permissioned Blockchain provides a platform to secure the interactions among a group of stake-holders that have a common goal but who do not fully trust each other. By relying on the identities of the peers, a permissioned Blockchain can use any of the traditional consensus algorithms like Byzantine fault tolerance. Note that the consensus algorithm is for validating and ordering transactions in the Blockchain and is independent of the smart contracts that implement the application or business logic. Hence, the choice of consensus mechanism is not relevant to the scope of this work.

For the reminder of the discussion, we will use the Hyperledger Fabric[6] as the reference Blockchain platform. The ledger in Fabric consists of *transaction logs* and *world state* (application data). For example in the collaborative shipping marketplace, shippers and carriers submit the demand and supply to the Blockchain through *transactions*, while the application data comprising of $\{\alpha_i\}$, $\{\gamma_j\}$ are *world state*. Smart contract environment in Fabric is generically called as *chaincode*. All the ledger operations are coded as functions (procedures) in chaincode and the agents invoke the appropriate functions, resulting in a transaction. Chaincode can also contain functions that get executed based on the world state (*smart* contracts in the traditional sense).

Existing Blockchain powered supply chain applications are focused primarily on: (i) *visibility*; (ii) *choreography* of collaborative business process execution [10] with applications in supplier financing, invoicing, and cross-border logistics (TradeLens[7]); and (iii) *provenance*[8], including di-

[5]http://bitcoin.org/
[6]https://www.hyperledger.org/projects/fabric
[7]https://www.tradelens.com
[8]https://www.provenance.org/technology

amonds[9] and food[10]. Our work introduces a new practical application of Blockchain for *collaborative decision making*. The nodes owned by the stake-holders have stakes in choosing the bundling opportunities for the shippers and matching with the carriers. Though *visibility* and *choreography* are highly relevant to digital freight marketplaces, business process execution is beyond the scope of this paper.

V. Collaborative Shipping using Blockchain

A. Architecture

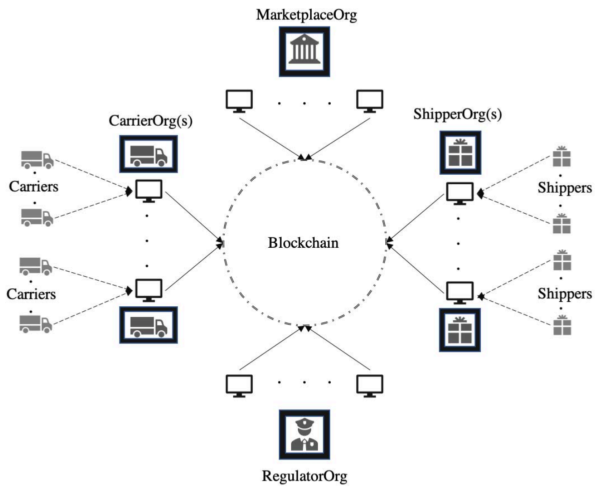

Figure 3. Architecture of Blockchain driven marketplace

Blockchain system is a network of *nodes* or *peers* that host the application logic, distributed ledger, and other distributed system protocols. Using the Fabric terminology, nodes are owned by *organizations* (stake-holders). An organization can own multiple nodes (for scalability) and a node is associated with only one organization. Figure 3 shows the proposed architecture with four types of organizations: 1) `MarketplaceOrg` to choreograph the various business processes; 2) `CarrierOrg` that represents a set of carriers; 3) `ShipperOrg` that represents a set of shippers; and 4) `RegulatorOrg` representing regulatory bodies to monitor, audit, and resolve conflicts. There can be other organizations relating to financial institutions and third party IoT services that are not included for brevity.

There can be multiple `CarrierOrg` and `ShipperOrg`, catering to different set of carriers and shippers, respectively. `CarrierOrg` and `ShipperOrg` can be associations or cooperatives or even third party services. Each of the organizations admit registered *users*, who can interact with the Blockchain using the nodes owned by the organization. Individual shippers and carriers are users who are admitted by their respective organizations. Spot shipping markets is populated with small scale shippers and carriers and it is not economically viable for each of them to own a node. Organizations usually provide layered privacy and access control to the users.

[9]https://diamonds.everledger.io/
[10]https://www.ibm.com/in-en/blockchain/solutions/food-trust

B. Transactions and Smart Contracts

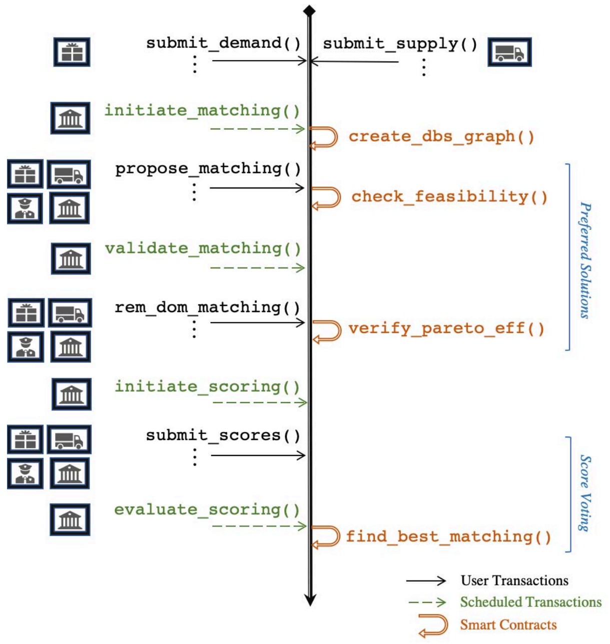

Figure 4. Transactions and Smart Contracts

For illustrative purpose, we categorize chaincode as *transactions* and *smart contracts*. Transactions are procedures that are invoked by the users or organization, whereas smart contracts are procedures that are in-turn invoked by transactions. Figure 4 shows the sequence of transactions and smart contracts that implement the steps in Figure 2(b). Individual shippers and carriers submit α_i and γ_j to the Blockchain using transactions `submit_demand()` and `submit_supply()`, through their respective organizations. `MarketplaceOrg` initiates the bundling and matching using the *scheduled transaction* `initiate_matching()`. Scheduled transactions are triggered by *cron* jobs running at the `MarketplaceOrg` to delineate different phases of bundling, solutions proposal, and scoring.

Transaction `initiate_matching()` invokes the smart contract `create_dbs_graph()` to create the DBS graph. As explained in Section III, this is a deterministic operation and is executed as a chaincode. Even though this operation takes exponential time in worst case (all possible combinations of demands as bundles and cost allocation for each of the possible (k, j) pair), it is preferred to be executed as smart contract to ensure that all stake-holders have one single well defined DBS graph to generate efficient matching solutions. Once the DBS graph is generated, the users can read it from the Blockchain *world state* to their local systems using a read transaction (not shown in the figure). Using the DBS graph, users generate matching solutions locally, and submit the preferred solutions to the Blockchain using transaction `propose_matching()`. This transaction invokes the smart contract `check_feasibility()` to check if the solutions are feasible with respect to the constraints (13) - (15). Transactions containing infeasible solutions are rejected.

In addition to feasibility, smart contracts can also check for efficiency. However, checking whether a solution is efficient is in itself requires solving an optimization problem that finds an efficient solution [6]. To avoid such computational burden on the chaincode, we implement this by enabling the stake-holders to submit proof on dominated or inefficient solutions (step (B3) in Figure 2). The scheduled transaction `validate_matching()` ends the solutions proposal phase and initiates the validation step. Users submit proofs on inefficient solutions by providing a dominating solution using `rem_dom_matching()`. Smart contract `verify_pareto_eff()` verifies and removes dominated solution from the candidate pool of efficient solutions.

Score voting procedure is initiated by scheduled transaction `initiate_scoring()`. Users submit scores for all the candidate solutions using `submit_scores()`. Scheduled transaction `evaluate_scoring()` triggers the smart contract `find_best_matching()`, which aggregates the scores and chooses the matching solution with highest score. Random resolution in case of ties can be implemented using pseudo random procedures that will ensure that all nodes break the tie with the same solution to achieve consensus.

Figure 4 illustrates only the important chaincode procedures at a very high level. Evident but important procedures for process execution are intentionally left out. For example, score voting would require far more procedures to ensure that scores are secured during the score phase so that other stake-holders do not alter their scores. The standard CRUD (create, read, update, delete) and access control procedures that usually make up majority of chaincode have been left out. Also to fortify against infinite loops and errors, and disincentivize malicious behavior, one would need judiciously selected protocols for rating of users/nodes and bans on users/nodes in scoring and solution proposals. All of the above are standard procedures and outside the scope of this work.

As a collaborative decision making technique, the application naturally involves intervention from the users for solution proposal, checking for dominated solutions proposed by others, and scoring. The majority of computational burden is thus shifted to the users, which can be practically infeasible to implement without automation. Indeed all of the above can be executed programmatically with minimal manual intervention. Multiple efficient solutions can be generated by solving a series of single optimization problems using ϵ-constraint method by setting grid values for objectives [9] that align with the self-interests of the stake-holders. Scoring of all candidate solutions can also be automated by creating set of scoring rules that represent the self-interests of the stake-holders.

VI. Conclusions

In this paper, we designed a collaborative shipping marketplace for spot shipments *sans* orchestrator using Blockchain. The main contribution of the paper is the first-of-a-kind implementation of *collaborative decision making* on Blockchain. The proposed eclectic approach using techniques from multi criteria optimization, cooperative game theory, and electoral voting, can be extended in diverse ways to orchestrator driven applications in other domains.

References

[1] T. Liu and C. Zhao, "Impacts of freight consolidation and truck sharing on freight mobility," Transportation Consortium of South-Central States, Tech. Rep., 2010.

[2] S. Creemers, G. Woumans, R. Boute, and J. Beliën, "Tri-vizor uses an efficient algorithm to identify collaborative shipping opportunities," *INFORMS Journal on Applied Analytics*, vol. 47, no. 3, pp. 244–259, 2017.

[3] S. H. Tijs and T. S. H. Driessen, "Game theory and cost allocation problems," *Management Science*, vol. 32, no. 8, pp. 1015–1028, 1986.

[4] C. Vanovermeire, D. Vercruysse, and K. Sorensen, "Analysis of different cost allocation methods in a collaborative transport setting," University of Antwerp, Faculty of Business and Economics, Working Papers, Apr. 2013.

[5] J. S. Dyer, *Maut — Multiattribute Utility Theory*. New York, NY: Springer New York, 2005, pp. 265–292.

[6] M. Ehrgott, "Multiobjective optimization," *AI Magazine*, vol. 29, no. 4, pp. 47–57, 2008.

[7] S. A. Khan and S. Rehman, "Iterative non-deterministic algorithms in on-shore wind farm design: A brief survey," *Renewable and Sustainable Energy Reviews*, vol. 19, pp. 370 – 384, 2013.

[8] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computing*, vol. 3, no. 1, p. 1–16, 1995.

[9] M. Ehrgott, *Multicriteria Optimization*. Springer, 2005.

[10] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted business process monitoring and execution using blockchain," in *Business Process Management*, M. L. Rosa, P. Loos, and O. Pastor, Eds. Cham: Springer International Publishing, 2016, pp. 329–347.