Buying.com: a Decentralized e-Commerce Network

Kyriacos Pavlou, Sofoclis Zambirinis, Theo Mourouzis and Raghu Bala

October 16, 2018

Buying.com is a decentralized, e-Commerce Network with cybermediation platform features, allowing businesses and individuals access to products at wholesale prices. Furthermore, Buying.com leverages existing distributors along with its own microdistribution network in order to achieve two-hour threshold delivery time for fast-moving consumer goods. The current paper gives an overview of the Buying.com and its unique value proposition and expounds on the technical aspects behind the platform. Specifically, it covers the Stellar blockchain technology used by Buying.com and its Stellar Consensus Protocol (Federated Byzantine Agreement). It provides a solution to the problem of multi-objective optimization of last-mile logistics and discusses several aspects of platform governance as well as the cryptoeconomics of the ecosystem.

Contents

1 Introduction

E-commerce sales in the past few years has accelerated tremendously with reports that the global B2B e-commerce market was worth $7.7 trillion in 2017 (1). There are some 1.3 million e-commerce companies in the U.S. alone. Worldwide, that figure rises to 2-3 million e-commerce companies, excluding China. The opportunity to convert more global businesses into new age e-merchants is substantial.

The current ecosystem for e-commerce has evolved from the mid-90s during the first wave of internet technologies, with most major brands and retailers maintaining a web and mobile e-commerce presence. Several dominant players such as Amazon.com, Walmart.com and others have emerged for mass market retailing of a wide variety of products. If one were to deconstruct the current state of the e-commerce ecosystem, it would involve several operational components: Storefronts (Shopify), Payment Processing (Paypal, Stripe), Shipping and Fulfillment (Fedex, UPS), Backoffice processing (Salesforce).

Despite the history and the growth of this industry, *inefficiencies persist throughout the ecosystem*:

- **Last Mile Distribution:** This is a problem not only for small retailers but big retailers as well. Amazon has recently purchased Toys R Us and Whole Foods to some extent to expand their distribution channels. The recent introduction of Amazon Flex (2) where people can sign up to deliver goods for Amazon aims to address this but even then last mile peer-to-peer microdistribution is not yet a reality.

- **Purchasing Power:** While more consumers go online to purchase goods and services, there is no efficient mechanism to consolidate their purchases to derive wholesale pricing.

- **Fragmentation:** Smaller retailers do not have access to a vertically integrated service platform to pose a serious threat to larger retailers such as Amazon. This results in their having to use a fragmented set of operational components.

2 Buying.com: The Solution to e-commerce Inefficiencies

Buying.com is a company that aims to reshape the landscape of e-commerce by providing solutions to the inefficiencies of the ecosystem thus benefiting both the supply and demand sides of the equation. *Distributed ledger technology* (blockchain) provides Buying.com with new asymmetric advantages that enables it to apply solutions which were not available in earlier-generation platforms. The *unique value proposition* of the platform comprises several innovative solutions.

- **Microdistribution:** With Buying.com e-commerce becomes fully decentralized by turning every garage into a distribution center. In addition to distribution centers run by Buying.com, now anyone can become a node on the Buying.com network by staking tokens (buy-in) and leveraging existing yet *latent storage capacity* in their garage to become a microdistribution center. This solves the last mile logistics issue that has plagued e-commerce for the longest time. It is similar to how Airbnb turns every spare room in a home to a hotel room, or how Uber leverages excess driving capacity to turn every automobile into a taxi. *Fast moving consumer goods* (FMCG) ordered on the Buying.com app can be delivered to consumers within two hours. This empowers all e-commerce and retail players to not only compete, but in many cases outperform established giants in the industry that have distinct advantages at the moment. A fully decentralized solution addressing delivery inefficiencies will be disruptive and transformative for the e-commerce sector.

- **Bulk Pricing:** Consumers and businesses would like to enjoy large purchase pricing discounts but that is not always possible because *minimum order quantities* (MOQ) needed for bulk pricing are not achievable. The Buying.com protocol leverages the purchasing power of millions of consumers and small businesses to reach MOQ levels. This enables online stores and consumers to receive direct from manufacturer pricing through bulk order quotes with best possible pricing on products.

- **Real-Time Logistics:** Leverages blockchain solutions to deliver real-time logistics data, provides transparent, smart-contract enforced audit trails, and protects users with data encryption. Manufacturers, businesses and consumers will all have seamless access to real-time shipping data to optimize dropshipping costs provided by a permissionless, public blockchain.

- **BUY token:** The Buying.com protocol utilizes its own cryptocurrency, named BUY, for transactions on the Buying.com platform.

- **Transaction Payments:** Buying.com's protocol transactions will escrow payments until customer receives the ordered goods and is satisfied. Once the delivery is cleared, funds are released. This will significantly reduce *chargebacks*, a commercial problem costing retailers billions every year.

- **Transparent and Auditable:** Blockchain technology offers customers a fraud-proof digital ledger of verifiable transactions and hence a transparent, immutable audit trail.

The current paper focuses on the technical aspects of the Buying.com platform and how these affect the overall ecosystem. Section 3 covers the blockchain technology used by Buying.com, namely, the Stellar platform. The next section (Section 4) gives a detailed analysis and proposed solutions to the problem of multi-objective optimization of last-mile logistics. Section 5 deals with the platform governance (i.e., consensus algorithm, Ricardian contracts, proof-of-delivery mechanisms) while the subsequent section discusses network scalability and attendant effects (Section 6). The paper concludes with a presentation of the cryptoeconomics of the ecosystem.

3 The Buying.com Platform and Protocol

3.1 Decentralized Network Architectures

Traditional Client-Server architectures are currently being supplanted by decentralized computing platforms wherein ecosystems of organizations and individuals participate and share data. A decentralized network consists of peers that can run independently of each other. The power to transmit information is distributed among a network of servers, instead of being driven from one primary source. For this reason, data silos are a thing of the past in decentralized computing, and this is made possible through the use of Distributed Ledgers.

3.2 Distributed Ledger Technology

Distributed Ledgers can be developed using a number of different technologies including blockchain (e.g., Ethereum Hyperledger) and directed acyclic graphs (e.g. IPFS, IOTA), and there can be both permissioned and public distributed ledgers. Layered on top of distributed ledgers, depending on the application at hand, is the concept of cryptocurrency. This is particularly applicable in public distributed ledgers such as Ethereum, Bitcoin, IOTA, and others. A distributed ledger is a database that is consensually shared and synchronized across the network and spread across multiple sites, institutions or geographies. It allows transactions to be immutable and publicly verifiable thereby making a cyberattack more difficult.

Another useful concept in decentralized computing is that of *smart contracts*. A smart contract can be thought of as code that acts upon data stored in a distributed ledger, very similar to how stored procedures and triggers in traditional database technology operate on data stored in tables. So, taken in aggregate, a decentralized network could be thought of as a combination of network, logic, and data.

STELLAR VS. ETHEREUM

	Stellar	Ethereum
Average Verification Time	5 seconds	3.5 minutes
Price	Minor commission for the transaction (.00001 XLM ~ = $ 0.0000002). 10 XLM / deposit offer (returned when the offer is completed or canceled). There is no payment for gas for settlements.	Depends on the complexity of the calculations, the transaction speed and the fixed cost of the Ethereum. The average cost of transaction is 0.094 US dollars.
Features	A library of basic abstractions that can generate complex behavior. See the Stellar Developer's Guide.	All that the mind can imagine, and many things that he was not considered.
Security	Decentralized network: everyone can run the Stellar Core node and check transactions. You can choose validators to improve security.	Decentralized network: everyone can start a node and check transactions. There is no built-in function for selecting approved validators.
	Atomic transactions, consisting of simple declarative operations, lead to an increase in the number of code being scanned and fewer security errors.	The full programming capabilities of Turing allow you to get less testable code and increase the risk of vulnerable vulnerabilities.

Figure 1: Comparison of Stellar to Ethereum. Reproduced from https://www.stellar.org.

3.3 Stellar Network

A public blockchain infrastructure provides an ideal solution to address the problems outlined in the previous section and implement the unique value proposition of Buying.com in a trustless environment. The Buying.com Protocol is based on the Stellar blockchain network (3).

The Stellar decentralized ledger records a list of all the balances and transactions belonging to every single account on the network. A complete copy of the global Stellar ledger is hosted on each server that runs the Stellar software. Any entity can run a Stellar server.

Figure 1 compares Stellar to Ethereum in terms of average verificaton time, price per trans-

action, features, and security.

The Stellar servers communicate and sync with each other to ensure that transactions are valid and get applied successfully to the global ledger. This entire process of coming to consensus on the Stellar network occurs approximately every 2-5 seconds and uses the Stellar Consensus Protocol (SCP) which is an implementation of Federated Byzantine Agreement (FBA). The Buying.com protocol uses a modified (SCP). A detailed explanation of the consensus algorithm can be found in Section 5.2.

3.4 Node Operation

Node operators can participate to the network in multiple ways. From an operational point of view "watchers" and "basic validators" are about the same (they both compute an up-to-date version of the ledger). "Archivers" or "Full validators" publish into a history archive which has additional cost. Figure 2 below summarizes the level of participation to the network of the different nodes.

Level of participation to the network

As a node operator you can participate to the network in multiple ways.

	watcher	archiver	basic validator	full validator
description	non-validator	all of watcher + publish to archive	all of watcher + active participation in consensus (submit proposals for the transaction set to include in the next ledger)	basic validator + publish to archive
submits transactions	yes	yes	yes	yes
supports horizon	yes	yes	yes	yes
participates in consensus	no	no	yes	yes
helps other nodes to catch up and join the network	no	yes	no	yes
Increase the resiliency of the network	No	Medium	Low	High

From an operational point of view "watchers" and "basic validators" are about the same (they both compute an up to date version of the ledger). "Archivers" or "Full validators" publish into an history archive which has additional cost.

Figure 2: Types of nodes and their level of participation in the Stellar network. Reproduced from https://www.stellar.org.

It is recommended that all micordistribution nodes, delivery drivers, suppliers, dropshippers and third-party retailers participate in the network as full validators in order increase network resiliency. Consumers can participate as basic validators since they will need to participate in the consensus algorithm.

4 Delivery Optimization

When a customer places an order through Buying.com, the order is assigned to a distribution center where the order will be delivered from, according to the procedure and rules described in the following subsection. When time comes to schedule the delivery from that particular distribution center, and given the set of customers waiting to be served, the driver will be selected and the optimal route will be constructed. In general, deliveries are scheduled at the end of the day for next-day delivery, or every 2 hours for 2-hour delivery. Subsections 4.1, 4.2 and 4.3 describe how the distribution center, the driver and the optimal route are selected, repsectively. Finally, subsection 4.4 describes how to dynamically update existing routes when new order requests arrive, by inserting customers in already planned routes in an optimal way, when this is possible.

Customers that place an order for FMCG will have the option to request either a 24-hour delivery or a 2-hour delivery. Obviously, a 2-hour delivery will come with a higher delivery price. Under the 2-hour delivery guarantee, in the worst case, a driver will be selected to pickup the products from the distribution center and ensure delivery to the customer within 2 hours from the time when the order was placed, without combining any other orders. It is very important that such orders arrive on time, otherwise the reputation of the driver will be negatively impacted. Of course, in case where such a delay occurs but it is the distribution center's fault, then the reputation of the distribution center will be negatively impacted instead. While on a two-hour delivery duty, the driver's route and schedule will not be modified dynamically.

The normal mode of delivery will be the 24-hour delivery guarantee. This applies to all items

that do not fall under the FMCG category, as well as to FMCG for which the customer did not choose the 2-hour delivery option. At the end of each day (or more frequently, if appropriate), for each distribution center, the routes for the following 24 hours are scheduled in an optimal way. These routes can later be modified dynamically during execution, upon new customer requests.

4.1 Delivery Optimization Part I: Distribution Center Node Selection

Let K be the set of all customers. Suppose that customer $k \in K$ has just placed an order through the Buying.com platform for items $1, 2, ..., n$ with associated quantities $q_1, q_2, ..., q_n$. We need to decide which distribution center will be selected as the pickup node for the order, among all available distribution centers in the region that have stock of the requested quantities of all the items ordered.

Let $J = \{1, 2, ..., |J|\}$ be the set of all nodes that represent distribution centers. Let $J^* \subseteq J$ be the subset of distribution centers that: (a) are located within a radius R from customer k, (b) have stock of the necessary quantities for all the items ordered, and (c) are open and available at the date and time of interest. We restrict our choice of the distribution center within the set J^*. Specifically, for each distribution center $j \in J^*$, we evaluate:

1. The distance $d_{j,k}$ from the distribution center j to the location of customer k who placed the order. In general, we would prefer $d_{j,k}$ to be as small as possible. Note that $d_{j,k}$ is bounded by R (i.e. $0 \leq d_{j,k} \leq R$). We wish to minimize $\frac{d(j,k)}{R}$, which is a real number in the interval $[0, 1]$.

2. The *distribution center's reputation score* R_j, which is a real number in the interval $[0, 1]$. The value 0 represents the worst reputation score of 0%, whereas the value 1 represents the best reputation score of 100%. Please refer to Section 5.4 for more details on how R_j is constructed. In general, we would prefer a distribution center with a higher rating. Therefore, we wish to maximize R_j, or equivalently to minimize $-R_j$.

3. The time ψ_j that has elapsed since the last time an order was fulfilled by distribution center j, where $\psi_j \geq 0$ for all $j \in J^*$. In general, a distribution center with a higher value of ψ_j will be preferred. Incorporating ψ_j in the objective function will give a degree of fairness in the process of selecting the distribution center. Specifically, distribution centers that haven't been selected as pickup nodes recently, will be favored. This metric will help to avoid ending up with a few centers that managed to get high ratings initially, getting most of the orders all the time, while all other centers get very few orders and eventually opt-out of the agreement. In order to normalize the respective component objective, we need to define $\psi_{max} = \max\{\psi_j | j \in J^*\}$. We wish to maximize $\frac{\psi_j}{\psi_{max}}$, where $0 \leq \frac{\psi_j}{\psi_{max}} \leq 1$, or equivalently to minimize $-\frac{\psi_j}{\psi_{max}}$.

4. Assume that for each item i that is sold through the Buying.com platform and has an expiration deadline, we have identified a threshold date, after which the item is considered to *expire soon*. Let $q^E_{i,j}$ be the number of pieces of item i in distribution center j that expire soon. Then, $min\{q^E_{i,j}, q_i\}$ is the number of pieces of item i in distribution center j that expire soon and will be included in the order, in case the order will be fulfilled by distribution center j. In general, we would prefer selecting distribution center j_1 over distribution center j_2, if the first one has more items that expire soon, compared to the second one. Therefore, we wish to select the distribution center j which minimizes the term $-\frac{\sum_{i=1}^{n} min\{q^E_{i,j}, q_i\}}{\sum_{i=1}^{n} q_i}$, which is equivalent to maximizing the term $\frac{\sum_{i=1}^{n} min\{q^E_{i,j}, q_i\}}{\sum_{i=1}^{n} q_i}$. The latter is a real number in the interval $[0, 1]$ and denotes the proportion of the number of items in the order which expire soon, if the order is fulfilled by the distribution center j. It is a measure of urgency to deliver from distribution center j, due to potential expiration of its stock that is related to the order. Note that, since $0 \leq min\{q^E_{i,j}, q_i\} \leq q_i$, we will always have $0 \leq \sum_{i=1}^{n} min\{q^E_{i,j}, q_i\} \leq \sum_{i=1}^{n} q_i$, which leads us to the inequality $0 \leq \frac{\sum_{i=1}^{n} min\{q^E_{i,j}, q_i\}}{\sum_{i=1}^{n} q_i} \leq 1$.

5. Let M_{total} be the order's total worth, in tokens. Define M_j as the number of tokens missing from the distribution center j to cover the order's total worth M_{total}, if the number of tokens that it holds is less than M_{total}; or 0 otherwise (i.e. $M_j = 0$ if the distribution center j holds at least as many tokens as M_{total}). We wish to minimize $\frac{M_j}{M_{total}}$. Note that since $0 \leq M_j \leq M_{total}$, we get $0 \leq \frac{M_j}{M_{total}} \leq 1$. By incorporating the term $\frac{M_j}{M_{total}}$ in the objective, we allow the possibility of selecting a distribution center that owns fewer tokens than the amount of tokens equivalent to the value of the stock. However, in such a case, the greater the deficiency in tokens, the less likely it is for a distribution center to be selected. On the other hand, if we wish to restrict our choice of the distribution center among the ones which hold at least as many tokens as the order's total worth, then we can consider only the distribution centers with $M_j = 0$, and ignore the last component of the objective function (or equivalently, set $p_5 = 0$, where p_5 is defined below).

Combining all the factors mentioned above, we can calculate the *distribution center's objective function* $F(j)$ separately for each $j \in J^*$, in order to find the distribution center which minimizes $F(j)$, as defined below:

$$\min_{j \in J^*} \; F(j) := p_1 \frac{d_{j,k}}{R} - p_2 R_j - p_3 \frac{\psi_j}{\psi_{max}} - \; p_4 \frac{\sum_{i=1}^{n} \min\{q_{i,j}^E, q_i\}}{\sum_{i=1}^{n} q_i} + \; p_5 \frac{M_j}{M_{total}} \tag{1}$$

with weights $p_\nu \in [0, 1]$ for $\nu = 1, 2, ..., 5$ such that $\sum_{\nu=1}^{5} p_\nu = 1$. The importance of each component objective can be captured by assigning appropriate values to the weights p_ν. For example, if the first component objective is more important than the fourth one, then we can assign a higher value to p_1 than to p_4.

The objective function for the optimal selection of distribution center presented above, is the weighted sum of five component objectives, that seeks to find the distribution center j which simultaneously: (i) minimizes the distance of the distribution center from customer k who placed the order, (ii) maximizes the reputation of center j, (iii) maximizes the elapsed time since the last time an order was fulfilled by distribution center j, (iv) maximizes the measure of urgency

to deliver from distribution center j due to expiration of the stock of distribution center j which is related to the order, and (v) minimizes the missing tokens from distribution center j. Additionally, each one of the five component objectives is normalized.

For each order placed through the platform, we calculate the above objective function separately for each distribution center j in the area which has the goods in stock, and the single distribution center j^* which yields the minimum value of the above function is selected.

4.2 Delivery Optimization Part II: Driver Node Selection

After an order has been placed and the distribution center has been selected as the pickup node, we need to select the driver that will pick up the order from the distribution center and carry out the delivery. In short, for the driver selection, we consider all available drivers within a radius r from the distribution center, and for each one of them, we calculate the driver's objective function, which is based on three factors: (i) the distance between the current location of the driver and the distribution center, (ii) the reputation of the driver, and (iii) the last time the driver was chosen for a delivery. The top x drivers with respect to the objective function are notified, and the first one to respond gets the order. If multiple responses come within a narrow window of y seconds, preference is shown to the one with the best value of the objective function.

More specifically, let $L = \{1, 2, ..., |L|\}$ be the set of all nodes that represent drivers. We assume that each driver $l \in L$ comes with a specific vehicle of known capacity C_l. The value C_l is by default the total capacity of vehicle l, unless the vehicle is partially loaded, in which case it is the driver's responsibility to update his vehicle's capacity, so that the updated C_l value now reflects the available or remaining capacity. Note that, in the extreme case when a driver accepts an order while his vehicle is partially loaded and, upon arrival at the pickup location, the order does not fit in the vehicle because the driver failed to report the true available capacity of his vehicle, this is considered to be a violation of the agreement between the driver and the company, by the former party. In this case, the driver will be penalized and his reputation will

be negatively affected.

Let $L^* \subseteq L$ be the subset of the drivers that: (a) are within a radius r from the distribution center at the particular time, (b) are available for work at the time, (c) have vehicles with enough capacity that can accommodate the order and (d) have vehicles with the necessary equipment needed to load and unload the orders (in case where such an equipment is needed). For each one of those drivers $l \in L^*$, we calculate the following:

1. The distance $d(l, j)$ from current location of driver l to the distribution center j where the order will be picked-up from. In general, we prefer $d(l, j)$ to be as small as possible. Therefore, we wish to minimize $d(l, j)$, or equivalently to minimize $\frac{d(l,j)}{r}$, where $0 \leq \frac{d(l,j)}{r} \leq 1$.

2. The *reputation* r_l *of driver* l, which is a real number in the interval $[0, 1]$. The value 0 represents the worst reputation score of 0%, whereas the value 1 represents the best reputation score of 100%. Please refer to Section 5.4 for more details on how r_l is constructed. In general, we would prefer a driver with a higher rating. Therefore, we wish to maximize r_l, or equivalently to minimize $-r_l$.

3. The time $t_l \geq 0$ that has elapsed since the last time an order was fulfilled by driver l (counting from the time that the previous route of driver l finished). In general, a driver with a higher t_l will be preferred. Let $t_{max} = \max\{t_l | l \in L^*\}$. We wish to maximize $\frac{t_l}{t_{max}}$, where $0 \leq \frac{t_l}{t_{max}} \leq 1$, or equivalently to minimize $-\frac{t_l}{t_{max}}$. Incorporating this factor in the objective function will give a degree of fairness to the selection of drivers, in a sense that it will tend to favor drivers that haven't been selected recently for delivery. The reason for using this metric is to avoid the scenario where the few selected drivers who get the best ratings in their first few orders, end up with being assigned almost all of the deliveries later on, while the remaining drivers 'starve' and eventually opt-out of the agreement.

Taking into account the above factors, the driver's objective function is calculated separately for each driver $l \in L^*$. Specifically, the objective at the driver selection stage is to find $l \in L^*$ that minimizes the *driver's objective function*, $f(l)$, which is defined as follows:

$$\min_{l \in L^*} f(l) := \rho_1 \cdot \frac{d(l,j)}{r} - \rho_2 r_l - \rho_3 \frac{t_l}{t_{max}} \tag{2}$$

where $\sum_{i=1}^{3} \rho_i = 1$ and $\rho_i \in [0,1], \forall\, i = 1,2,3$.

The above objective function is the weighted sum of three normalized component objectives, with weights ρ_i, $i = 1,2,3$, that essentially seeks to find the driver l which simultaneously: (i) minimizes the distance from the distribution center to the driver, (ii) maximizes the driver's score and (iii) maximizes the time that has elapsed since the last time an order was fulfilled by driver l.

Instead of simply selecting the single driver with the minimum value of the objective function $f(l)$, the drivers with the x best (lowest) values of the objective $f(l)$ are notified, and the first one to respond gets the order. However, if multiple responses come within a narrow window of y time units, preference is shown to the one with the best (lowest) value of $f(l)$.

Note that the following information needs to be communicated to the drivers, before they accept a route: time of start of the route, estimated duration of the route (or upper bound), and reward or payment for the driver.

4.3 Delivery Optimization Part III: Route Selection

In this subsection we present a mathematical model for selecting the optimal delivery route, once the distribution center, the customers waiting to be served from the particular distribution center, and the driver have all been selected. Specifically, we present a variation of the Traveling Salesman Problem with Time Windows (TSPTW) formulation, which we have adapted in order to solve the problem of determining the optimal route to be traversed by a specific driver and the schedule of visits to the customers, once the driver has been selected and the customers that

will be served in the respective route have been specified.

In general, the TSPTW is the problem of determining a minimum cost tour in which a set of nodes are visited exactly once within their requested time windows. More specifically, in the *adapted TSPTW* that we propose, we have a set of customers waiting to be served and a single vehicle (traveling salesman) that must depart from a specific starting point, it will then directly visit the distribution center to load the orders, it will visit each of the customers exactly once and finish at a specific end-point. Each customer i is associated with a service time s_i, i.e. the amount of time that the vehicle needs to spend at the customer once the service starts; and a time window $[a_i, b_i]$ defined by its ready time a_i and due date b_i. The time of start of service of any customer must lie within the respective time windows. This means that the time when the service finishes may actually be later than the due date. Furthermore, if the vehicle arrives at a customer earlier than their ready time, it must wait until the ready time. The objective is to minimize the total travel time.

We formulate the *adapted TSPTW* on a graph $G = (I, A)$, where $I = \{0, 1, 2, \ldots, n+2\}$ is the set of nodes and A is the set of arcs, defined below. Node $n+1$ represents the starting point and node 0 represents the end-point. Node $n+2$ represents the distribution center. Any two or even all three of the nodes 0, $n+1$ and $n+2$ may coincide, in case they correspond to the same geographical location. For instance, in the special case when the distribution center uses its own vehicle and driver, the starting point node $n+1$, the end-point node 0, and the distribution center node $n+2$ will all coincide. Let $I_1 = \{1, 2, \ldots, n\}$ be the set of customers. The set of arcs $A \subseteq I \times I$ is defined as $A = \{(n+1, n+2)\} \bigcup \{(n+2, j) : j \in I_1 \cup \{0\}\} \bigcup \{(i, j) : i \in I_1,\ j \in I_1 \cup \{0\}, i \neq j\}$. Also, for each $i \in I$, define $\Delta^+(i) = \{j \in I : (i, j) \in A\}$, and $\Delta^-(i) = \{j \in I : (j, i) \in A\}$.

Let $t_{ij} \geq 0$ be the travel time from node i to node j, for all $(i, j) \in A$. Let $[a_i, b_i]$ and s_i be the time window and service time associated with node i respectively, for all $i \in I$. For the starting point $n+1$ and end-point 0, we have $s_{n+1} = 0$ and $s_0 = 0$. Note that s_{n+2} represents

the amount of time needed from the time when the driver arrives at the distribution center $n+2$, until the time he or she departs from it, which includes the time needed to load the goods.

Let x_{ij} be the number of times that arc (i,j) is traversed, for all $(i,j) \in A$. Let w_i be the time of start of service of customer i, for all $i \in I_1 \bigcup \{n+2\}$. Let w_{n+1} be the time of departure from node $n+1$, and let w_0 be the time of arrival at node 0. Let M be large positive integer (say, $M = 10^{10}$). The quantities n, M, t_{ij}'s, s_i's, a_i's and b_i's are known parameters, whereas x_{ij}'s and w_i's are the decision variables.

The vehicle must start from node $n+1$, which denotes the vehicle's starting position at time 0, and then go directly to the distribution center, which is represented by node $n+2$. The vehicle must then visit all customer nodes in the set I_1, and finish at node 0, which denotes the vehicle's end-point.

The problem is formulated as a mixed-integer linear program (MILP) as follows:

$$\text{minimize} \sum_{(i,j)\in A} t_{ij} x_{ij} \tag{3}$$

subject to:

$$w_{n+1} = 0 \tag{4}$$

$$x_{n+1,n+2} = 1 \tag{5}$$

$$\sum_{j\in\Delta^+(i)} x_{ij} = 1 \qquad \forall\, i \in I \setminus \{0\} \tag{6}$$

$$\sum_{i\in\Delta^-(j)} x_{ij} = 1 \qquad \forall\, j \in I \setminus \{n+1\} \tag{7}$$

$$w_i + s_i + t_{ij} - w_j \leq (1 - x_{ij})M \qquad \forall\ (i,j) \in A \tag{8}$$

$$a_i \leq w_i \leq b_i \qquad \forall\ i \in I \setminus \{n+1\} \tag{9}$$

$$x_{ij} \in \{0,1\} \qquad \forall\ (i,j) \in A \tag{10}$$

$$w_i \geq 0 \qquad \forall\, i \in I \tag{11}$$

The objective (3) is to minimize the total travel time. Equation (4) states that the vehicle must depart from the starting point $n+1$ at zero time. Equation (5) states that it must then visit node $n+2$, which is the distribution center (in order to load the goods to be delivered). Equation (6) ensures that the vehicle leaves each node exactly once, apart from the endpoint node 0. Similarly, equation (7) ensures that the vehicle enters each node exactly once, apart from the starting node $n+1$. Constraints (8)-(9) guarantee the feasibility of the schedule, with respect to time considerations.

Note that there is no check for the capacity constraint in this model. We assume that this happened in a previous stage, and specifically that the total demand of all the customers in the set I_1 does not exceed the vehicle's capacity.

The MILP presented above can be implemented using a mathematical programming solver like CPlex and it can be used to find the exact optimal solution for small instances. However, the TSP and its variants, including the adapted TSPTW presented here, are NP-hard. Therefore, for large instances, appropriate heuristics may need to be developed.

Note: The adapted TSPTW can easily be extended for the case of more than one vehicles, into the *adapted Vehicle Routing Problem with Time Windows (adapted VRPTW)*. A comprehensive study of the Vehicle Routing Problem and its variants, including the classical VRPTW, can be found in Toth & Vigo (2014) (4).

4.4 Delivery Optimization Part IV: Dynamically updating current routes, upon new order requests.

Scheduled routes may be modified dynamically upon new order requests, with the exception of 2-hour delivery routes which are not allowed to be altered once the driver has accepted the original order. In this subsection we will discuss how to dynamically insert a customer in an already scheduled route in an optimal way, for the cases where it is allowed to dynamically alter the routes.

When a request for a new order comes in, the distribution center (DC) where the relevant order will be picked up from will first be selected, according to the rules described in subsection 4.1. The new customer who placed the order may be inserted dynamically in a route which is currently being executed, incorporated in an already scheduled route whose execution has not started yet, or a new route may have to be created for that particular customer. Consider the following scenarios:

1. Suppose that there is already a scheduled route that involves pickup of items from the particular DC, but either the execution of this route has not started yet, or it has started but the vehicle has not left the DC yet. Then, if by adding the customer to that route the total load will exceed the capacity of the vehicle, we need to construct another route for that customer and either use another vehicle, or wait until the first vehicle finishes its route to be used again. In this case, depending on the time windows, we may wait until more orders arrive before we select another driver and schedule a new route, according to the rules of subsections 4.2 and 4.3.

   If on the other hand, by adding the new customer to an already scheduled route whose execution has not started (or it has started but the vehicle has not left the DC yet), the total load does not exceed the capacity of the vehicle, then we add the new customer to the existing set of customers I_1 that were to be served by the particular vehicle, and we re-optimize, by solving the MILP described in section 4.3 with the new set of customers.

In this case, depending on the geographic location of the nodes, the travel times, service times and time windows, the order in which customers are visited in the new route may differ from the original one, but the new route will be optimal with respect to minimizing the total travel time, and it will include the new customer (not necessarily at the end of the route). Of course, it is possible that due to the time windows, the new problem is infeasible; in which case, we need to create another route, as described in the previous paragraph.

2. Suppose that there is no route scheduled for the specific DC. Then obviously we need to select an available driver and construct a new route for that customer. As before, depending on the time windows of that customer, we may or may not be able to wait until more orders arrive before we select a driver and create a new route, as described in subsections 4.2 and 4.3.

3. Suppose that there is at least one scheduled route that involves pickup of items from the selected DC, whose execution has already started and whose respective vehicle has already left the DC. Let $K = \{1, 2, ..., |K|\}$ be the set of all such $|K|$ routes (where $|K| \geq 1$). We wish to select the single route $k^* \in K$ to insert the new customer, and the right position within that route, so that all customers are served within their requested time windows, the capacity constraint is always satisfied and the increase in the total travel time after including the new customer is minimized. In general, the sequence in which customers are visited may completely change after rescheduling.

   - **Strategy A (re-optimize):** For each $k \in K$ separately, we perform the following:

     We add the new customer to the set of customers, remove the served ones, update n as the number of the remaining customers of the route, and relabel the remaining customers as nodes $1, 2, ..., n$, as well as the nodes $n + 1$ and $n + 2$ as the current position of the vehicle and the DC node, respectively. We then solve again the

adapted TSPTW using the model of subsection 4.3, and we get the total travel time $T_1(k) := \sum_{(i,j)\in A} t_{ij}x_{ij}$ of vehicle k, concerning the remaining customers and including the new customer.

We then repeat the process described in the previous paragraph, but without including the new customer. We then get the total travel time $T_0(k) := \sum_{(i,j)\in A} t_{ij}x_{ij}$ of vehicle k, concerning the remaining customers and excluding the new customer.

We then select the single vehicle k^* for which $T_1(k^*) - T_0(k^*) = \min\{T_1(k) - T_0(k)|k \in K\}$, i.e. the one which gives the minimum increase in the travel time after including the extra customer.

- **Strategy B:** For each $k \in K$ separately, we perform the following:

  Find the point on the remaining part of the scheduled route k that is the closest to the DC. Suppose that originally, vehicle k was supposed to pass from this point at time $\theta(k)$. Let n be the number of the remaining customers that were scheduled to be served after $\theta(k)$, increased by one (for the new customer). Denote by $n+2$ the position on the remaining part of the scheduled route k that is the closest to the DC. Then, with $n+2$ as the new starting position and relabeling as $\{1, 2, ..., n\}$ the set of the customers that were scheduled to be served after time $\theta(k)$ extended by including the new customer, solve again the adapted TSPTW of subsection 4.3, to re-optimize the part of the route after time $\theta(k)$, and get the total travel time $T_3(k) := \sum_{(i,j)\in A} t_{ij}x_{ij}$ of vehicle k after time $\theta(k)$, including the new customer.
  Using the same starting node and starting time $\theta(k)$, we repeat the process described in the previous paragraph, but without including the new customer. We then get the

total travel time $T_2(k) := \sum_{(i,j)\in A} t_{ij}x_{ij}$ of vehicle k after time $\theta(k)$, excluding the new customer.

We then select the single vehicle k^* for which $T_3(k^*) - T_2(k^*) = \min\{T_3(k) - T_2(k)|k \in K\}$, i.e. the one which gives the minimum increase in the travel time after including the new customer.

Therefore, by using either strategy A or strategy B, we can select the single route $k^* \in K$ in which the new customer must be inserted. The process described above also provides the way to construct the updated route k^*.

Note that in scenario 3, the proposed strategies are both heuristic ones and may give optimal or sub-optimal solutions. Also, for both strategies A and B, we assume that at the time when re-optimizing starts, the sum of the total load of each vehicle and the demand of the new customer, does not exceed the capacity of the vehicle. Strategy A may work well in cases where the time windows are tight and dramatically affect the routes. For instance, it may be preferred when there is urgency to serve the new customer early. Strategy B may be more appropriate in cases where there is flexibility at the time when the new customer may be served.

5 Platform Governance

In this section issues pertaining to governance, consensus and settlement are covered.

5.1 Self-Governance

Members of the Buying.com platform are integral to the governance and operations of the network. A participant can be considered a member of network if they have a Buying.com account. Such an account can be set up using the Buying.com application. The app and associated account is the same for all members be they consumers, microdistribution node operators or delivery drivers. All members form the core governing community of the platform. The platform

has no default voting processes, campaigns, or constitution. The governing community decides on an ad hoc basis to give consensus (proved against the network) on roadmapped platform functionality. Such functionality may include:

- New delivery areas to cover by the microdistribution network.
- Modification to existing services (pricing, dropshipping, chargeback options etc)
- New services offered by the network (drone delivery, assembly services)
- New rewards and partner integrations.

Participation in the decision making of the Buying.com platform is voluntary. A member with the Buying.com app will be prompted to choose between two or more choices. These choices will allow for the platform to aggregate and deduce the most demanded functionality. With this method, the platfrom is able to govern itself in terms of core product, focusing exclusively on demanded functionality first, with minimal time/energy debt to the member.

5.2 Consensus Algorithm: Federated Byzantine Agreement

The Buying.com protocol uses a modified Stellar Consensus Protocol (SCP) an implementation of Federated Byzantine Agreement (FBA).

The definitions and the discussion below are taken from Mazières (5).

FBA achieves robustness through *quorum slices*: individual trust decisions made by each node that together determine system-level quorums. SCP makes no assumptions about the rational behavior of attackers and does not presuppose a unanimously accepted membership list of nodes.

A quorum is a set of nodes sufficient to reach agreement. A quorum slice is the subset of a quorum convincing one particular node of agreement. A quorum slice may be smaller than a quorum. Non-federated Byzantine agreement requires all nodes to accept the same slices. Because every member accepts every slice, traditional systems do not distinguish between slices

and quorums. The downside is that membership and quorums must somehow be pre-approved, precluding open membership and decentralized control.

FBA generalizes Byzantine agreement to accommodate a greater range of settings. FBA's key innovation is enabling each node to chose its own quorum slice set. System-wide quorums thus arise from individual decisions made by each node. Nodes may select slices based on any number of criteria such as reputation or financial arrangements. In some settings, no individual node may have complete knowledge of all nodes in the system, yet consensus is still be possible.

SCP has the advantage of being *free from blocked states* in which consensus is no longer possible. In this sense SCP prioritizes security over network liveness. SCP is the first provably safe consensus mechanism to exhibit four key properties simultaneously:

- *Decentralized control:* Anyone can participate in the consensus with no central authority approving individual nodes or overall consensus.
- Low latency: The network can reach consensus at timescales commensurate with online transactions (in the order of seconds).
- Flexible trust: Any user node can choose to trust any combination of parties they see fit.
- Asymptotic security: The digital signatures and hash families used by SCP can protect against adversaries with huge computing power.

5.3 Ricardian Contracts

The digital issuance of instruments can be viewed as the issuance of contracts. A Ricardian contract is the issue (6). Such smart contracts not only consist of code but additionally are allowed to contain legal prose. The rationale behind this is to give the code legitimacy that is rooted in the associated legal prose (7). Buying.com uses Ricardian contracts as a way to attach terms and conditions on a per-transaction basis.

5.4 Microdistribution and Delivery Driver Reputation System

A reputation system will be developed to asses microdistributor nodes as well as delivery drivers. This will be helpful in determining staking amounts, inventory allocation and dynamic pricing. The reputation system that Buying.com will implement will be based on the well-documented Eigentrust algorithm for reputation management in P2P networks (8).

5.5 Real-Time Reconciliation

The Buying.com platform will provide an API to supplier partners or other third-party retailers that would allow them to monitor payment authorizations, dates and cash flows. This will be immensely helpful to such partners because it can achieve real-time reconciliation and automate many automate back office procedures.

5.6 Proof of Delivery

During deliveries, drivers are responsible to keep *Proof of Delivery*. When at the time of delivery the customer is present their signature will be required or some other biometric form of id (on the Buying.com app). This along with the date, time, and proof of arrival at the designated delivery location (GPS coordinates) will be recorded. If, on the other hand, the customer is not present at the delivery location (house) then the date of delivery, the time of delivery and proof of arrival at the designated (location coordinates GPS) will be recorded, along with a photograph of the place. Proof of Delivery will be achieved using a multi-signature scheme between the delivery driver, the consumer and the microdistribution node.

5.7 Settlement and Escrow Mechanism

An optional smart contract-based escrow mechanism will be used to settle delivery payments of orders of high-enough value. This escrow mechanism will be available to customers for a nominal fee. The customer will pay tokens into the escrow, then the vendor will release the

goods which are delivered to the customer via a microdistribution node. Proof of delivery as described in Section 5.6, will be employed by this escrow mechanism to ascertain delivery and partially establish initial costumer satisfaction before releasing funds to the vendor.

The Buying.com platformed has also provisioned for a returns processing mechanism. If a customer files an intent to return the delivered goods, then a smart contract allocates the released funds from the vendor to the escrow. Once the customer ships the returned items and the vendor accepts the return—ascertained by Proof of Acceptance, akin to Proof of Delivery—then the smart contract releases tokens to the customer.

5.8 Chargeback Minimization

A chargeback is the return of funds to a consumer, initiated by the issuing bank of the instrument used by a consumer to settle a debt. Even though chargebacks were created as a form of consumer protection, dated industry regulations have allowed chargebacks to pose a threat to retailers. Consumers can use chargebacks for a number of wrong reasons, like avoiding restocking fees on order returns, "buyer's remorse," not acting promptly resulting in time limit expiration and so on (9). Chargebacks can incur transaction fees for the retailer or even fines, and thus have a serious adverse impact on business sustainability. Minimizing chargebacks has been a challenge due to the number of parties involved and the complexity of their interactions. The escrow mechanism utilized by the Buying.com platform for returns processing constitutes an innovative way to minimize chargebacks.

5.9 Third-Party Retailer Integration

The Buying.com platform needs to bring in other retailers, manufacturers, wholesalers for *drop shipping* to work. The *Genesis* feature will allow all retailers, e-commerce players, distributors, wholesalers, even peer-to-peer network participants to upload their inventory. Genesis will know where and what is located in every neighborhood on every street across the United States of America. The asking price, condition, description of every item in Genesis will be known.

This information will be geo-fenced unique to each users location.

To achieve real-time knowledge of existing inventory would require third parties to expose their inventory to the buying.com platform. However, such third parties might be reluctant to share inventory information publicly. This relates to inventory disclosure under regulatory reporting. Reporting Inventory balances of a company's operating segment (financial, manufacturing, raw materials, work-in-progress) can disclose previously hidden risks and thus affect valuation negatively (10, 11).

In order to void the above complication Buying.com third-parties will be required to disclose only necessary inventory. Since IPFS is already used for record keeping of KYC/AML it can also store inventory in order avoid posing unnecessary burden to the blockchain. The system in addition to keeping the inventory it can can keep a user-defined threshold for each inventory item so that as soon as inventory numbers drop below the threshold a re-order is triggered automatically.

6 Network Scalability and Effects

Network effects are crucial for a company like Buying.com because such effects are the most important means to build defensibility for a tech company. The Buying.com ecosystem exhibits several network effects.

- **General Direct Network Effects:** The increased usage of the Buying.com service either as it applies to the microdistribution network or to MOQ ordering will lead to a direct increase in the value of the service and its users. Moreover, because of the topology of the network, certain subgraphs corresponding to the regional delivery networks will be densely connected. Such densely connected subgraphs cement people's commitment to the network and according to Reed's Law (12) the true value of such a network increases *exponentially* with respect to the number of nodes.

- **Protocol Network Effects:** Such effects arise when a computational standard is used by all participating nodes in the network and new potential users can join the network by employing said protocol. This is potentially a very strong network effect because once the protocol has been adopted it is extremely difficult to replace. In the case of Buying.com one of the more well-known blockchain protocols, namely the Stellar protocol, will be leveraged to expedite the Buying.com protocol establishment and its ubiquitous adoption.

- **Personal Utility Network Effects:** Buying.com will have the users' personal identity tied to the network. This makes the daily usage of the network essential to the personal and professional lives of all users. This applies to several categories of users on the Buying.com: network suppliers, consumers, users responsible for order fulfillment, owners of microsdistribution nodes some of which have a reputation score assigned to them (e.g., delivery driver and microsdistribution nodes). For people not being on the Buying.com network this will present significant impediments in the personal and professional since they cannot take advantage of the unique value Buying.com has to offer.

- **Direct Market Network Effects:** Buying.com enhance already existing offline professional distribution networks between suppliers, brick and mortar stores and fulfillments centers. It will achieve this by moving the entire integrated network online and by introducing to the network a finer localization granularity for delivery.

- **N-Sided Marketplace and Platform Effects:** There are several sides to the Buying.com network but they can be roughly distinguished between to buyers and sellers and distributors. A network like Buying.com's is hard to disrupt because it offers a better value proposition to *all three sides simultaneously*. Consumers get faster delivery times at cheaper prices. Distribution and order fulfillment is decentralized allowing people to enter the gig economy with insignificant entry cost, and sellers can optimize last mile delivery costs.

7 Cryptoeconomics – Tokenomics

The following subsections cover aspects of the token economy of the Buying.com network such as staking, pricing, and incentivisation and rewards schemes.

7.1 Staking Mechanism

A crucial aspect of accepting new microdistributor nodes to the existing Buying.com ecosystem is *staking*. During the onboarding process, prospective microdistributor nodes need to stake, i.e., deposit with the Buying.com platform a specified amount of BUY tokens. This staking of tokens has a dual function. First, it functions as an investment into the ecosystem and as a mechanism to encourage network growth in terms of size and consequently in terms of token value. Secondly, these tokens function as *collateral* to the value of the inventory held by a microdistributor. In fact, a microdistributor *cannot hold inventory valued at more than the amount staked*.

The staked amount is flat for all new microdistributor nodes and can be periodically revised depending on the rating of the node provided by the network's reputation system. The reputation system is described in Section 5.4.

7.2 Dynamic Delivery Pricing

The Buying.com platform will leverage algorithms for optimized order delivery (vid. Section 4) to also dynamically determine pricing. Other factors that will be taken into consideration when determining the price include but are not limited to the type of goods delivered (FMCG or not), the delivery time requested by the consumer, demand forecasting, MOQ level and the driver/microdistribution node reputation score.

7.3 Incentivization and Reward Schemes

The Buying.com platform will offer several ways to incentivize engagement and behavior beneficial to the network for both microdistributors and consumers. These incentives will take the form of rewards. More specifically:

- **Holding Costs:** These are associated with costs of storing inventory that remains unsold. Such costs include insurance, security, obsolescence, and rent where applicable. Microdistributor nodes need to be incentivized to store goods that are not as fast moving as others. This incentivization can be achieved rewarding nodes with a higher percentage of the delivery cost.

- **Delivery Options:** Consumers placing an order using a the Buying.com will be given several delivery options.

    1. Door-to-door home delivery within 24 hours is the standard option.
    2. Door-to-door home delivery within 2-hours is only applicable to FMCG.
    3. Option to pick-up from a local microdistribution node (e.g., walk three doors down to the neighbor's garage to pick up your order).

    The reasons for the pick-up option being more attractive to certain consumers becomes clear when one considers issues of security and convenience. This options caters to people with busy schedules who might not be present at home during delivery times. Hence, any parcel left outside the house becomes liable to theft. Such consumers might prefer to pick up their order at their convenience. For this reason no discount will be offered for the pick-up option. This reflect a growing trend among established industry players like Amazon of making delivery more *customer-centric*. The delivery destination is no longer tied to a particular place (e.g., home) but is customer location-aware.

- **Subscription Plan:** When ordering FMCG on the Buying.com app, the system will the consumer the option to an option to switch to a subscription plan. Pushing consumers towards recurrent orders helps with better demand forecasting and inventory optimization for the microdistribution nodes. Switching to a subscription plan will be incentivized by offering additional price reductions.

- **Recommender System:** A product recommendation system alerts consumers to further price reductions opportunities. Buying.com has the ability to detect ordering patterns (e.g., locality and seasonality) by using AI and machine learning techniques and recommend to the consumer to join a particular order pool. This has the effect of maximizing the probability of achieving MOQ levels and thus wholesale pricing.

References

1. Statista, Global B2B e-commerce gross merchandise volume (GMV) from 2013 to 2017 (in billion U.S. dollars) (2018). `https://www.statista.com/statistics/705606/global-b2b-e-commerce-gmv/` (accessed August 2018).

2. Amazon.com, Amazon Flex. `https://flex.amazon.com/about` (accessed October 2018).

3. Stellar Development Foundation, Stellar network. `https://www.stellar.org` (accessed August 2018).

4. P. Toth, D. Vigo, *Vehicle routing: problems, methods, and applications* (SIAM, 2014).

5. D. Mazieres, The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus, *Stellar Development Foundation* (2015).

6. I. Grigg, The Ricardian Contract, *First IEEE International Workshop on Electronic Contracting* (IEEE, 2004), pp. 25–31. http://209.197.106.187/papers/ricardian_contract.ps (accessed August 2018).

7. M. Valenta, P. Sandner, Comparison of Ethereum, Hyperledger Fabric and Corda, *Tech. rep.*, Frankfurt School Blockchain Center Working Paper (2017).

8. S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The Eigentrust Algorithm for Reputation Management in P2P Networks, *Proceedings of the 12th International Conference on World Wide Web*, WWW '03 (ACM, New York, NY, USA, 2003), pp. 640–651.

9. Chargebacks911, What is a Chargeback? https://chargebacks911.com/chargebacks/ (accessed August 2018).

10. CFO.com, FASB Eyes Big Changes in Inventory Disclosure (2017). http://ww2.cfo.com/financial-reporting-2/2017/02/fasb-changes-inventory-disclosure/ (accessed August 2018).

11. Financial Accounting Standards Board, Proposed Accounting Standards Update (2017). https://goo.gl/ejUdw9 (accessed August 2018).

12. D. P. Reed, The Law of the Pack, *Harvard Business Review* pp. 23–24 (2001).