()

When is it Beneficial to Provide Freelance Suppliers with Choice? A Hierarchical Approach for Peer-to-Peer Logistics Platforms

Seyed Shahab Mofidi, Ph.D.
Rensselaer Polytechnic Institute, mofids@rpi.edu,

Jennifer A Pazour, Ph.D.
Rensselaer Polytechnic Institute, pazouj@rpi.edu,

This paper proposes and evaluates a new hierarchical approach to peer-to-peer logistics platforms, recasting the platform's role as one providing personalized menus of requests to freelance suppliers. A bilevel optimization formulation explicitly models the two stage decision process: first, the platform determines which set of requests to recommend to which suppliers, and second, suppliers have a choice to select which request (if any) to fulfill. By harnessing the problem's structure, the computationally expensive mixed-integer linear bilevel problem is transformed into an equivalent single-level problem that is computationally superior. A computational study based on ride-sharing quantifies the value of providing suppliers with choices. When a platform's knowledge of suppliers' selections is imperfect, our hierarchical approach outperforms existing recommendation methods, namely, centralized, many-to-many stable matching, and decentralized approaches. We show that a platform's lack of knowledge over suppliers' selections can be compensated by providing choices in environments with either inflexible suppliers or when suppliers' utilities have higher variance than the platform's utilities. In these environments, providing choices and recommending alternatives to more than one supplier can be beneficial to not only the platform, but also freelance suppliers and demand requests.

Key words: bi-level optimization, recommendation sets, choice, ride-sharing, crowdsourced delivery

1. Introduction.

Peer-to-peer resource sharing platforms, exemplified by companies like Uber and Airbnb, are a disruptive new business model that match independent suppliers with demand requests. Nonexistent just ten years ago, it is now a $75 billion industry (Allen 2015). Because the resources are accessed when needed and not owned by a central system, supply capacity is elastic, and can be scaled up and down, as well as moved in response to changing demand requirements. If requests can be interleaved with suppliers' planned tasks, platforms can tap into otherwise underutilized or idling capacity. However, these systems are inherently more complex than traditional systems because supply capacity is not set. Instead, a fluid set of suppliers must be enticed to provide access to their resources.

To manage this complexity, internet-based platforms facilitate the interactions between suppliers (who provide access to their resources) and demand requests.

Peer-to-peer logistics platforms match independent suppliers to logistics requests (e.g., transport people or goods, store goods, fulfill orders). A wide range of such platforms exist across the supply chain. For example, companies like Uber and Lyft operate platforms matching people that need rides with people who are willing to provide them. Whereas, Deliv and Grubhub are platforms for last-mile delivery. Flexe and Ware2Go are platforms for warehouse space and order fulfillment capabilities. Instacart is a platform for order-fulfillment and last-mile delivery of groceries. Cargomatic connects shippers with local licensed carriers. The nonprofit American Logistics Aid Network matches logistics capabilities with community and agency disaster needs. To capture the variety of peer-to-peer logistics platforms, in what follows, we use *supplier* to refer to the ad hoc supply participants (who provide access to their resources), and *request* to denote demand for a supplier's resource.

Existing peer-to-peer logistics platforms use either a centralized or a decentralized approach. A centralized approach, such as used by Uber and Instacart, prioritizes meeting demand commitments and enabling a quick time to match. However, supplier preferences are not considered. As a result, who is able to participate is limited, and utilizing underutilized capacity is not prioritized. For example, Uber takes a rider-centric view and ignores driver's preferences or planned routes. The driver has 15 seconds to accept a recommended request (without knowing its destination), and is penalized for low acceptance rates (Cook 2015). A decentralized approach, such as used by Flexe and Cargomatic, makes all requests available to all suppliers. Suppliers must spend time to find suitable matches, which limits quick resource allocation. For example, suppliers spent two hours a week (on average) finding matches in Taskrabbit's decentralized system (Newton 2014). Due to myopic supplier selections, reduced systematic performance, in which some requests receive multiple selections and others receive none, is an observed problem in decentralized platforms (Cullen and Farronato 2014, Einav et al. 2016, Fradkin 2017, Horton 2014).

This research proposes a new hierarchical approach, in which the platform - without control nor perfect knowledge of suppliers' utilities - uses choices estimated from suppliers' past behavior to increase participation. As shown in Figure 1, we recast the platform's role as one providing personalized recommendations (i.e., a menu of requests) to a set

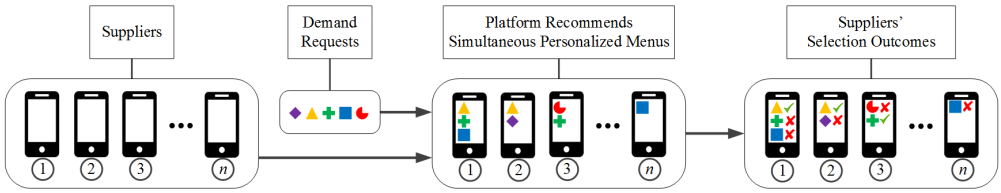

Figure 1 **A hierarchical approach to decentralized resource allocation.**

of suppliers. The platform first decides how multiple, simultaneous recommendations are made. The same request may be recommended to multiple suppliers to decrease the time to match and to hedge against suppliers' autonomy to decline recommendations. Second, suppliers have autonomy to select requests (if any) from the personalized menus, resulting in some requests not selected and others selected by more than one supplier. This hierarchical approach enables a quick time to match, and does not require suppliers to explicitly provide preference information for all requests. It also allows for systematic coordination of suppliers' resources to try and reduce duplicate and rejected requests. Suppliers retain autonomy to select requests that can be interleaved with their planned activities, which both increases who is willing to participate as suppliers, as well as improves resource utilization of existing, idle capacity.

The objectives of this research are to determine when providing suppliers with choices is beneficial to a platform that must coordinate demand requests with decentralized owned resources, and to quantify this benefit to the platform, the suppliers, and the demand requests under different environmental factors. In doing so, this paper makes the following contributions. This work is the first to explicitly model both the platform's decisions and the suppliers' selection decisions in an optimization framework. We develop a new bi-level optimization framework to decide how a platform should make simultaneous supplier recommendations. This is challenging because the platform's objective is influenced by both the platform's recommendation decisions and the suppliers' interdependent selection outcomes. A reformulation technique exploits the problem's structure. We prove our single-level reformulation technique is equivalent to the bi-level formulation and show it can solve moderate-sized problems quickly. Using this framework, we answer the open research question: when is it beneficial for a platform to use personalized recommendations to a set of freelance suppliers to coordinate demand requests? We simulate different scenarios using ride-sharing as a computational study. When a platform is uncertain about suppliers' selections, the hierarchical recommendation model outperforms existing centralized,

many-to-many stable matching, and decentralized approaches. In platforms where uncertainty over suppliers' selection exists, choices help the platform when (1) suppliers are inflexible with a high no-choice utility; and (2) suppliers' utility values have higher variance than the platform's utility values. Choices increase the chance of enticing supplier participation, and in both cases, the increase in participation likelihood outweighs the match's reduced platform benefit likelihood. However, when (1) the platform's benefit values have higher variance than the suppliers' expected utilities or (2) suppliers are flexible, offering additional choices, on average, can reduce the platform's benefit.

2. Literature Review.

Resource sharing platform research is emerging. Many descriptive studies exist (Bardhi and Eckhardt 2012, Deakin et al. 2010, Hampshire and Sinha 2011, Murphy 2016, Narasimhan et al. 2018, Rougès and Montreuil 2014, Shaheen and Cohen 2013, Steininger and Bachner 2014, Weber 2016). Prescriptive models are growing (Benjaafar et al. 2018, Jiang and Tian 2016). Subgroups include shared mobility, such as crowdsourced delivery and ride-sharing systems, in which drivers with excess capacities (e.g., empty seats, extra delivery space) are assigned to perform additional tasks (e.g., parcel delivery or providing a passenger a ride) (Cleophas et al. 2019, Mourad et al. 2019). Dynamic ride-sharing focuses on matching drivers with riders to share one-time trips (Agatz et al. 2012, Furuhata et al. 2013, Li et al. 2014, Liu and Li 2017, Masoud and Jayakrishnan 2017, Nourinejad and Roorda 2016, Pelzer et al. 2015); all consider either a centralized or decentralized approach. Many capture more complicated scenarios; for example, Stiglic et al. (2016) study the use of relay points for drivers to match with more than one rider. Lee and Savelsbergh (2015) and Arslan et al. (2018) consider a dedicated fleet of drivers as a backup plan for serving requests not matched. However, at their core, all assume driver full compliance of the platform's decisions. None incorporate supplier choice in the optimization model.

While the needs of independent freelances have recently received great public attention, it has attracted less academic research attention. One exception is Wang et al. (2017), in which the platform's and drivers' preferences are known and considered by enforcing dynamic ride-sharing matches to be stable. They empirically show that not taking the preferences of drivers into account could lead to unsustainable ride-sharing systems in the long run. Other works capture drivers' willingness to participate and availabilities with

additional constraints (i.e., limiting number of stops or detour distances but only if the information is provided by the drivers in advance (Arslan et al. 2018, Archetti et al. 2016). A centralized approach to crowdsourced delivery requires drivers to provide bids a day in advance (Kafle et al. 2017). Bottom-up agent-based approaches include Kleiner et al. (2011), Winter and Nittel (2006), Xing et al. (2009), in which individual suppliers match themselves based on self-interest and without centralized supervision. While not motivated by resource sharing, related is Powell et al. (2000), in which a platform recommends a single alternative to a single dispatcher who can accept or reject it. Powell et al. (2000) neither incorporates discretion decisions into the optimization model nor considers dependencies in systematic performance due to multiple suppliers' decisions. Thus, our approach is innovative, as none of the existing research for matching supply and demand in platforms considers hierarchical approaches modeling a set of suppliers with discretion in the optimization model.

This is the first work to determine if personalized recommendations made to suppliers can be used to coordinate decentralized resource allocation decisions. Related is supply chain coordination literature, given its shared focus on coordinating decentralized entities (Cao and Zhang 2011). However, coordination mechanisms are designed to influence *demand* using pricing (Bernstein and Federgruen 2005, Chen et al. 2001), inventory discounts (Qi et al. 2004), information sharing (Ha et al. 2017, Lee and Whang 1999), buy-back contracts (Heydari et al. 2017, Krishnan and Winter 2010), and revenue sharing (Cachon and Lariviere 2005, Govindan and Popiuc 2014, Tsay and Agrawal 2004). In contrast, this work considers personalized recommendations as mechanisms to coordinate decentralized *supply*. To entice suppliers, Cachon et al. (2017) analyze platform pricing and wage contracts; while Bai et al. (2018), Chen and Sheldon (2016), explore dynamic pricing, but none consider recommendation sets.

Recommender systems are a well-studied response to the problem of creating personalized recommendations that a platform hopes will be of interest to a certain user. While different approaches to recommender systems exists, all "rely on some kind of representation of users' preferences" (Price and Messinger 2005). "Since information about users is incomplete and uncertain and since preference representations are often based on simplified models of users and alternatives, the single best alternative for the user is uncertain. The common solution to this problem is to present the user with a set of the top-k alternatives

in the hope that one of these alternatives will be the true best" (Price and Messinger 2005). While these deterministic methods do not guarantee optimal recommendation sets in the stochastic environment, they are commonly used and were the starting point of this field. The field of recommender systems literature varies from this work as what is being recommended has an infinite capacity and no penalty exists for something not being selected. Recommendation decisions are made independently for each user. Methods focus on predicting user preferences, not combinatorial optimization (Ekstrand et al. 2011, Lops et al. 2011).

Market theory is a well-studied decentralized approach to distributed resource allocation. Related are centralized matching algorithms that take rank-order preferences of bipartite sets and suggest matches achieving a given criteria, such as stability, Pareto optimality, strategy proofness, social welfare (Gale and Shapley 1962, Roth and Sotomayor 1992). Achieving optimal performance for all criteria is not possible due to incompatibility (Anshelevich et al. 2013). Constraints to overcome imbalances and to meet quotas are studied in a decentralized market (Fragiadakis et al. 2016, Kamada and Kojima 2014), but a hierarchical approach using recommendation sets in matching mechanisms has not been studied. While combinatorial auctions allow interdependency among a user's utility through request bundles, interdependency in systematic performance is not captured. Markets for peer-to-peer resource sharing platforms have been cumbersome in practice due to effort to elicit supplier information, and for suppliers to identify requests and determine bids (Einav et al. 2016). In this work, the platform prioritizes match quickness and supplier effort reduction. Thus, personalized recommendations are a substitute for a market, and systematic performance, influenced by interdependent suppliers' selections, is not additive.

The platform must consider suppliers' selection behavior when making personalized recommendations, because suppliers have autonomy not to comply with the recommendations they do not prefer. Thus, our hierarchical approach will be formulated as a bi-level program. Also known as *Stackelberg Leader-Follower Games* (Bracken and McGill 1973, Von Stackelberg 1952), bi-level programs model a hierarchy, in which a leader makes decisions that affect the followers' feasible decision set (Colson et al. 2007, Dempe 2018). Bi-level optimization problems are recognized as difficult (Dempe and Franke 2016); even linear bi-level programming is NP-hard (Audet et al. 1997, Bard 1998, Frangioni 1995). Focus has been on developing specialized algorithms for specific problems. Effective specialized approaches

exist for network interdiction, which model the attacker-defender problem (Sullivan et al. 2014). In transportation network design problems, the network design and its capacities are leader decisions; followers solve shortest path problems (Brotcorne et al. 2001, Farahani et al. 2013, Gao et al. 2005). Assortment optimization determines what product assortment to offer to an aggregated set of shoppers, who then make consumer selection choices from the recommended assortment (Kök et al. 2015). Capacity constraints of a single shared assortment have been considered in assortment optimization (Feldman and Topaloglu 2015). In all problem types, a single aggregate leader decision is made and shared among the followers. For example, all users make routing or interdiction decisions using the same shared network, and all consumers make purchasing selections from the same product assortment. Revenue management considers what assortment of products (e.g., flight tickets and prices) to display to what segment of users at varying times to sell remaining capacity of a finite resource (Bitran and Caldentey 2003, Gallego et al. 2014, Phillips 2005). Existing revenue management work assumes demand selections occur sequentially and recommendations are independent of each other. Similarly, dynamic assortment optimization considers personalized recommendations of finite capacities to sequentially arriving customers (Bernstein et al. 2015, Johari et al. 2016). In contrast, this work makes multiple, *simultaneous* personalized recommendations. Existing work in bi-level optimization either models the leader problem as deciding (i) a single aggregate recommendation, or (ii) multiple independent recommendations. The bi-level models proposed here are innovative because the leader problem has to make multiple personalized recommendations, in which platform performance is dependent on multiple selection outcomes.

3. Hierarchical Approach: A Bi-Level Optimization Framework

The bi-level optimization framework (BLF) models the platform as the leader, who decides what personalized subset of demand requests to recommend to a set of decentralized independent suppliers (the followers). A single period (static) model is presented. Thus, we assume that the platform makes recommendations at defined decision epochs (e.g., Grubhub resolves a static optimization problem every minute). At each decision epoch, the following sets exist: a set of potential suppliers $S = \{1, 2, \ldots, n\}$, a set of demand requests to be fulfilled $R = \{1, 2, \ldots, m\}$, and the no-choice set N, which has a single element $N = \{0\}$. Recommendations to suppliers are a subset of the set of alternatives A, which is the union

of the requests and the no-choice sets, i.e., $A = R \cup N$. Given all suppliers must be recommended the no-choice alternative, the use of a no-choice set is what allows us to capture suppliers' ability to have autonomy to reject all recommended requests and not participate. The menu size, θ, denotes the required number of integer requests (excluding the no-choice alternative) to be recommended to each supplier.

We model the case in which the set of suppliers have specific preferences for the set of alternatives. We assume the platform can estimate the expected value of each suppliers' utility for each alternative. This estimate is based on suppliers' past behavior collected as historical information by the platform (e.g., past destinations given time of day and current origin), and current information captured via the platform (e.g., suppliers' current GPS location). Therefore, a deterministic optimization model is presented, in which the platform inputs the expectation of suppliers' estimated utilities (ν_{ij}) to make recommendation decisions.

Demand requests are assumed to have a capacity of 1 and have no discretion (i.e., demand users will accept any supplier). Recommendation menus are made simultaneously to the set of suppliers and overlapping requests can be recommended to more than one supplier. Thus, demand request $i \in R$ can experience one of three observed outcomes after suppliers' selections: A request can be selected by (1) multiple suppliers (a collision); (2) no suppliers (a rejection); or (3) one and only one supplier (a 1-to-1 match). These are platform outcomes, which occur due to multiple suppliers' selections and are captured in the leader objective function in (1).

Decision Variables:

x_{ij} 1 if the platform recommends alternative $i \in A$ to supplier $j \in S$; 0 otherwise.
y_{ij} 1 if supplier $j \in S$ selects alternative $i \in A$; 0 otherwise.
z_i Integer number of suppliers who select request $i \in R$, but are not matched (collision)
w_i 1 if no supplier selects request $i \in R$ (rejection); 0 otherwise.

Bi-Level Optimization Formulation (BLF):

$$\max \quad \sum_{i \in R} \sum_{j \in S} c_{ij} y_{ij} - \sum_{i \in R} d_i z_i - \sum_{i \in R} r_i w_i \tag{1}$$

$$s.t. \quad \sum_{i \in R} x_{ij} = \theta \quad \forall j \in S \tag{2}$$

$$\sum_{j \in S} x_{ij} \leq a_i \quad \forall i \in R \tag{3}$$

$$x_{0j} = 1 \quad \forall j \in S \tag{4}$$

$$\sum_{j \in S} y_{ij} \leq z_i + 1 \quad \forall i \in R \tag{5}$$

$$1 - \sum_{j \in S} y_{ij} \leq w_i \quad \forall i \in R \tag{6}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in A, \quad \forall j \in S \tag{7}$$

$$z_i \in \{\mathbb{Z}_{\geq 0}\} \quad \forall i \in R \tag{8}$$

$$w_i \in \{0,1\} \quad \forall i \in R \tag{9}$$

$$\max \sum_{i \in A} \sum_{j \in S} \nu_{ij} y_{ij} \tag{10}$$

$$s.t. \quad y_{ij} \leq x_{ij} \quad \forall i \in A, \quad \forall j \in S \tag{11}$$

$$\sum_{i \in A} y_{ij} \leq 1 \quad \forall j \in S \tag{12}$$

$$0 \leq y_{ij} \leq 1 \quad \forall i \in A, \quad \forall j \in S \tag{13}$$

The leader problem is captured in (1)-(9). In (1), the platform takes a holistic view by maximizing it's expected benefit. If request $i \in R$ is selected by supplier $j \in S$, c_{ij} denotes the platform's benefit. We let d_i denote the linear collision penalty per supplier who selected request i, but was not matched, and r_i denote the rejection penalty per request if request $i \in R$ is not selected by any suppliers. Thus, the platform's objective function depends on the suppliers' decisions (i.e., notice y_{ij} and not x_{ij} in (1)). Also, (1) captures the dependencies among suppliers' selections by accounting for collisions and rejected request penalties. With z_i being an integer variable, the second term in the platform's objective function penalizes linearly per number of suppliers who selected a request but did not fulfill it. For example, if request i is selected by three suppliers, only one supplier can provide the service, two other suppliers are told by the platform their service is not needed, and the corresponding z_i variable is 2. To model a platform that only receives benefit in a 1-to-1 match, d_i should be set such that $d_i > max_{j \in S} \{c_{ij}\}, \forall i \in R$.

Constraints (2) enforce that all suppliers $j \in S$ must be recommended menus of size θ. By changing θ values, we use this model to quantify the impact of providing different number of choices on the platform and suppliers' performance (see Sections 6 and 7). Constraints (3) enforce at most a_i suppliers can be recommended request $i \in R$. Constraints (4) enforce the no-choice alternative has to be recommended to all suppliers. Thus, all suppliers have autonomy to choose the no-choice alternative, i.e., have autonomy to choose not

to participate. Constraints (5) and (6) capture the dependency among multiple suppliers, whether a request is selected by more than one supplier (collision); or it is not selected by any suppliers (rejection), respectively. Constraint (7) - (9) provides bounds on the leader's decision variables.

The suppliers' (followers) problem is captured in (10)-(13). Suppliers are assumed myopic rational utility maximizers without a commitment to the platform and make decisions independently of other suppliers. Multiple follower decisions are captured in a single objective function, i.e., maximize suppliers utilities, summed over all suppliers in (10). Because each of the suppliers' decisions are made independently of the other suppliers, the lower-level problem given in (10)-(13) is decomposable by supplier j into multiple, independent follower problems. For alternative $i \in A$, ν_{ij} denotes the platform's expectation of supplier $j \in S$ utility. These expectation values are assumed all to be greater than or equal to zero (i.e., $\nu_{ij} \geq 0 \forall i \in A, \forall j \in S$). Constraints (11) enforce suppliers' selections to be a subset of their recommendation set. In (12), supplier $j \in S$ must select at most one alternative, from the requests recommended and no-choice option. Therefore, if the supplier's no-choice option has higher utility than any of the other recommended requests, the lower-level problem captures that the supplier will choose the no-choice option (which represents the supplier not participating). The followers' decision variables, while binary, can be relaxed to continuous in constraints (13) because in standard form, the lower level problem's constraint coefficient matrix is unimodular and each element of the right hand side vector is either 0 or 1 (integers) (Shapiro 1979).

4. A Specialized Solution Approach

The bilevel optimization framework in this research is a discrete-continuous linear bilevel programing problem (DCL-BLPP). Although general algorithms and open sourced solvers exist (Bard and Moore 1992), challenges in solving large scale bi-level problems remain (Bard 1998, Zeng and An 2014). Therefore, we develop a specialized solution approach to convert the BLF problem, represented in (1)-(13), into an equivalent single level problem. The reformulation technique is focused on the lower-level problem (i.e., the suppliers myopic selections). Specifically, we replace the lower level objective function in (10) with a set of linear constraints. These constraints enforce a set of logical, if-then statements: if the platform recommends a given recommendation set, then the supplier will select the

alternative recommended with the highest utility. This is achievable because the platform makes recommendations based on the assumption that suppliers' selection behaviors are based on maximizing their expected utility values, ν_{ij}. For example, if $\nu_{1j} > \nu_{2j} > \nu_{3j}$, supplier j prefers alternative $i=1$ over $i=2$ over $i=3$ (i.e., $1 \succ_j 2 \succ_j 3$). Thus, the platform can determine supplier j will select alternative 3, only if alternatives 1 and 2 are not recommended (i.e., if $x_{1j}=0$, $x_{2j}=0$, and $x_{3j}=1$, then $y_{3j}=1$). Thus, supplier j's selection behavior of choosing the recommended alternative with the supplier's highest utility – previously captured by the lower level objective function (10) – can be captured with a set of linear constraints in (14). The constraint set (14) uses a new set of binary variables t_{ij}, which is 1 if $x_{ij}=1$ (i.e., if supplier j selects alternative i); 0 otherwise.

$$\begin{aligned} +x_{1j} \leq t_{1j},\ y_{1j} \geq t_{1j}\ for \quad k=1 \\ -x_{1j}+x_{2j} \leq t_{2j},\ y_{2j} \geq t_{2j}\ for \quad k=2 \\ \vdots \quad \vdots \quad \vdots \\ -x_{1j}-x_{2j}-x_{3j}\ldots-x_{(q-1)j}+x_{qj} \leq t_{qj},\ y_{qj} \geq t_{qj}\ for \quad k=q \end{aligned} \tag{14}$$

Constraints (14) represent the set of linear constraints for only one supplier j. Because each supplier can have a different highest alternative, second highest, and so on, we use the notation k to represent an alternative's ranking index, given supplier j's preference profile. These constraints represent all the possible selection outcomes for supplier j and capture that the only way supplier j will pick its kth most desired alternative, is if none of the $k-1$ and higher alternatives are offered in the recommendation. Supplier j will pick their $k=1$ alternative if it is recommended (refer to the first line in (14)). However, supplier j will pick their $k=2$ highest alternative only if their first highest alternative is NOT recommended (refer to the second line in (14)). The least preferred alternative has an index of $k=q$ and supplier j will only pick their qth highest alternative if no other alternative is recommended. Similarly, we can write the logical expressions for all other suppliers based on their preferences. We represent the coefficients of x_{ij}, y_{ij}, and t_{ij} in (14) by α_{ijk}, β_{ijk}, and γ_{ijk}, respectively. These coefficient are used in the constraints (15)- (17), which capture the selection decision of suppliers for all possible recommendation sets. Thereby, Algorithm 1 takes suppliers' expected utility matrix $V=[\nu_{ij}]$, and outputs coefficient tensors $A=[\alpha_{ijk}]$, $B=[\beta_{ijk}]$, and $\Gamma=[\gamma_{ijk}]$. The numerical example below shows how the third dimension of the tensor (i.e., k) is constructed with this algorithm. This example has 3 suppliers and

3 alternatives, and the given V matrix, $\nu_{ij} = \left[\begin{smallmatrix} 3 & 2 & 1 \\ 2 & 1 & 2 \\ 1 & 3 & 3 \end{smallmatrix}\right]$. Then, for this example, the resulting tensor values for α_{ijk} are: $\left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{smallmatrix}\right]$ for $k=1$; $\left[\begin{smallmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & -1 \end{smallmatrix}\right]$ for $k=2$, and $\left[\begin{smallmatrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{smallmatrix}\right]$ for $k=3$. For $\beta_{ijk} = \gamma_{ijk}$ are: $\left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{smallmatrix}\right]$ for $k=1$; $\left[\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{smallmatrix}\right]$ for $k=2$, and $\left[\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right]$ for $k=3$. The layer $k=1$ of the tensors represents the situation where all suppliers are given their first preference. For this example, $\alpha_{311} = \beta_{311} = \gamma_{311} = 1$ construct constraints that forces $y_{31} = 1$ if $x_{31} = 1$ through (15)- (17).

Algorithm 1 Output Coefficients for Logical Expression Constraints

// Initial Call: $A \leftarrow 0_{q \times n \times q}$, $B \leftarrow 0_{q \times n \times q}$, $\Gamma \leftarrow 0_{q \times n \times q}$, where $q = |k|$
for $j = 1$ to n **do**
  for $k = 1$ to q **do**
    $i \leftarrow \arg\max_{i \in A} V_j$
    $\alpha_{ijk} \leftarrow 1$, $\beta_{ijk} \leftarrow 1$, $\gamma_{ijk} \leftarrow 1$
    Remove i from V_j
    for $k' = k+1$ to q **do**
      $\alpha_{ijk'} \leftarrow -1$
Return A, B, and Γ

$$\sum_{i=1}^{q} \alpha_{ijk} x_{ij} - \sum_{i=1}^{q} \gamma_{ijk} t_{ij} \leq 0 \quad \forall j \in S, \quad \forall k \in K \tag{15}$$

$$\sum_{i=1}^{q} \gamma_{ijk} t_{ij} - \sum_{i=1}^{q} \beta_{ijk} y_{ij} \leq 0 \quad \forall j \in S, \quad \forall k \in K \tag{16}$$

$$t_{ij} \in \{0,1\}, \quad \forall i \in A, \quad \forall j \in S \tag{17}$$

The reformulation technique (RFT) is to solve the following single-level optimization model: maximize (1), subject to (2) - (9), (11) - (13), (15) - (17). That is, the reformulation technique keeps the upper level objective and constraints, which continue to enforce the platform outcomes due to interdependence in multiple suppliers' selections, and adds the lower-level objective as a set of constraints. The specific reformulation technique used here adds $q \times n$ variables and $2q \times n$ constraints to the original BLF problem. However, the formulation could be completed without the use of t_{ij} variables and advanced approaches could be applied to generate constraints only as needed in an iterative approach.

THEOREM 1. *RFT and BLF are equivalent.*

Proof: Equivalency requires for a given solution to the upper level problem in BLF (i.e., given a recommendation matrix x_{ij}), the feasible region of (11) -(13), (15) - (17) is a unique point equivalent to the optimal solution to (10)-(13). We provide the proof for one driver j and extend it for $\forall j \in S$. Given $x_{ij} \in \{0,1\}$ and constraint set (4) enforces that the no-choice alternative is always offered to all suppliers, it is guaranteed that at least one of the x_{ij} variables is nonzero (i.e., $x_{ij} \neq 0, \exists i \in A$). Then, according to (11), the upper bounds for corresponding y_{ij} variables are one in the BLF. Due to unimodularity of the lower-level of the BLF and (12), only one of the y_{ij} variables can be equal to one. Because the lower-level of the BLF objective, given in (10), is to maximize, all ν_{ij} values are greater than or equal to zero, and the lower-level problem captures suppliers' independent decisions, a single y_{ij} is set to one in BLF's optimal solution that has the highest corresponding multiplier ν_{ij} among the alternatives in the recommendation set, i.e., $y_{ij} = 1$, where $i = \arg\max_{i \in A | x_{ij} \neq 0} \nu_{ij}$. In RFT, because the upper level constraint (4) still applies, at least one of the x_{ij} variables will be nonzero (i.e., $x_{ij} \neq 0, \exists i \in A$). This implies that only one of the constraints in (15) for each j is binding, and thus the lower bound on one of the t_{ij} variables is one. Consequently, due to (16), the lower bound on the corresponding variable y_{ij} is also one. For verification, consider the first two lines in (14) assuming two alternatives, $i = 1$, $i = 2$. If the recommendation vectors are $x_{ij} = (1,0)$ or $x_{ij} = (1,1)$, this will result in $t_{1j} = 1$ and thus $y_{1j} = 1$. This makes the first row of constraint (14) binding. Similarly, for the recommendation vector $x_{ij} = (0,1)$, this makes the second row of the constraint set binding, resulting in $t_{2j} = 1$ and thus $y_{2j} = 1$. Moreover, the upper bound for only one of the y_{ij} variables can become one because of (12). Therefore, it is guaranteed that one and only one $y_{ij} = 1$ is equal to one for each supplier j. Due to Algorithm 1, line 4, the binding constraint has y_{ij} with the highest corresponding multiplier ν_{ij}, i.e., $y_{ij} = 1$, where $i = \arg\max_{i \in A} \nu_{ij}$. As suppliers' selections in the lower level are independent, this argument can be extended for all drivers $\forall j \in S$. End Proof.

In Appendix A we test the computational performance of the reformulation technique (RFT) against a common bi-level solution approach. We find that the reformulation technique is computationally superior and able to solve reasonably-sized problem instances quickly.

5. Optimal Recommendation Set Properties

In this section, we explore properties about the optimal solutions from our bi-level optimization problem. A natural question is why not use the recommendation set when $\theta = 1$ and then add choices when the menu size requirement increases. However, as we show in Theorem 2, the optimal recommendation sets are not guaranteed to be a subset of smaller menu sizes.

THEOREM 2. *The optimal recommendation set with menu size $\theta = 1$ is not guaranteed to be a subset of the optimal recommendation set with a menu size θ greater than one.*

Proof: Proof by counterexample: Let there be two suppliers $S = \{1, 2\}$ and three requests $R = \{1, 2, 3\}$, and the no-choice alternative is not offered, i.e., $N = \emptyset$. The platform's benefit matrix and the suppliers' expected utility matrices are $c_{ij} = [(1, 1), (2, 3), (3, 2)]$, and $\nu_{ij} = [(3, 1), (2, 3), (1, 2)]$, respectively, and $d_i = r_i = 1$. If the menu size is $\theta = 1$, the optimal recommendation matrix and the selection matrix are $x_{ij} = [(0, 0), (0, 1), (1, 0)]$, and $y_{ij} = [(0, 0), (0, 1), (1, 0)]$, respectively. Now consider the same utility values, but with a menu size of $\theta = 2$; the optimal recommendation matrix and the selection matrix are: $x_{ij} = [(0, 1), (1, 0), (1, 1)]$, and $y_{ij} = [(0, 0), (1, 0), (0, 1)]$, respectively. When the menu size is two, supplier 2 is recommended alternative 2. Whereas, with a menu size of 1, alternative 2 is removed from the recommendation set and instead, alternative 1 and 3 are recommended. The optimal recommendation set with $\theta = 1$ is not a subset of the optimal recommendation set with a larger menu size. End Proof.

Consequently, creating recommendation sets, with a given menu size, for multiple suppliers is challenging due to the interaction effects of suppliers' outcomes captured in the platform's objective function. The problem is less challenging if the menu size is flexible; i.e., when constraint (2) is relaxed from a "=" to a "$\leq$" constraint. The optimal solution to the flexible menu size is equivalent to solving for a single menu size (i.e., $\theta = 1$) and then adding alternatives to suppliers recommendations that are strictly less attractive to the suppliers (i.e., with lower ν_{ij} values than ν_{0j}). In Theorem 3, we prove for *flexible* menu sizes that if a platform has perfect knowledge about suppliers' selections, providing choice cannot improve the platform's objective value.

THEOREM 3. *It is guaranteed that when suppliers are rational utility maximizer, and their selection behavior is deterministically known to the platform, in a flexible menu situation, adding one more alternative either degrades the platform's objective function value or it remains the same.*

Proof: An optimal solution to BLF, when constraint (2) is relaxed from a "=" to a "$\leq$" constraint, is to recommend at most one request to all suppliers, i.e., to have $\sum_{i \in R} x_{ij} \leq 1 \forall j \in S$. Adding additional alternatives has only two outcomes. Either the recently added alternative has a higher supplier utility value than the supplier's current selection, or the supplier's utility value of the new alternative is less than or equal to the supplier's current selection. In the former case, the supplier would select the new alternative instead of the previous selection to increase her utility, which would result in an equal or worst benefit for the platform. Because otherwise, if an alternative can increase both the platform's objective function and the supplier's utility, that alternative would be in the optimal recommendation set in the first place. In the latter case, the supplier has no incentive to change her selection, assuming that in the case of alternatives with equal supplier utility values, the supplier will always select with a tie breaking rule known to the platform. Thus, the benefit of the platform remains the same in this case, because the supplier does not change her selection. Instead, multiple optimal solutions may exist, but an optimal solution is one without additional alternatives. End Proof.

6. Computational Study Structure

Acquiring suppliers' utilities in advance is typically not feasible because eliciting utility values of all requests is time consuming and tedious for suppliers. Therefore, suppliers' selections are typically not fully known by the platform. Instead, exogenous factors play a role in supplier selection, resulting in a platform being able to only partially estimate suppliers' selection outcomes through, for example, multi-attribute random utility models (Huber 1974, Train 2009). In such cases, providing choices to suppliers has the potential to improve the platform's objective function. As the menu size θ increases, suppliers have a higher chance to be recommended a request they are willing to select. This benefits the platform, up to a point. However, due to misalignment between the suppliers and the platform's utilities, larger values of θ lead to higher chances of suppliers selecting a request with lower platform benefit. Also, as the number of choices increases, less systematic

coordination occurs. Thus, higher chances of rejected or collided requests negatively impact the platform's objective function. Therefore, in this section we explore the concept of providing choice when a platform has uncertainty about suppliers' selections.

We develop a computational study to investigate whether personalized recommendation sets can be used as a coordination mechanism for distributed resources when the platform has neither perfect knowledge nor control over suppliers. We use as a case study, a ride-sharing platform, in which (1) demand requests are riders who request a ride from one point of interest (origin) to another (destination), (2) suppliers are drivers who travel from their origins to their destinations and may be willing to offer a ride if it meets their criteria; and (3) the platform receives requests from riders and facilitates matches by recommending a personalized set of ride requests to each driver.

Our goals are to (1) compare the performance of our proposed hierarchical model with existing platform matching methods, namely, centralized, decentralized, and many-to-many stable matching mechanisms; (2) quantify the platform's value of providing choices in on-demand environments; and (3) determine under what scenarios it is better to recommend only one alternative, a few alternatives, or all alternatives, and to investigate when it is useful for the recommendations to contain overlapping alternatives. To achieve these goals, our computational experiment consists of four phases represented in Figure 2, which together captures a platform that can only partially estimate the selection outcomes of suppliers when it makes recommendations.

In phase one, data generation, we describe how a ride-sharing application is used to generate a set of computational experiments. Factors vary the distribution of origin-destination (O-D) points for riders and drivers, rejection penalties, expected thresholds, and platform estimation errors, which account for the platform's lack of visibility into drivers' true selection outcomes. In the second phase, a set of optimization engines input expected values of parameters to determine personalized recommendations for a set of drivers. We consider four optimization engines, namely a centralized, decentralized, many-to-many stable matching, and our hierarchical approach. Third, in the simulation environment, the drivers' selections (i.e., the reaction of drivers to the platform's recommendations) are simulated. Finally, in phase four, we compare the platform's performances when the four recommendation models are used to determine recommendations. We report the platform's objective function values attained after the drivers' actual selections, as a function of menu size

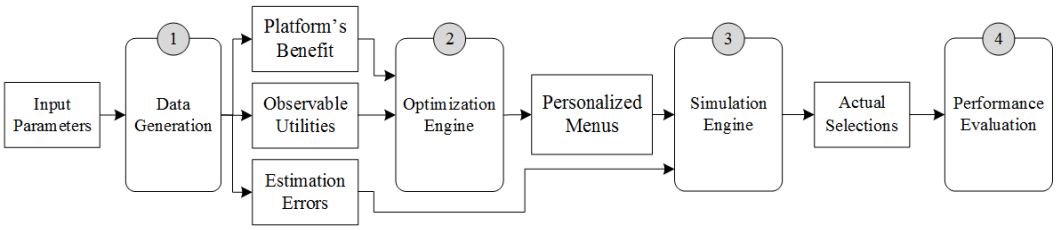

Figure 2 Phases of the Computational Study

and experimental factors. We also report on the value of choice for the drivers and riders. We implement all phases in MATLAB 8.5 (R2015a) programming language, using Gurobi Optimization 7.5.2. as the solver, on a quad-core 1.9 GHz CPU with 32GB RAM.

6.1. Data Generation Based on a Ride-Sharing Application

The specific ride-sharing platform we consider is as follows. First, we consider the static version, meaning the platform solves the recommendation problem periodically and we focus on one period only. Solving the snap shot version, while a simplification, allows us to accomplish our three goals. Second, we assume that n riders and m drivers exist in the system, and $n = m = 10$. All riders simultaneously request rides by announcing their trip's O-D pairs. However, drivers' willingness to provide a ride depends on the characteristics of the ride request. The drivers in the platform only allow the platform to observe their origin and do not disclose their destination. Instead, the platform estimates drivers' destinations based on past history. This results in the platform able to perfectly determine c_{ij} values, but only partially estimate drivers' utilities for a ride, ν_{ij}. Therefore, the optimization engines use ν_{ij}, whereas, we incorporate uncertainty about drivers' destinations in the simulation phase. Drivers have the capacity to provide at most one ride (i.e., one pickup and delivery of a rider) per trip, and no ride transfers are allowed. The platform has to recommend a set (a menu) of ride requests to drivers. Drivers can choose at most one alternative. All drivers are recommended and have the option to choose the no-choice alternative (i.e., decide not to participate in the ride-sharing platform by not accepting any of the ride requests recommended). We also set $a_i = n = 10, \forall i \in A$, which means a single alternative can be recommended to all drivers in the system. The collision penalty, $d_i = 1.5, \forall i \in R$. We test all integer values of the menu size θ in the range of $[1, 10]$. The first two columns of Table 1 provide the five factors and their level values in our design of experiments. We conduct a full factorial design, which results in 288 cases. We generate 1440 instances by

replicating each case 5 times, in which a replication is a specific outcome generated from the given probability distributions.

6.1.1. Expected Utility Estimation We assume the ride-sharing platform operates in an unit square zone; that is, all drivers' and riders' O-D are specified by a 2-tuple (x, y) containing coordinates in two-dimensional Euclidean space in $[0,1] \times [0,1]$. The trip of rider i is denoted by the rider's origin and destination O_i, and D_i, respectively. Similarly, O_j and $\hat{D}_j$, denote the origin and expected destination of driver j's trip, respectively. In the context of ride-sharing, the distribution of O-D points may vary. Riders destinations may concentrate in a particular region (e.g., to be dropped off at a football stadium); riders origins may also concentrate (e.g., most of the weekend night rides originate from certain downtown areas). Similarly, drivers' destinations and origins may be concentrated due to commuting patterns. We generate O-D pairs under three scenarios, in which the first letter determines the distribution of the origin point and the second letter the distribution of the destination point. In all cases, we randomly generate x and y coordinates for points independently. **Uniform-Uniform (UU):** A trip is equally likely to start and end at any point inside the unit square, i.e., $x \sim U(0,1)$ and $y \sim U(0,1)$. **Uniform-Centered (UC):** A trip is equally likely to start at any point inside the unit square, but it is more likely to end at a point near the center. The origin's coordinates are $x \sim U(0,1)$ and $y \sim U(0,1)$. The destination is normally distributed with $x \sim N(\mu = 0.5, \sigma^2 = 0.12^2)$ and $y \sim N(\mu = 0.5, \sigma^2 = 0.12^2)$. **Centered-Uniform (CU):** Origins are concentrated in the center, but destinations are spread uniformly across the zone. The origin's coordinates are normally distributed with $\mu = 0.5$, and $\sigma = 0.12$. However, the destination's coordinates are uniformly distributed as $x \sim U(0,1)$ and $y \sim U(0,1)$. We consider these three scenarios for riders and for drivers, resulting in nine cases.

As defined in Section 3, the platform's benefit of having request i be fulfilled by driver j is captured by c_{ij}. The platform's benefit is a linear function of the Euclidean distance from driver j's origin to rider i's origin, as shown in (18). The value c_{ij} is deterministically known to the platform and prioritizes matches with a driver with an origin close to the rider's origin. In the unit square, the distance between the origins of any driver j and any rider i is at most $\sqrt{2}$ and at least zero.

$$c_{ij} = \sqrt{2} - |O_i - O_j| \quad \forall i \in R, \forall j \in S \tag{18}$$

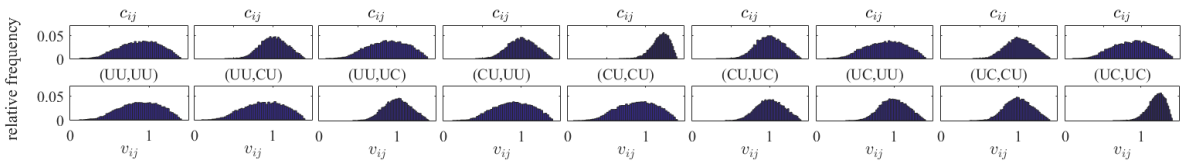

Figure 3 **Distributions of platform benefit values, c_{ij}, and estimated drivers benefit values, ν_{ij}, for the nine O-D pair combinations.**

The drivers' goals are modeled as minimizing their empty travel distances after dropping off the rider at the rider's destination. We assume a driver's destination can be estimated based on past travel history, which the platform captures in the estimated destination $\hat{D}_j$. We define the expected utility of driver j for ride request i in (19).

$$\nu_{ij} = \sqrt{2} - |D_i - \hat{D}_j| \quad \forall i \in R, \forall j \in S \tag{19}$$

Figure 3 provides the relative frequencies of the 100 elements of c_{ij} and ν_{ij} for each of the nine O-D pair distribution combinations. This generation approach provides a variety of distributions for utility matrices ν_{ij} and c_{ij}, which represent varying alignment between the platform and drivers utilities. For example, if both riders and drivers O-D pairs are generated by a UC scenario, the destinations of both groups occur around the center, but their origins are spread out. Thus, the ν_{ij} values have less variance than the c_{ij} values (see Figure 3 right-most distributions). Our data also captures a group of drivers interested in similar rides due to similarities in their individual planned trips. To test this, we calculate the Pearson correlation coefficients of ν_{ij} values between all pairwise combinations of drivers, and similarly of c_{ij} values between all pairwise combinations of riders. In all instances, there exists at least a pair of drivers who have similar preferences (i.e., the maximum pairwise ν_{ij} correlation value is close to 1) (Mofidi 2018).

6.1.2. Estimation Errors Driver j's true utility for ride request i (i.e., u_{ij}) is measured as the distance between the driver's and rider's true destinations. Although O_i, D_i, and O_j are known to the platform, D_j is unknown. We define driver j's true utility as $u_{ij} = \sqrt{2} - |D_i - D_j|$, where $D_j = \hat{D}_j + \epsilon_j$. We specify ϵ_j as a 2-tuple $(x_{\epsilon_j}, y_{\epsilon_j})$ vector, and assume x_{ϵ_j} and y_{ϵ_j} are generated independently with mean zero and equal variances. Thus, u_{ij} is decomposed into an estimated value observable to the platform and an error term, i.e., u_{ij} is the vector sum of ν_{ij} and ϵ_j. As described in Table 1, we consider two variance levels for the destination error. This approach models the platform's uncertainty of supplier selection

via imperfect estimate of drivers' destinations. This is a surrogate measure to account for the platform's lack of visibility into drivers' true selection outcomes. Alternatively, the error term ϵ_j can represent other exogenous factors influencing drivers' utilities not captured by the platform's historical data.

6.1.3. No-Choice Alternative Utilities The no-choice alternative corresponds to a driver deciding not to select any of the recommended ride requests. If the no-choice alternative is selected by a driver, the platform receives no benefit, and $c_{0j} = 0, \forall j \in S$. However, the no-choice alternative can have value for a driver, as it represents a driver who would rather go directly to their planned destination than participate in the ride-sharing platform. A driver will select the no-choice alternative only if the utility is larger than all recommended ride requests. To be consistent with Section 6.1.1, we define the utility of no-choice as a function of the maximum distance a driver is willing to travel empty after dropping off a rider at their destination. Beyond this threshold, the driver would not offer a ride. We model driver true utility of the no-choice option as only partially estimated by the platform. To do so, we generate an expected utility for no-choice (i.e., $\nu_{0j)}$) based on a driver's past no-choice utilities and calculated in (20).

$$\nu_{0j} = \sqrt{2} - \delta_j \quad \forall j \in S \tag{20}$$

where δ_j is the expected empty travel distance threshold of driver j, which we refer to as the expected threshold throughout this study. To differentiate inflexible drivers with flexible ones, Table 1 provides low and high expected threshold values. We simulate an error in this estimate, such that driver j's utility for the no-choice alternative is $u_{0j} = \nu_{0j} + e_j$. A realization of the true utility is simulated by randomly sampling e_j from a normal distribution with two levels given in Table 1.

6.2. Optimization Engines

We analyze four techniques to generate recommendation sets: a centralized (*ctr*), decentralized (*dec*), many-to-many stable matching (*msm*), and our proposed hierarchical model (*hrc*). These are all deterministic optimization approaches, and the expected value of all parameters are used as inputs to the optimization models. The *hrc* recommendation is determined by solving the single-level optimization problem, resulting from the reformulation technique. The *dec* recommendation shows all available alternatives to all drivers.

This does not require optimization, and is also equivalent to the *hrc* recommendation with a menu size of $\theta = 10$. The *ctr* recommendation solves the traditional assignment problem using only platform utilities (i.e., c_{ij} values, but ignoring ν_{ij} values). The *ctr* always recommends each driver exactly one request and no two drivers are recommended the same ride request. While *ctr* ignores drivers' no-choice preferences, this information can be captured in a centralized optimization engine that recommends a single ride request to all drivers by solving *hrc* with $\theta = 1$.

For *msm*, we solve the many-to-many stable matching model with a maximum cardinality objective function. We solve the linear programming representation, which enforces the pairwise stability criteria via blocking pair constraints, as defined in Roth et al. (1993). Sotomayor (1999) highlights the usefulness of pairwise-stable matching in competitive labor markets and proves there always exists a pairwise-stable matching. In many-to-many matching, a match is a recommendation that forms a connection between a rider to a driver. So, a one-to-one match means a ride request is only recommended to a single driver and the drivers' menus contain one recommended ride request. A many-to-many matching denotes recommending a subset of ride requests to a subset of drivers. For example, a 2-to-2 matching means each ride request is recommended to two drivers and each driver's menu contains two possible ride requests. Stability means that no other ride requests other than the current recommendation set exists by which both the platform and the drivers can individually improve their objective function values simultaneously. The pairwise stability constraints ensure that either alternative i is recommended to supplier j, or another alternative $i^{'}$ is recommended to supplier j with a corresponding supplier utility greater than or equal to the utility of i, or the platform recommends the alternative i to supplier $j^{'}$ that has a benefit of greater than or equal to recommending it to supplier j.

To investigate the value of choice, we force *hrc* and *msm* models to show each driver j a menu size θ incremented by one unit from 1 to 10. The case of a menu size of 10 does not require optimization because the solution is to show each driver all requests. Therefore, for each of the 1440 instances, we solve 19 optimization problems (i.e, 1 for *ctr*, and 9 each for the *hrc* and *msm* models).

6.3. Simulation Environment and Performance Measures

For a given recommendation matrix, we simulate drivers' selections and then compare the recommendation models' performance when the platform has neither control nor perfect

knowledge about the drivers' utilities. The drivers are assumed to be rational utilitarian individuals. Thus, given the recommendation set of each driver (i.e., x_{ij}), and the driver's true utility vector (i.e., u_{ij}), the actual selection of driver j (denoted by $\hat{y}_{ij}$) is the alternative in the menu of driver j that has maximum cardinal utility, i.e., $i_{max} = \arg\max_{i \in A}(u_{ij} \odot x_{ij}) \, \forall j \in S$; and $\hat{y}_{ij} = 1$ if $i = i_{max}$; 0 o.w. $\odot$ is the element-wise multiplication. After extracting the drivers' actual selection matrix $\hat{Y} = [\hat{y_{ij}}]$, the actual collision vector $\hat{z}_i$, and the actual rejection vector $\hat{w}_i$ are the minimum values that enforces constraints (5), and (6), respectively, when y_{ij} is replaced with $\hat{y_{ij}}$. We calculate the platform's simulated objective function value (φ) by calculating the objective function given in expression (1) using the values of $\hat{y}_{ij}$, $\hat{z}_i$, and $\hat{w}_i$. For each instance and recommendation engine, we perform 100 simulation replications by generating 100 error values. We use φ to refer to the average platform's objective function value over the 100 simulation replications. Because φ can contain positive or negative values, defining relative performance between the different methods is not possible. Thus, we rescale the objective function values (using unity-based normalization) to a $[0, 1]$ range by defining each method's platform's achievement ratio (π^{meth}) as in (21). In our computational study, an upper bound to φ is $10\sqrt{2}$ when all drivers are matched with maximum $c_{ij} = \sqrt{2}$ values. This upper bound is not always achievable because of the way O-D pairs are generated. A lower bound has $\varphi = -27$, which occurs when the platform collects zero benefit and incurs maximum penalties of -1.5 for 9 collisions and 9 rejections.

$$\pi^{meth} = \frac{(\varphi - min(\varphi))}{(max(\varphi) - min(\varphi)} = \frac{(\varphi + 27)}{(10\sqrt{2} + 27)} \, \forall meth = \{hrc, \quad ctr, \quad msm, \quad dec\} \tag{21}$$

7. Results and Insights

In this section, we compare the platform's achievement ratios (π) of the *hrc* recommendation method with the *ctr*, *dec*, and *msm* methods, and analyze performance as a function of varying control factors and menu sizes. Figure 4 displays each recommendation model's achievement ratio (π) averaged over all 1440 instances and grouped by menu sizes, $\theta = 1, 2, \ldots, 10$. Although the achievement ratios are not a function of menu size θ for neither *ctr* nor *dec*, we draw a constant line parallel to the x-axis for easier visual comparison. On average, *hrc* outperforms the other models when $\theta \geq 4$. Also, on average, $\theta = 1$ achieves the highest performance for both *hrc* and *msm* (however, as we subsequently explore, the best performance does not always occur with $\theta = 1$). Moreover, for a menu size of $\theta = 1$,

the achievement ratios vary whether a *hrc*, *ctr*, or *msm* is used. Although all recommend drivers a single ride request, what is recommended varies due to each approach solving a different optimization model. For the case with $\theta = 10$, the recommendation set is to show all drivers all requests, which is the same recommendation for *hrc*, *msm*, and *dec*.

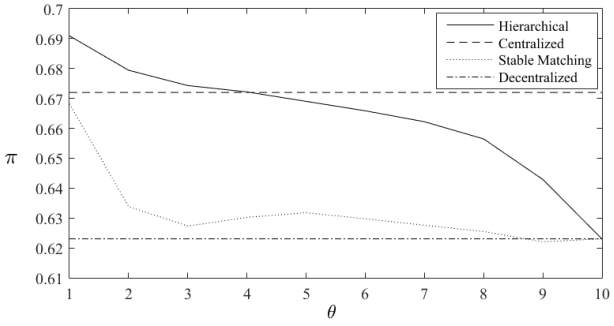

Figure 4 **Achievement ratios (π) averaged over all 1440 instances, as a function of menu size θ.**

7.1. Impact of Factors on Platform Performance and Value of Choice

In this section we perform a similar analysis, but broken down by factor levels, namely, expected threshold, destination error, O-D pair distributions, and rejection penalties.

By categorizing the instances by low and high expected thresholds, Figure 5 displays two distinct patterns for the achievement ratios. Therefore, the value of choice is influenced by drivers' no-choice utilities. When the expected threshold is low drivers are inflexible; the average platform objective function of *hrc* is at its maximum for the menu size of $\theta = 7$ (Figure 5a, left). With low expected thresholds, the platform gets value by providing more choices, because choices increase the chance of enticing inflexible drivers to participate. However, high expected thresholds mean more flexible drivers, who are more likely to participate even if the platform's recommendations are less aligned with their preferences, i.e., riders need to be dropped off longer distances from their destination. With a high expected threshold, the best average performance for *hrc* and *msm* occurs when $\theta = 1$. As the number of choices increases, flexible drivers select requests with high driver utility, but which hurt the platform's objective. Hence, by providing more than two choices, the platform is sacrificing its benefit. The discrimination between when choice is valuable is further supported by comparing the outcomes of the *dec* and *ctr* methods. In the presence of a low expected threshold (Figure 5a, left), *dec* outperforms *ctr*. However, the opposite result occurs when the expected threshold increases (i.e., when drivers have low no-choice

utilities). On average, higher values of π can be obtained when the expected threshold is high, regardless of recommendation method.

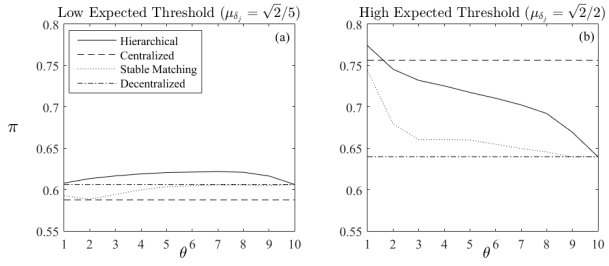

Figure 5 **Achievement ratios (π) as a function of menu size θ, averaged over given expected thresholds: Figure 5a. (left) low expected threshold; Figure 5b. (right) high expected threshold.**

Figure 6 visualizes the pairwise comparison between achievement ratios (π) of *hrc* (horizontal axis) and *ctr* (vertical axis) models across varying menu sizes θ. This analysis supports the previous findings, but at a finer comparison. A dot in Figure 6 is a pairwise comparison between one of the 1440 instances' achievement ratios of *hrc* versus *ctr*. A dot on the diagonal line indicates that for a particular instance, the two models achieve the same performance. A dot below the diagonal line indicates *hrc* outperforms *ctr*. Dark dots identifies an instance with a high expected threshold; a light dot an instance with a low expected threshold. In Figure 6, *hrc* results in better achievement ratios when the menu size θ is small. For a single-alternative menu ($\theta = 1$), *hrc* has the highest performance rate (the portion of the dots below the diagonal is highest). Focusing on the colors, when an observation has a low expected threshold (light dots), *hrc* outperforms *ctr* most of the times across all menu sizes. However, when the expected threshold is high (dark dots), the improved performance of *hrc* decreases as the menu size θ increases (the dark dots transition from below the diagonal to above). In the presence of low thresholds, *hrc* accommodates the inflexible drivers' preferences and provides better recommendations than *ctr*. However, this advantage only works up to a point. As the menu size θ increases, accommodating drivers' preferences comes at a cost of degrading platform performance. To achieve higher benefit for the platform, on average, the menu size should be set with lower values when the expected threshold is high and with larger values when the expected threshold is low.

Figure 7 explores the impact of the destination error and the expected threshold factors on achievement ratios. The destination error captures the platform's uncertainty over

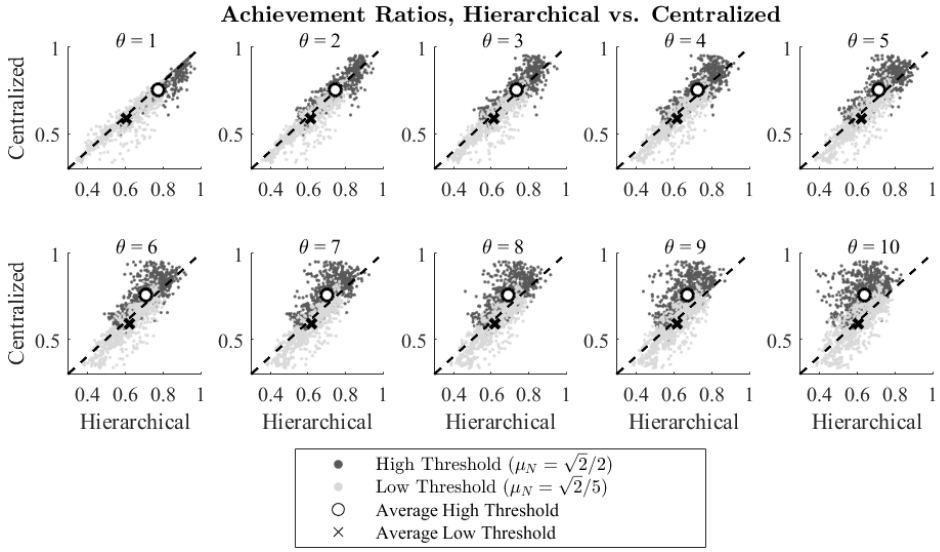

Figure 6 **Pairwise comparison between achievement ratios (π) of hrc and ctr (vertical axis) models across varying menu sizes θ, for high (dark color) and low (light color) expected threshold levels.**

drivers' selections and is varied at two levels, low and high. Comparing the left and right plots in Figure 7, the achievement ratio is higher, on average, when the error is low (the platform is less uncertain about the drivers' selections). However, when this uncertainty is high and the expected threshold is low (Figure 7d, bottom right), on average the value of π increases as the menu size θ increases. Also, *dec* outperforms *ctr*. Thus, choice has value in the presence of high driver selection uncertainty and inflexible drivers. When the platform has only a vague idea about drivers' selections, the more alternatives recommended, the higher the likelihood of a selection. Whereas, discretion is less beneficial when the platform can predict drivers' selections with more precision. In this case (Figures 7, a and c), *hrc*'s achievement ratio, averaged across low destination instances, has a strictly-decreasing behavior as a function of menu size, regardless of threshold levels. Also, *ctr* has higher π values than *dec*. The platform obtains higher objective function values (on average) when more knowledge about the drivers' destinations are available. On the other hand, with high uncertainty about inflexible drivers' selections, the platform's objective function can be improved if more choices are provided to inflexible drivers (see Figures 7, b and d). That is, the platform's lack of knowledge over inflexible drivers selections can be compensated by providing more choices.

Figure 8 explores the impact of requests' and drivers' O-D pair distributions on the platform's performance. Choice is not useful when the requests' and drivers' O-D distribution

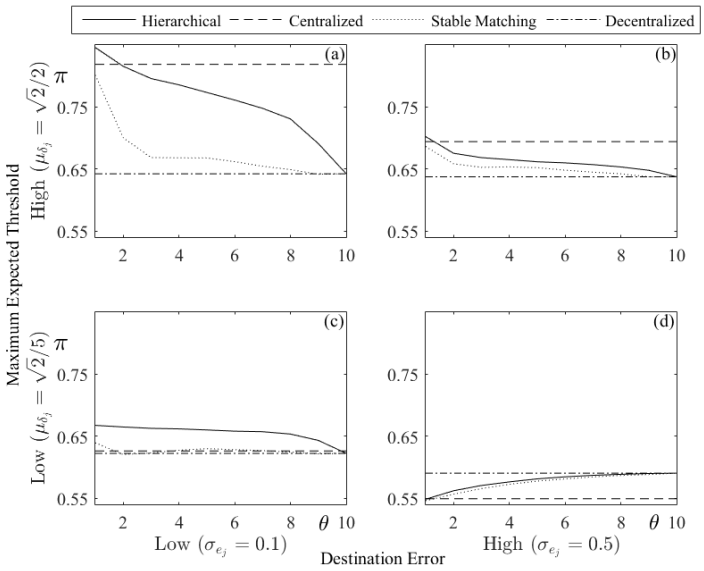

Figure 7 **Achievement ratios (π) as a function of menu size θ, averaged by expected thresholds and destination errors.**

are both UC (see Figure 8c). A UC-UC scenario results in the steepest decrease for *hrc* and *msm* as a function of menu size. It's achievement ratio with $\theta = 1$ has the highest π value on average among all nine scenarios. This occurs because the destinations of both drivers and requests are clustered in the center, and the drivers' utility values have low variance (see Figure 3, bottom right). Whereas, requests' origins and drivers' are uniformly scattered in UC-UC. Thus, when the platform's benefit values have higher variance than the drivers' expected utilities, offering additional choices, on average, jeopardizes significantly the platform's benefit. Because drivers are utility maximizers, by offering more choices, they improve their utility, although only by a little. While, the platform loses much more utility. The reverse happens when requests' and drivers' O-D distributions are (CU, CU) (see Figure 8e). On average, *hrc* is superior, and achieves the highest average π for $\theta = 7$. Also, *dec* outperforms *ctr*. For this scenario, the values of c_{ij} are very close to each other (see Figure 3), resulting in the platform being almost indifferent between the assignment of drivers to requests. On the other hand, the large range of ν_{ij} values means drivers have strong and variable preferences. Similar phenomenon occurs in two other scenarios, in which riders requests' and drivers' O-D distributions are (1) CU and UU (Figure 8h); and are (2) UU and CU (Figure 8d). In both scenarios, the variance of drivers' utility values is higher than the variance of the platform's benefit values. In these cases, providing choice

is beneficial because the increase in likelihood of participation outweighs the reduced platform benefit of a match. In general, the menu size depends on the O-D pair distributions of riders and drivers in the region the platform operates. Providing choice is beneficial for the platform, on average, when drivers' destinations are scattered. On the other hand, when riders are distributed across the region, the platform needs to act more like a dispatcher, recommending a small number of alternatives to increase its performance.

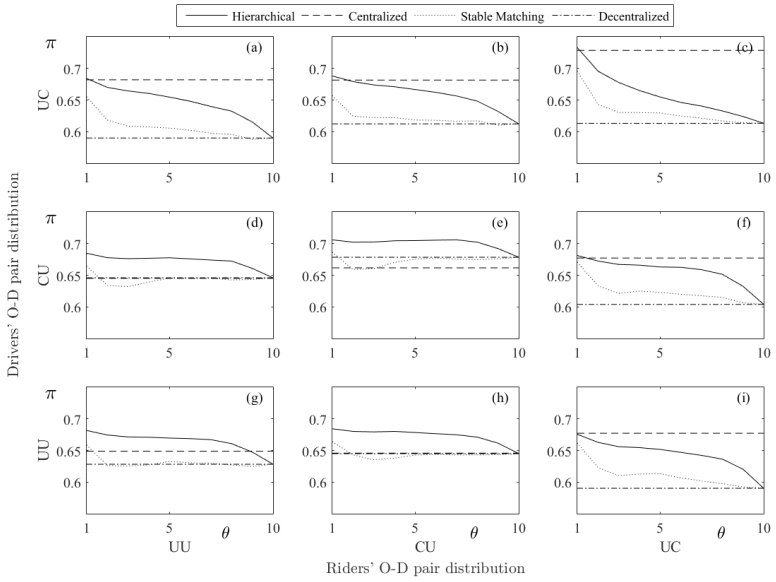

Figure 8 **Achievement ratios (π) as a function of menu size θ, averaged by O-D pair distributions.**

Figure 9 explores the impact of the rejection penalty value. As the menu size of *hrc* increases, average π values decrease across different penalty levels. Moreover, the value of π is highly influenced by the rejection values (i.e., higher rejection penalties lead to decreasing objective function values), regardless of recommendation engine.

7.2. Best Menu Size

Table 1 provides the relative frequency of the best menu size θ^* from *hrc*. For each instance, we select the best menu size θ^* as the menu size achieving *hrc*'s maximum achievement ratio π, out of all θ values 1 to 10. This information is then followed by a set of columns that provide the percentage of instances in which: (a) a single choice was best (i.e., $\theta^* = 1$), (b) providing recommendations for suppliers to choose from was best (i.e., $2 \leq \theta^* \leq 9$), or (c) providing all requests to all suppliers (i.e., $\theta^* = 10$) was best. Among the 1440 instances, the best menu size is equal to 1 in 54.9% of instances and equal to 10 in 11% of the instances.

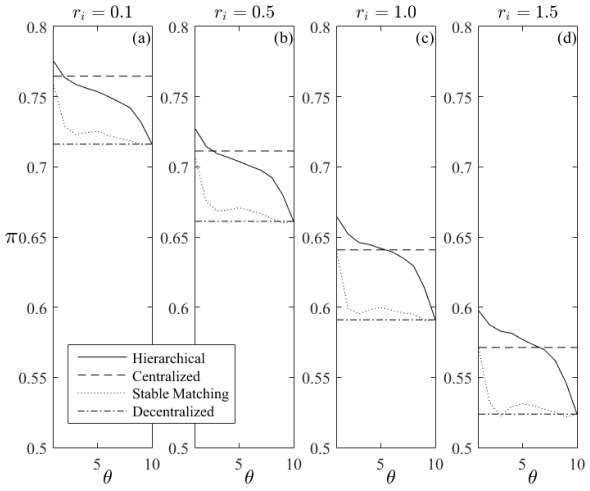

Figure 9 **Achievement ratios (π) as a function of menu size θ, averaged by rejection penalty values.**

In 34% of the instances, the best menu size is between 2 and 9, and in these instances providing choices and recommending alternatives to more than one driver is beneficial. Factor values influence the distribution of best menu sizes. The factor with the highest percentage of best menu size between 2 and 9 is when the expected threshold is low. Thus, when drivers are less flexible and more picky, providing a menu size between 2 and 9 is most beneficial. Also, a relatively high percentage of a best menu size being between 2 and 9 occurs for the case of UC, CU, as well as CU, CU. For UC, CU both ν_{ij} values and c_{ij} values are concentrated. For CU, CU, the platform's c_{ij} values are concentrated, but the ν_{ij} values are variable.

7.3. Quantifying Methods' Performance

Next, we measure the percent difference, Δ, between the achievement ratios of *hrc* with the best menu size θ^* and the other models. Similar to *hrc*, we define *msm*'s best menu size θ^{s*} as the menu size achieving the highest achievement ratios and use this for comparisons. Table 2 provides the achievement ratios of each method. For *hrc* and *msm*, we report the achievement ratios when their corresponding best menu size is used, i.e., $\pi^{hrc}_{\theta^*}$ and $\pi^{msm}_{\theta^{s*}}$. Next, for each level, we report the percent of instances in which *hrc* achieves better performance than other approaches; that is, the percentage of instances, in which $\pi^{hrc}_{\theta^*} \geq \pi^{ctr}$, and in which $\pi^{hrc}_{\theta^*} \geq \pi^{msm}_{\theta^{s*}}$. We did not report the difference between *hrc* and *dec*, because in all instances, *hrc* achieves a π value equal or higher than for *dec*. With θ^*, *hrc* outperforms *ctr* and *msm* in 87.3% and 82.4% of the times, respectively. Finally, Table 2

Table 1 The distribution of the best hrc menu size θ^*, broken down by factor value. Followed by the percentage when the best menu size is 1, between 2 and 9, and 10.

Factor	Levels and Values	1	2	3	4	5	6	7	8	9	10	1	2-9	10
Average over all 1140 instances		54.9	4.3	3.0	3.5	2.9	4.0	4.9	5.2	6.1	11.0	54.9	34.0	11.0
expected threshold	low: $\delta_j =\sim N(\mu_{\delta_j}=\sqrt{2}/5, \sigma^2_{\delta_j}=0.1^2)$	21.7	5.3	4.6	5.7	4.4	6.1	8.9	9.6	11.7	22.1	21.7	56.3	22.1
expected threshold	high: $\delta_j =\sim N(\mu_{\delta_j}=\sqrt{2}/2, \sigma^2_{\delta_j}=0.1^2)$	88.2	3.3	1.4	1.4	1.4	0.0	1.9	1.0	0.8	0.6	88.2	11.3	0.6
rejection penalty	$r_i = 0.1$	58.1	4.4	4.2	3.3	1.9	3.9	3.6	5.6	6.7	8.3	58.1	33.6	8.3
rejection penalty	$r_i = 0.5$	57.5	4.2	3.1	2.2	3.1	3.1	5.0	5.3	5.6	11.1	57.5	31.4	11.1
rejection penalty	$r_i = 1.0$	54.7	5.6	1.9	3.6	2.5	3.6	5.0	5.6	6.7	10.8	54.7	34.4	10.8
rejection penalty	$r_i = 1.5$	49.4	3.1	2.8	5.0	4.2	5.6	6.1	4.4	5.6	13.9	49.4	36.7	13.9
destination error	low: $\epsilon_j =\sim N(\mu_{\epsilon_j}=0, \sigma^2_{\epsilon_j}=0.1^2)$	63.3	6.5	4.4	5.4	3.3	3.9	5.1	3.6	3.3	1.0	63.3	35.7	1.0
destination error	high: $\epsilon_j =\sim N(\mu_{\epsilon_j}=0, \sigma^2_{\epsilon_j}=0.5^2)$	46.5	2.1	1.5	1.7	2.5	4.2	4.7	6.8	8.9	21.1	46.5	32.4	21.1
no-choice error	low: $e_j =\sim N(\mu_{e_j}=0, \sigma^2_{e_j}=0.1^2)$	52.6	2.9	2.6	2.9	2.9	3.3	4.9	5.6	6.8	15.4	52.6	31.9	15.4
no-choice error	high: $e_j =\sim N(\mu_{e_j}=0, \sigma^2_{e_j}=0.1^2)$	57.2	5.7	3.3	4.2	2.9	4.7	5.0	4.9	5.4	6.7	57.2	36.1	6.7
riders' O-D, drivers' O-D distributions	UU, UU	51.3	5.6	3.1	2.5	4.4	1.9	5.6	6.3	8.1	11.3	51.3	37.5	11.3
riders' O-D, drivers' O-D distributions	UU, UC	54.4	6.9	3.8	5.6	1.9	3.8	5.0	1.9	6.3	10.6	54.4	35.0	10.6
riders' O-D, drivers' O-D distributions	UU, CU	52.5	4.4	1.3	1.9	3.8	2.5	4.4	5.0	10.6	13.8	52.5	33.8	13.8
riders' O-D, drivers' O-D distributions	UC, UU	52.5	3.8	4.4	5.0	4.4	5.6	3.8	9.4	5.6	5.6	52.5	41.9	5.6
riders' O-D, drivers' O-D distributions	UC, UC	84.4	1.9	0.6	0.0	3.8	1.3	1.9	3.1	2.5	0.6	84.4	15.0	0.6
riders' O-D, drivers' O-D distributions	UC, CU	51.9	4.4	4.4	4.4	1.3	8.8	7.5	6.3	5.0	6.3	51.9	41.9	6.3
riders' O-D, drivers' O-D distributions	CU, UU	47.5	5.6	3.8	4.4	1.3	5.6	5.0	6.9	3.8	16.3	47.5	36.3	16.3
riders' O-D, drivers' O-D distributions	CU, UC	59.4	3.1	3.8	5.0	1.3	1.9	3.1	1.9	4.4	16.3	59.4	24.4	16.3
riders' O-D, drivers' O-D distributions	CU, CU	40.6	3.1	1.9	3.1	4.4	5.0	8.1	6.3	8.8	18.8	40.6	40.6	18.8

reports the maximum, mean, and standard deviation of the percent difference in π between hrc and other methods, across all levels of the control factors. On average, hrc improves upon msm, ctr, dec, and hrc with a single recommendation, 5.2%, 2.6%, 10.7%, and 2.6%, respectively, but achieves improvements as high as 40.8% over msm and 23.6% over ctr.

Table 2 also illustrates that environmental factors influence platform performance. Regardless of methods, drivers who are more flexible (which have higher expected thresholds) result in higher platform performance compared with less flexible drivers (which have low expected thresholds). Rejection penalty values and estimation errors are both negatively correlated with platform performance. Finally, platform performance increases when both the drivers and the riders have a centralized origin or destination, and the drivers and riders O-D patterns match (i.e., either both CU, CU, or both UC, UC).

7.4. The Value of Choice to Drivers and Riders

Here, we assess the value of choice for the two other entities in the system (i.e., drivers and riders). As the number of choices increases, the lower level objective function will increase. However, because of duplicates and collisions, not all drivers will be ultimately matched with the request they selected. Hence, the lower-level objective function is not useful to

Table 2 Design factors, levels and values, followed by each method's associated achievement ratios (using best menu sizes), number of instances in which hrc achieves equal or better performance, as well as the maximum, average, and standard deviation of the percent difference in achievement ratios between hrc and other methods.

								$\Delta(\pi^{hrc}_{\theta^*}, \pi^{msm}_{\theta^{**}})$			$\Delta(\pi^{hrc}_{\theta^*}, \pi^{ctr})$			$\Delta(\pi^{hrc}_{\theta^*}, \pi^{dec})$			$\Delta(\pi^{hrc}_{\theta^*}, \pi^{hrc}_{\theta=1})$		
Factor	Levels and Values	$\pi^{hrc}_{\theta^*}$	$\pi^{msm}_{\theta^{**}}$	$\pi^{ctr}_{\theta^{**}}$	$\pi^{dec}_{\theta^{**}}$	$\pi^{hier}_{\theta^*} \geq \pi^{ctr}$	$\pi^{hier}_{\theta^*} \geq \pi^{msm}_{\theta^{**}}$	max	mean	std	max	mean	std	max	mean	std	max	mean	std
Average over all 1140 instances		70.6	68.6	67.2	62.3	87.3	82.4	40.8	5.2	7.4	23.6	2.6	3.8	58.4	10.7	10.5	29.7	2.6	5.0
expected threshold	low: $\delta_j =\sim N(\mu_{\delta_j} = \sqrt{2}/5, \sigma^2_{\delta_j} = 0.1^2)$	63.7	62.6	58.8	60.6	83.6	83.2	40.8	8.0	8.9	19.5	1.6	2.7	32.2	4.6	5.7	29.7	5.1	6.1
expected threshold	high: $\delta_j =\sim N(\mu_{\delta_j} = \sqrt{2}/2, \sigma^2_{\delta_j} = 0.1^2)$	77.5	74.6	75.6	64.0	91.0	81.5	27.7	2.4	3.9	23.6	3.6	4.3	58.4	16.8	10.8	9.3	0.2	0.9
rejection penalty	$r_i = 0.1$	78.3	76.7	76.5	71.6	84.2	78.9	15.0	2.4	3.5	17.7	1.9	2.9	31.4	8.0	7.5	8.7	1.1	1.9
rejection penalty	$r_i = 0.5$	73.9	72.0	71.1	66.1	87.5	84.4	21.5	3.8	4.9	17.8	2.4	3.2	40.4	9.6	9.0	13.6	1.8	3.0
rejection penalty	$r_i = 1.0$	68.1	65.8	64.1	59.1	89.4	85.0	35.7	6.1	7.5	20.3	3.0	4.0	45.8	11.8	10.9	20.3	2.9	4.8
rejection penalty	$r_i = 1.5$	62.1	59.9	57.1	52.4	88.1	81.1	40.8	8.6	10.3	23.6	3.1	4.6	58.4	13.4	13.2	29.7	4.8	7.5
destination error	low: $\epsilon_j =\sim N(\mu_{\epsilon_j} = 0, \sigma^2_{\epsilon_j} = 0.1^2)$	76.3	73.1	72.2	63.2	86.0	85.8	40.8	5.6	8.1	23.6	4.0	4.4	58.4	16.2	11.0	15.5	1.0	2.1
destination error	high: $\epsilon_j =\sim N(\mu_{\epsilon_j} = 0, \sigma^2_{\epsilon_j} = 0.5^2)$	64.9	64.1	62.1	61.4	88.6	78.9	34.1	4.8	6.7	14.6	1.1	2.2	28.4	5.1	6.2	29.7	4.3	6.3
no-choice error	low: $e_j =\sim N(\mu_{e_j} = 0, \sigma^2_{e_j} = 0.1^2)$	72.1	69.7	66.8	62.7	96.9	85.3	40.8	8.1	8.7	23.6	3.1	4.3	58.4	11.6	11.9	29.7	3.4	6.1
no-choice error	high: $e_j =\sim N(\mu_{e_j} = 0, \sigma^2_{e_j} = 0.1^2)$	69.1	67.5	67.6	61.9	77.6	79.4	27.3	2.3	4.3	22.1	2.1	3.1	44.9	9.7	8.9	17.5	1.9	3.4
riders' O-D, drivers' O-D distributions	UU, UU	69.8	67.9	64.9	62.8	92.5	81.3	39.5	7.4	8.3	19.2	2.4	3.4	37.0	9.0	8.6	25.2	2.8	5.1
riders' O-D, drivers' O-D distributions	UU, UC	70.2	67.3	68.2	59.0	80.6	86.9	28.9	3.3	6.4	20.3	3.7	4.2	58.4	14.4	14.0	29.6	3.1	5.8
riders' O-D, drivers' O-D distributions	UU, CU	70.2	68.9	64.5	64.6	95.6	76.9	35.7	8.5	7.7	13.5	1.6	3.3	34.8	7.2	8.1	25.7	2.9	5.3
riders' O-D, drivers' O-D distributions	UC, UU	68.7	67.1	67.7	59.0	75.0	76.9	21.7	1.6	4.4	18.3	2.1	3.3	41.3	12.7	10.8	14.5	2.1	3.3
riders' O-D, drivers' O-D distributions	UC, UC	73.6	70.1	72.8	61.3	84.4	94.4	21.3	1.2	3.4	23.6	4.4	4.5	37.4	15.5	9.8	10.5	0.6	2.0
riders' O-D, drivers' O-D distributions	UC, CU	69.2	68.0	67.7	60.4	78.1	70.6	25.4	2.4	4.9	15.0	1.6	3.0	43.2	11.6	10.1	16.7	2.0	3.4
riders' O-D, drivers' O-D distributions	CU, UU	70.2	68.6	64.6	64.5	98.1	83.8	40.8	8.5	8.4	16.0	2.0	3.2	38.8	7.4	7.8	26.0	3.0	5.3
riders' O-D, drivers' O-D distributions	CU, UC	70.9	68.0	68.2	61.2	84.4	88.1	29.3	4.3	6.9	19.5	3.8	4.3	50.3	12.4	12.3	29.7	3.6	6.5
riders' O-D, drivers' O-D distributions	CU, CU	72.8	71.5	66.2	67.8	96.9	82.5	34.3	9.5	8.4	18.9	1.6	3.2	28.9	6.1	7.0	28.6	3.7	5.9

U denotes the point is uniformly distributed between 0 and 1, and C denotes the point is normally distributed with $\mu = 0.5$ and $\sigma^2 = 0.12^2$. Drivers' destinations are estimated.

measure the ultimate outcome of suppliers' utilities. Instead, we take both a rider and driver perspective and report the following rates based on the simulation results after one recommendation cycle:

- *1-to-1 match rate (F)*: number of ride requests selected by a unique driver divided by m.
- *Hot rider rate (H)*: number of ride requests selected by more than one driver divided m.
- *Angry rider rate (W)*: number of ride requests not selected by any driver divided by m.
- *Serviced ride rate (G)*: number of ride requests matched to a driver divided by n.
- *Angry driver rate (Z)*: number of drivers who selected a ride request but was not matched due to collisions, divided by n.
- *Repelled driver rate (Q)*: number of drivers who selected the no-choice selection, divided by n.

Tables 3 and 4 provide these rates from the riders' and drivers' perspectives, respectively. These rates are presented for all methods and factor values. The hrc column presents the results assuming the best menu size was used, i.e., θ^*. We also present the special case of the hrc approach when $\theta = 1$. The ctr and the hrc with $\theta = 1$ both always recommends

each driver exactly one request and no two drivers are recommended the same ride request. The difference is that *ctr* ignores expected driver no choice utilities, but *hrc* with $\theta = 1$ considers drivers' expected no choice utilities. We present *hrc* with $\theta = 1$ to illustrate that the improved performance rates of the *hrc* approach is not due solely to considering expected no-choice preferences. Instead, the benefit comes from the *hrc* method's ability to consider drivers' no-choice preferences, its ability to offer a personalized recommendation set to suppliers to choose from, and to offer overlapping requests to drivers.

Overall all instances, the best serviced ride rate is achieved by *hrc* at 52.8% (i.e., 49.9% for 1-to-1 matches and 2.9% hot requests), followed by *msm*. The *dec* and *ctr* models have similar serviced ride rates at 43.4%. These results are based on a single recommendation cycle. In practice, the angry ride requests would be offered in subsequent recommendation cycles until a given service rate is achieved. Despite a higher rate of repelled drivers, compared to *msm* and *dec*, the average angry driver rate of *hrc* is the lowest at 47.2%. While *ctr* results in zero angry drivers, it has the highest repelled driver rate of 56.6%, because drivers reject *ctr*'s single-sized menu recommendation and do not participate. A centralized approach considering drivers' no choice utilities (i.e., *hrc* with $\theta = 1$), does better with 50.4% repelled drivers, but still not as good as *hrc*. The *msm* recommendation is second (after *hrc*) due to a low average angry driver rate of 7.5%. As excess discretion creates collisions, *dec* has the worst average underutilized resource rate. Choice is thus problematic even for the drivers themselves. These results indicate that neither *ctr* nor *dec* are a suitable recommendation mechanism. Although they represent two extremes, both provide a low rate of 43.4% serviced rides, and a high angry rider rate of 56.6%. Instead, we find that providing choices and recommending alternatives to more than one driver can improve the average serviced ride rate. Consequently, using recommendation sets as a coordination mechanism is beneficial to not only the platform, but also drivers and requests.

The performance rates in Tables 3 and 4 vary for different input parameter levels. However, the trends identified for the platform objective in previous sections are aligned to driver and rider performance rates. For example, with flexible drivers (high expected threshold) the 1-to-1 match rates and serviced ride rates are higher regardless of methods compared to the case of inflexible drivers.

Table 3 Riders' perspective rates for the different recommendation methods and factors, where F denotes the 1-to-1 match rate, H denotes the Hot rider rate, and W the Angry rider rate.

		hrc			*ctr*			*msm*			*dec*			*hrc* with $\theta=1$		
Factor	Levels and Values	F	H	W	F	H	W	F	H	W	F	H	W	F	H	W
Average over all 1140 instances		49.9	2.9	47.2	43.4	0.0	56.6	44.0	6.7	49.3	27.5	15.9	56.6	47.7	0.8	51.5
expected threshold	low: $\delta_j =\sim N(\mu_{\delta_j}=\sqrt{2}/5, \sigma^2_{\delta_j}=0.1^2)$	34.8	4.8	60.4	25.4	0.0	74.6	31.5	6.5	62.0	26.1	10.8	63.1	29.9	1.7	68.4
expected threshold	high: $\delta_j =\sim N(\mu_{\delta_j}=\sqrt{2}/2, \sigma^2_{\delta_j}=0.1^2)$	65.0	1.1	34.0	61.3	0.0	38.7	56.5	6.9	36.5	28.9	21.1	50.0	65.4	0.0	34.6
rejection penalty	$r_i=0.1$	49.9	2.4	47.7	43.2	0.0	56.8	44.2	6.2	49.6	27.5	15.8	56.7	47.5	0.8	51.6
rejection penalty	$r_i=0.5$	50.1	2.8	47.1	43.6	0.0	56.4	44.2	6.6	49.2	27.5	16.0	56.5	47.8	0.9	51.3
rejection penalty	$r_i=1.0$	50.1	3.0	46.9	43.4	0.0	56.6	43.7	7.1	49.2	27.3	16.1	56.6	48.0	0.8	51.2
rejection penalty	$r_i=1.5$	49.3	3.6	47.1	43.2	0.0	56.8	44.0	7.0	49.0	27.7	15.8	56.5	47.3	0.8	51.9
destination error	low: $\epsilon_j =\sim N(\mu_{\epsilon_j}=0, \sigma^2_{\epsilon_j}=0.1^2)$	62.2	2.5	35.3	54.0	0.0	46.0	52.7	8.0	39.3	28.1	19.5	52.4	61.6	1.1	37.2
destination error	high: $\epsilon_j =\sim N(\mu_{\epsilon_j}=0, \sigma^2_{\epsilon_j}=0.5^2)$	37.5	3.4	59.0	32.7	0.0	67.3	35.3	5.5	59.2	26.9	12.4	60.7	33.7	0.5	65.8
no-choice error	low: $e_j =\sim N(\mu_{e_j}=0, \sigma^2_{e_j}=0.1^2)$	53.0	3.3	43.8	42.5	0.0	57.5	45.9	7.5	46.6	27.8	17.4	54.8	50.2	0.4	49.4
no-choice error	high: $e_j =\sim N(\mu_{e_j}=0, \sigma^2_{e_j}=0.1^2)$	46.8	2.6	50.6	44.3	0.0	55.7	42.1	5.9	51.9	27.2	14.5	58.3	45.1	1.3	53.6
riders' O-D, drivers' O-D distributions	UU, UU	49.7	3.0	47.3	38.6	0.0	61.4	44.5	6.5	49.1	31.2	15.9	52.8	46.9	0.8	52.4
riders' O-D, drivers' O-D distributions	UU, UC	49.4	3.8	46.8	45.6	0.0	54.4	40.7	8.9	50.5	24.3	17.0	58.7	46.7	1.0	52.3
riders' O-D, drivers' O-D distributions	UU, CU	49.4	3.2	47.5	37.9	0.0	62.1	44.6	7.0	48.4	31.4	15.6	53.0	47.0	0.7	52.2
riders' O-D, drivers' O-D distributions	UC, UU	46.7	2.1	51.3	44.3	0.0	55.7	43.6	4.0	52.5	23.2	14.4	62.4	44.2	1.2	54.6
riders' O-D, drivers' O-D distributions	UC, UC	57.6	0.7	41.7	55.5	0.0	44.5	49.4	5.6	45.0	27.9	16.9	55.2	56.8	0.2	43.0
riders' O-D, drivers' O-D distributions	UC, CU	47.6	2.1	50.3	45.2	0.0	54.8	43.7	4.7	51.6	22.9	14.7	62.4	45.6	1.1	53.3
riders' O-D, drivers' O-D distributions	CU, UU	49.1	3.6	47.3	38.4	0.0	61.6	44.2	6.8	49.0	30.8	15.6	53.6	46.8	0.7	52.4
riders' O-D, drivers' O-D distributions	CU, UC	50.6	3.4	46.0	46.1	0.0	53.9	40.9	9.7	49.4	24.4	17.5	58.0	47.5	1.1	51.4
riders' O-D, drivers' O-D distributions	CU, CU	48.9	4.7	46.4	38.5	0.0	61.5	44.5	7.4	48.0	31.3	15.8	52.9	47.3	0.8	51.9

Table 4 Drivers' perspective rates for the different recommendation methods and factors, where G denotes the Serviced ride rate, Z denotes the Angry driver rate, and Q the Repelled driver rate.

		hrc			*ctr*			*msm*			*dec*			*hrc* with $\theta=1$		
Factor	Levels and Values	G	Z	Q	G	Z	Q	G	Z	Q	G	Z	Q	G	Z	Q
Average over all 1140 instances		52.8	3.3	43.8	43.4	0.0	56.6	50.7	7.5	41.7	43.4	23.6	32.9	48.5	1.1	50.4
expected threshold	low: $\delta_j =\sim N(\mu_{\delta_j}=\sqrt{2}/5, \sigma^2_{\delta_j}=0.1^2)$	39.6	5.5	54.9	25.4	0.0	74.6	38.0	7.6	54.4	36.9	14.8	48.2	31.6	2.1	66.3
expected threshold	high: $\delta_j =\sim N(\mu_{\delta_j}=\sqrt{2}/2, \sigma^2_{\delta_j}=0.1^2)$	66.0	1.2	32.7	61.3	0.0	38.7	63.5	7.5	29.0	50.0	32.4	17.6	65.4	0.0	34.5
rejection penalty	$r_i=0.1$	52.3	2.7	45.0	43.2	0.0	56.8	50.4	6.9	42.7	43.3	23.4	33.3	48.4	1.0	50.6
rejection penalty	$r_i=0.5$	52.9	3.1	44.0	43.6	0.0	56.4	50.8	7.3	42.0	43.5	23.7	32.8	48.7	1.0	50.3
rejection penalty	$r_i=1.0$	53.1	3.4	43.4	43.4	0.0	56.6	50.8	8.0	41.2	43.4	24.0	32.5	48.8	1.1	50.1
rejection penalty	$r_i=1.5$	52.9	4.1	42.9	43.2	0.0	56.8	51.0	8.0	41.0	43.5	23.4	33.0	48.1	1.1	50.8
destination error	low: $\epsilon_j =\sim N(\mu_{\epsilon_j}=0, \sigma^2_{\epsilon_j}=0.1^2)$	64.7	2.7	32.6	54.0	0.0	46.0	60.7	8.9	30.4	47.6	30.7	21.7	62.8	1.5	35.8
destination error	high: $\epsilon_j =\sim N(\mu_{\epsilon_j}=0, \sigma^2_{\epsilon_j}=0.5^2)$	41.0	4.0	55.1	32.7	0.0	67.3	40.8	6.2	53.0	39.3	16.6	44.1	34.2	0.7	65.1
no-choice error	low: $e_j =\sim N(\mu_{e_j}=0, \sigma^2_{e_j}=0.1^2)$	56.2	3.7	40.1	42.5	0.0	57.5	53.4	8.5	38.1	45.2	26.6	28.2	50.6	0.4	49.0
no-choice error	high: $e_j =\sim N(\mu_{e_j}=0, \sigma^2_{e_j}=0.1^2)$	49.4	3.0	47.6	44.3	0.0	55.7	48.1	6.6	45.4	41.7	20.7	37.7	46.4	1.7	51.9
riders' O-D, drivers' O-D distributions	UU, UU	52.7	3.4	43.9	38.6	0.0	61.4	50.9	7.1	42.0	47.2	21.5	31.3	47.6	0.9	51.4
riders' O-D, drivers' O-D distributions	UU, UC	53.2	4.4	42.4	45.6	0.0	54.4	49.5	10.1	40.3	41.3	30.5	28.2	47.7	1.3	51.0
riders' O-D, drivers' O-D distributions	UU, CU	52.5	3.5	43.9	37.9	0.0	62.1	51.6	7.8	40.6	47.0	20.9	32.2	47.8	0.8	51.4
riders' O-D, drivers' O-D distributions	UC, UU	48.7	2.3	49.0	44.3	0.0	55.7	47.5	4.3	48.2	37.6	20.9	41.5	45.4	1.6	52.9
riders' O-D, drivers' O-D distributions	UC, UC	58.3	0.7	41.0	55.5	0.0	44.5	55.0	5.9	39.1	44.8	23.7	31.5	57.0	0.2	42.8
riders' O-D, drivers' O-D distributions	UC, CU	49.7	2.3	48.0	45.2	0.0	54.8	48.4	5.0	46.6	37.6	21.8	40.6	46.7	1.4	51.9
riders' O-D, drivers' O-D distributions	CU, UU	52.7	4.1	43.3	38.4	0.0	61.6	51.0	7.9	41.1	46.4	21.3	32.4	47.6	0.9	51.6
riders' O-D, drivers' O-D distributions	CU, UC	54.0	3.9	42.1	46.1	0.0	53.9	50.6	11.3	38.1	42.0	31.0	27.0	48.6	1.4	50.0
riders' O-D, drivers' O-D distributions	CU, CU	53.6	5.4	41.0	38.5	0.0	61.5	52.0	8.5	39.5	47.1	21.2	31.7	48.1	0.9	51.0

8. Conclusions and Future Research

In peer-to-peer resource sharing systems, the platform does not set capacity; instead, capacity must be enticed from decentralized suppliers who own the resources. This is the

first work to study how platforms can coordinate decentralized resources to fulfill demand requests using personalized recommendations (i.e., providing a menu of choices). We propose a new hierarchical approach, in which the platform first decides the composition of multiple, simultaneous personalized recommendations. Then, suppliers have autonomy to select/reject the recommended requests. A novel bi-level optimization framework captures the platform's performance as a function of interdependent suppliers' selections. We develop an equivalent single-level reformulation technique that is shown computationally superior to an existing solution technique. Through a design of experiment with five levels and 1440 instances, we explore a platform with uncertainty about suppliers' selections and find that providing choices and recommending alternatives to more than one supplier simultaneously provide value, on average, to the platform, suppliers, and demand requests. When the platform is only partially able to estimate suppliers' utilities, the hierarchical approach creates recommendation sets with higher average platform performance than centralized, decentralized, and many-to-many stable matching approaches. A centralized approach does not perform well due to a higher chance of suppliers electing not to participate in the platform's assignment. However, too much choice can be problematic even for the suppliers themselves. As excess discretion creates collisions, the decentralized approach had the worst angry driver rate and lowest serviced ride rate. Thus, in uncertain environments, the proposed hierarchical approach can benefit the platform, drivers, and ride requests.

A platform with uncertainty about suppliers' selection outcomes can sometimes improve its performance by offering choices to suppliers, but not always. In platforms where uncertainty over suppliers' selection exists, choices help the platform when (1) suppliers are inflexible with a high no-choice utility; and (2) suppliers' utility values have higher variance than the platform's utility values. Choices increase the chance of enticing supplier participation, and in both cases, the increase in participation likelihood outweighs the match's reduced platform benefit likelihood. However, when (1) the platform's benefit values have higher variance than the suppliers' expected utilities or (2) suppliers are flexible, offering additional choices, on average, can reduce the platform's benefit.

This work provides the first proof-of-concept that recommendation sets can be used as a coordination mechanism for distributed resource allocation problems. It creates the foundation for a number of future research directions, including (1) incorporating stochastic

supplier selection into the optimization model, which would allow determining directly, via the optimization formulation, optimal menu size and request overlaps; (2) capturing dynamic aspects of arriving suppliers, requests, and recourse actions; (3) determining how best to influence suppliers' utilities through compensation decisions and how to learn suppliers' behaviors overtime; and (4) developing online optimization and heuristic methodologies. Future research also includes the opportunity to explore varied scenarios, for example, when suppliers can filter requests that meet their threshold, when suppliers can select and fulfill more than one request, when tasks require more than one supplier, and when tasks have additional capacity constraints.

Acknowledgments

This work was partially funded by the National Science Foundation CAREER award 1751801.

References

Agatz N, Erera A, Savelsbergh M, Wang X (2012) Optimization for dynamic ride-sharing: a review. *European Journal of Operational Research* 223(2):295–303.

Allen D (2015) The sharing economy. *Review-Institute of Public Affairs* 67(3):24.

Anshelevich E, Das S, Naamad Y (2013) Anarchy, stability, and utopia: creating better matchings. *Autonomous Agents and Multi-Agent Systems* 1–21.

Archetti C, Savelsbergh M, Speranza MG (2016) The vehicle routing problem with occasional drivers. *European Journal of Operational Research* 254(2):472–480.

Arslan AM, Agatz N, Kroon L, Zuidwijk R (2018) Crowdsourced delivery a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science* online.

Audet C, Hansen P, Jaumard B, Savard G (1997) Links between linear bilevel and mixed 0–1 programming problems. *Journal of Optimization Theory and Applications* 93(2):273–300.

Bai J, So KC, Tang CS, Chen X, Wang H (2018) Coordinating supply and demand on an on-demand service platform with impatient customers. *Manufacturing & Service Operations Management* .

Bard JF (1998) *Practical Bilevel Optimization: Algorithms and Applications*, volume 30 (Springer Science & Business Media).

Bard JF, Moore JT (1992) An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)* 39(3):419–435.

Bardhi F, Eckhardt GM (2012) Access-based consumption: the case of car sharing. *Journal of Consumer Research* 39(4):881–898.

Benjaafar S, Kong G, Li X, Courcoubetis C (2018) Peer-to-peer product sharing: Implications for ownership, usage, and social welfare in the sharing economy. *Management Science* .

Bernstein F, Federgruen A (2005) Decentralized supply chains with competing retailers under demand uncertainty. *Management Science* 51(1):18–29.

Bernstein F, Kök AG, Xie L (2015) Dynamic assortment customization with limited inventories. *Manufacturing & Service Operations Management* 17(4):538–553.

Bitran G, Caldentey R (2003) An overview of pricing models for revenue management. *Manufacturing & Service Operations Management* 5(3):203–229.

Bracken J, McGill JT (1973) Mathematical programs with optimization problems in the constraints. *Operations Research* 21(1):37–44.

Brotcorne L, Labbé M, Marcotte P, Savard G (2001) A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science* 35(4):345–358.

Cachon GP, Daniels KM, Lobel R (2017) The role of surge pricing on a service platform with self-scheduling capacity. *Manufacturing & Service Operations Management* 19:368–384.

Cachon GP, Lariviere MA (2005) Supply chain coordination with revenue-sharing contracts: strengths and limitations. *Management Science* 51(1):30–44.

Cao M, Zhang Q (2011) Supply chain collaboration: impact on collaborative advantage and firm performance. *Journal of Operations Management* 29(3):163–180.

Chen F, Federgruen A, Zheng YS (2001) Coordination mechanisms for a distribution system with one supplier and multiple retailers. *Management Science* 47(5):693–708.

Chen MK, Sheldon M (2016) Dynamic pricing in a labor market: surge pricing and flexible work on the uber platform. *EC*, 455.

Cleophas C, Cottrill C, Ehmke JF, Tierney K (2019) Collaborative urban transportation: Recent advances in theory and practice. *European Journal of Operational Research* 273(3):801–816.

Colson B, Marcotte P, Savard G (2007) An overview of bilevel optimization. *Annals of Operations Research* 153(1):235–256.

Cook J (2015) Uber's internal charts show how its driver-rating system actually works. *Business Insider,* `https://www.businessinsider.com/leaked-charts-show-how-ubers-driver-rating-system-works-2015-2`.

Cullen Z, Farronato C (2014) Outsourcing tasks online: matching supply and demand on peer-to-peer internet platforms. *Working Paper* .

Deakin E, Frick KT, Shively KM (2010) Markets for dynamic ridesharing? *Transportation Research Record: Journal of the Transportation Research Board* 2187(1):131–137.

Dempe S (2018) Bilevel optimization: theory, algorithms and applications. *Optimization On-line* URL `http://www.optimization-online.org/DB_HTML/2018/08/6773.html`.

Dempe S, Franke S (2016) On the solution of convex bilevel optimization problems. *Computational Optimization and Applications* 63(3):685–703.

Einav L, Farronato C, Levin J (2016) Peer-to-peer markets. *Annual Review of Economics* 8:615–635.

Ekstrand MD, Riedl JT, Konstan JA (2011) Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction* 4(2):81–173.

Farahani RZ, Miandoabchi E, Szeto WY, Rashidi H (2013) A review of urban transportation network design problems. *European Journal of Operational Research* 229(2):281–302.

Feldman JB, Topaloglu H (2015) Capacity constraints across nests in assortment optimization under the nested logit model. *Operations Research* 63(4):812–822.

Fortuny-Amat J, McCarl B (1981) A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society* 32(9):783–792.

Fradkin A (2017) Search, matching, and the role of digital marketplace design in enabling trade: evidence from airbnb. *NBER Working Paper* .

Fragiadakis D, Iwasaki A, Troyan P, Ueda S, Yokoo M (2016) Strategyproof matching with minimum quotas. *ACM Transactions on Economics and Computation* 4(1):6.

Frangioni A (1995) On a new class of bilevel programming problems and its use for reformulating mixed integer problems. *European Journal of Operational Research* 82(3):615–646.

Furuhata M, Dessouky M, Ordóñez F, Brunet ME, Wang X, Koenig S (2013) Ridesharing: the state-of-the-art and future directions. *Transportation Research Part B: Methodological* 57:28–46.

Gale D, Shapley LS (1962) College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1):9–15.

Gallego G, Ratliff R, Shebalov S (2014) A general attraction model and sales-based linear program for network revenue management under customer choice. *Operations Research* 63(1):212–232.

Gao Z, Wu J, Sun H (2005) Solution algorithm for the bi-level discrete network design problem. *Transportation Research Part B: Methodological* 39(6):479–495.

Govindan K, Popiuc MN (2014) Reverse supply chain coordination by revenue sharing contract: a case for the personal computers industry. *European Journal of Operational Research* 233(2):326–336.

Ha AY, Tian Q, Tong S (2017) Information sharing in competing supply chains with production cost reduction. *Manufacturing & Service Operations Management* 19(2):246–262.

Hampshire RC, Sinha S (2011) A simulation study of peer-to-peer carsharing. *2011 IEEE Forum on Integrated and Sustainable Transportation System (FISTS)*, 159–163.

Heydari J, Choi TM, Radkhah S (2017) Pareto improving supply chain coordination under a money-back guarantee service program. *Service Science* 9(2):91–105.

Horton JJ (2014) Misdirected search effort in a matching market: causes, consequences and a partial solution. *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, 357–357 (ACM).

Huber GP (1974) Multi-attribute utility models: a review of field and field-like studies. *Management Science* 20(10):1393–1402.

Jiang B, Tian L (2016) Collaborative consumption: Strategic and economic implications of product sharing. *Management Science* 64(3):1171–1188.

Johari R, Kamble V, Kanoria Y (2016) Know your customer: multi-armed bandits with capacity constraints. *Working Paper available at arXiv preprint arXiv:1603.04549* .

Kafle N, Zou B, Lin J (2017) Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. *Transportation Research Part B: Methodological* 99:62–82.

Kamada Y, Kojima F (2014) Efficient matching under distributional constraints: theory and applications. *The American Economic Review* 105(1):67–99.

Kleiner A, Nebel B, Ziparo VA, Srl A (2011) A mechanism for dynamic ride sharing based on parallel auctions. *Procedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 266–272.

Kök AG, Fisher ML, Vaidyanathan R (2015) Assortment planning: review of literature and industry practice. *Retail Supply Chain Management*, 175–236 (Springer).

Krishnan H, Winter RA (2010) Inventory dynamics and supply chain coordination. *Management Science* 56(1):141–147.

Lee A, Savelsbergh M (2015) Dynamic ridesharing: is there a role for dedicated drivers? *Transportation Research Part B: Methodological* 81:483–497.

Lee H, Whang S (1999) Decentralized multi-echelon supply chains: incentives and information. *Management Science* 45(5):633–640.

Li B, Krushinsky D, Reijers HA, Van Woensel T (2014) The share-a-ride problem: People and parcels sharing taxis. *European Journal of Operational Research* 238(1):31–40.

Liu Y, Li Y (2017) Pricing scheme design of ridesharing program in morning commute problem. *Transportation Research Part C: Emerging Technologies* 79:156–177.

Lofberg J (2004) Yalmip: A toolbox for modeling and optimization in matlab. *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, 284–289 (IEEE).

Lofberg J (2016) solvebilevel. online, `https://yalmip.github.io/command/solvebilevel/`.

Lops P, De Gemmis M, Semeraro G (2011) Content-based recommender systems: state of the art and trends. *Recommender Systems Handbook*, 73–105 (Springer).

Masoud N, Jayakrishnan R (2017) A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. *Transportation Research Part B: Methodological* 106:218–236.

McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. *Mathematical programming* 10(1):147–175.

Mofidi SS (2018) *Two Stage Resource Allocation Decisions in Modern Distribution*. Ph.D. thesis, Rensselaer Polytechnic Institute.

Mourad A, Puchinger J, Chu C (2019) A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological* .

Murphy C (2016) Shared mobility and the transformation of public transit. *The National Academies Transportation Research Board TCRP J-11/TASK 21* .

Narasimhan C, Papatla P, Jiang B, Kopalle PK, Messinger PR, Moorthy S, Proserpio D, Subramanian U, Wu C, Zhu T (2018) Sharing economy: Review of current research and future directions. *Customer Needs and Solutions* 5(1-2):93–106.

Newton C (2014) Taskrabbit is blowing up its business model and becoming the uber for everything. *The Verge,* `http://www.theverge.com/2014/6/17/5816254/taskrabbit-blows-up-its-auction-house-to-offer-services-on-demand`.

Nourinejad M, Roorda MJ (2016) Agent based model for dynamic ridesharing. *Transportation Research Part C: Emerging Technologies* 64:117–132.

Pelzer D, Xiao J, Zehe D, Lees MH, Knoll AC, Aydt H (2015) A partition-based match making algorithm for dynamic ridesharing. *IEEE Transactions on Intelligent Transportation Systems* 16(5):2587–2598.

Phillips RL (2005) *Pricing and Revenue Optimization* (Stanford University Press).

Powell WB, Towns MT, Marar A (2000) On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance. *Transportation Science* 34(1):67–85.

Price R, Messinger PR (2005) Optimal recommendation sets: Covering uncertainty over user preferences. *AAAI*, volume 10, 5.

Qi X, Bard JF, Yu G (2004) Supply chain coordination with demand disruptions. *Omega* 32(4):301–312.

Roth AE, Rothblum UG, Vande Vate JH (1993) Stable matchings, optimal assignments, and linear programming. *Mathematics of operations research* 18(4):803–828.

Roth AE, Sotomayor M (1992) Two-sided matching. *Handbook of Game Theory with Economic Applications* 1:485–541.

Rougès JF, Montreuil B (2014) Crowdsourcing delivery: new interconnected business models to reinvent delivery. *1st International Physical Internet Conference*, 28–30.

Shaheen SA, Cohen AP (2013) Carsharing and personal vehicle services: worldwide market developments and emerging trends. *International Journal of Sustainable Transportation* 7(1):5–34.

Shapiro J (1979) *Mathematical Programming: Structures and Algorithms* (Wiley).

Sotomayor M (1999) Three remarks on the many-to-many stable matching problem. *Mathematical social sciences* 38(1):55–70.

Steininger KW, Bachner G (2014) Extending car-sharing to serve commuters: an implementation in Austria. *Ecological Economics* 101:64–66.

Stiglic M, Agatz N, Savelsbergh M, Gradisar M (2016) Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review* 91:190–207.

Sullivan KM, Smith JC, Morton DP (2014) Convex hull representation of the deterministic bipartite network interdiction problem. *Mathematical Programming* 145(1-2):349–376.

Train KE (2009) *Discrete Choice Methods with Simulation* (Cambridge University Press).

Tsay AA, Agrawal N (2004) Channel conflict and coordination in the e-commerce age. *Production and Operations Management* 13(1):93–110.

Von Stackelberg H (1952) *The Theory of the Market Economy* (Oxford University Press).

Wang X, Agatz N, Erera A (2017) Stable matching for dynamic ride-sharing systems. *Transportation Science* 52:739–1034.

Weber TA (2016) Product pricing in a peer-to-peer economy. *Journal of Management Information Systems* 33(2):573–596.

Winter S, Nittel S (2006) Ad hoc shared ride trip planning by mobile geosensor networks. *International Journal of Geographical Information Science* 20(8):899–916.

Xing X, Warden T, Nicolai T, Herzog O (2009) Smize: a spontaneous ride-sharing system for individual urban transit. *Multiagent System Technologies*, 165–176 (Springer).

Zeng B, An Y (2014) Solving bilevel mixed integer program by reformulations and decomposition. *Optimization On-line* .

Appendix A: Computational Performance of Reformulation Technique

In this section, we test the computational performance of the reformulation technique (RFT) against a common bi-level solution approach, the Fortuny-Amat and McCarl (1981) method. This method, which we denote as BLF-KKT, derives the Karush-Kuhn-Tucker (KKT) conditions of the lower level problem (i.e., (10)-(13)) as a single-level mixed integer program. This translates the lower level problem into sets of constraints, incorporated into the upper level problem, resulting in a single level problem. Nevertheless, the complementarity terms are nonlinear; thus, we employ the YALMIP package, a MATLAB based toolbox provided by (Lofberg 2004), which models the complementary slackness conditions using a big-M approach. For more details, readers are referred to Lofberg (2016), McCormick (1976). This approach was chosen for comparison as KKT based reformulation methods remain popular to solve bilevel problems, in which the follower problem is replaced with nonlinear complementarity terms (Zeng and An 2014), and are a common benchmark due to it being computationally available for implementation in Lofberg (2016). We set $a_i = n \forall i \in$

A, such that no restriction exists on how many suppliers the platform can recommend a single request. All suppliers are offered the no-choice alternative, which has a platform benefit of zero. We randomly generate supplier j's utility for the no-choice alternative using the normal distribution $\nu_{(i\in N)j} \sim \sqrt{2} - N(0.282, 0.1^2)$. Collision and rejection penalties are $d_i = r_i = 0.1$. We test instances with varying menu sizes, for both balanced systems, in which $m = n$, and unbalanced systems, in which the number of requests (m) double or triple the number of suppliers (n) and vice versa. The platform's benefit matrix and suppliers' utility matrix are calculated based on generating O-D pairs for n suppliers and m requests as uniformly random variables in a unit square (see Section 6.1.1). Five replications generate different platform and suppliers utility values (i.e., different c_{ij} and ν_{ij} values).

To solve both approaches we adopt YALMIP, a MATLAB based toolbox, provided by Lofberg (2004) and call Gurobi Optimization 7.5.2. Computations are performed on a quad-core 2.7 GHz with 16GB RAM using MATLAB 8.5 (R2015a). The global run-time limit (including pre-solve and optimization) is set to 600 seconds. Table 5 reports each approach's maximum, average, and standard deviation of the runtime (in seconds) across five replications. If the runtime limit is reached without finding the optimal solution, the number of such instances (out of 5) is reported in parenthesis in the max column. As shown in Table 5, the RFT method always outperforms BL-KKT in terms of computational time, and both are exact approaches. The advantage in computational time becomes more evident as the problem size increases. The solution time for the BL-KKT grows very fast. For the largest problem size 35×35, BL-KKT fails to reach the optimal solution in 600 seconds for all θ values and all replications tested; while the reformulation technique finds an optimal solution in less than 6.5 seconds.

Table 5 Balanced network computational comparisons (in seconds) for BL-KKT and RFT methods.

		BL-KKT			RFT				BL-KKT			RFT				BL-KKT			RFT		
mxn	θ	Max	Ave.	Std	Max	Ave.	Std	θ	Max	Ave.	Std	Max	Ave.	Std	θ	Max	Ave.	Std	Max	Ave.	Std
5×5	1	2.2	2.2	0.1	0.2	0.2	0.0	5	2.8	2.7	0.1	0.2	0.2	0.0							
10×10	1	7.7	7.3	0.3	0.4	0.3	0.0	5	9.7	9.1	0.6	0.4	0.4	0.0							
15×15	1	26.2	24.9	0.8	0.4	0.4	0.0	5	31.5	30.0	0.9	0.5	0.5	0.0	10	29.4	28.2	0.8	0.5	0.4	0.0
20×20	1	80.2	78.2	1.7	0.4	0.4	0.0	5	89.7	87.3	1.8	0.8	0.6	0.1	10	90.3	88.2	1.9	0.7	0.6	0.1
25×25	1	229.1	218.7	8.0	0.5	0.4	0.0	5	240.6	227.6	10.4	1.1	1.0	0.1	10	237.4	227.2	8.5	1.2	1.0	0.1
30×30	1	600.0(1)	580.9	17.8	0.5	0.5	0.0	5	600.0(1)	584.1	17.1	3.4	2.1	1.1	10	600.0(2)	581.9	17.6	2.7	1.9	0.8
35×35	1	600.0(5)	600.0	0.0	0.5	0.5	0.0	5	600.0(5)	600.0	0.0	6.5	4.1	1.7	10	600.0(5)	600.0	0.0	6.5	4.1	1.9

Table 6 presents computational results for unbalanced networks. RFT computationally outperforms BL-KKT across all instances, with BL-KKT's runtime growth increasing at a higher rate. For both the BL-KKT and the RFT methods, the number of demand requests (m) influences the solution time more than the number of suppliers (n). As m increases, the number of combinations of selecting θ alternatives in the menu from m requests increases combinatorially. Even so, RFT solves all replications tested in less than 6 seconds.

Next, we test the computational limitations of our reformulation technique. To do so, we explore balanced problem instances with the same inputs as described for Tables 5 and 6. Table 7 reports the maximum, average, and standard deviation of the runtime (in seconds) and the optimality gap. If the optimal solution

Table 6 Unbalanced network computational comparisons (in seconds) for BL-KKT and RFT methods.

		BL-KKT			RFT					BL-KKT			RFT		
$m \times n$	θ	Max	Ave	Std	Max	Ave	Std	$m \times n$	θ	Max	Ave	Std	Max	Ave	Std
5×10	1	3.7	3.5	0.2	0.3	0.3	0.0	10×5	1	12.0	12.0	0.1	0.3	0.3	0.0
5×10	3	3.8	3.6	0.2	0.3	0.3	0.0	10×5	3	12.1	12.0	0.1	0.3	0.3	0.0
5×10	5	3.7	3.6	0.2	0.3	0.3	0.0	10×5	5	12.1	12.0	0.1	0.3	0.3	0.0
5×15	1	6.0	5.9	0.1	0.3	0.3	0.0	15×5	1	14.0	13.8	0.2	0.3	0.3	0.0
5×15	3	6.0	5.9	0.1	0.3	0.3	0.0	15×5	3	14.0	13.8	0.3	0.3	0.3	0.0
5×15	5	6.2	5.9	0.2	0.3	0.3	0.0	15×5	5	14.1	13.8	0.3	0.3	0.3	0.0
10×20	1	22.0	21.5	0.3	0.3	0.3	0.0	20×10	1	29.5	28.8	0.6	0.3	0.3	0.0
10×20	3	22.1	21.6	0.4	0.4	0.3	0.0	20×10	3	29.6	28.8	0.7	0.4	0.4	0.0
10×20	5	22.2	21.6	0.4	0.3	0.3	0.0	20×10	5	29.8	28.9	0.7	0.4	0.4	0.0
10×30	1	48.7	47.8	0.7	0.3	0.3	0.0	30×10	1	54.8	52.3	2.5	0.3	0.3	0.0
10×30	3	48.7	48.0	0.7	0.3	0.3	0.0	30×10	3	54.8	52.4	2.4	0.6	0.4	0.1
10×30	5	49.2	48.1	0.8	0.4	0.3	0.0	30×10	5	55.1	52.6	2.5	0.6	0.4	0.1
15×30	1	115.8	113.2	2.0	0.3	0.3	0.0	30×15	1	320.4	291.4	17.5	0.4	0.4	0.0
15×30	3	116.5	113.9	2.0	0.5	0.4	0.0	30×15	5	327.0	309.5	21.3	0.8	0.7	0.1
15×30	5	116.5	113.8	2.0	0.5	0.4	0.0	30×15	10	290.0	285.4	4.3	0.7	0.6	0.1
15×45	1	286.6	278.5	6.4	0.3	0.3	0.0	45×15	1	478.0	464.6	13.7	0.4	0.4	0.0
15×45	3	284.3	279.0	4.3	0.5	0.5	0.0	45×15	5	519.9	478.5	29.4	2.1	1.1	0.6
15×45	5	281.8	277.9	4.0	0.5	0.5	0.0	45×15	10	529.4	477.1	35.0	1.5	0.9	0.4
20×40	1	451.8	438.8	10.2	0.4	0.4	0.0	40×20	1	600.0(5)	600.0	0.0	0.4	0.4	0.0
20×40	5	458.1	445.4	12.6	0.7	0.6	0.1	40×20	5	600.0(5)	600.0	0.0	1.7	1.2	0.4
20×40	10	455.2	441.0	12.8	0.7	0.6	0.1	40×20	10	600.0(5)	600.0	0.0	1.7	1.2	0.4
20×60	1	600.0(5)	600.0	0.0	0.4	0.4	0.0	60×20	1	600.0(5)	600.0	0.0	0.4	0.4	0.0
20×60	5	600.0(5)	600.0	0.0	0.8	0.7	0.1	60×20	5	600.0(5)	600.0	0.0	6.0	3.9	2.0
20×60	10	600.0(5)	600.0	0.0	1.0	0.8	0.2	60×20	10	600.0(5)	600.0	0.0	5.8	3.3	1.7

is not found, we calculate the gap between the best bound and the best solution found within the 600 second time limit. As the problem size grows, RFT's solution time increases. Also, the menu size influences computational times. When $\theta = 1$, RFT provides optimal solutions for large problems quickly, e.g., the RFT found optimal solutions for 500×500 problem instances under 30.7 seconds. If $\theta > 1$, RFT provides solutions with low optimality gaps for moderate size problems (e.g., a maximum optimality gap less than 4.86% is reported for 100×100 problem sizes after 600 seconds). However, for large problem sizes of 500×500 when $\theta > 1$, quality solutions are not found in the 600 second time limit.

Table 7 RFT's computational runtime (in seconds) and optimality gap (in %).

		Time (seconds)			Optimality Gap (%)				Time (seconds)			Optimality Gap (%)				Time (seconds)			Optimality Gap (%)		
mxn	θ	Max	Ave.	Std	Max	Ave.	Std	θ	Max	Ave.	Std	Max	Ave.	Std	θ	Max	Ave.	Std	Max	Ave.	Std
50×50	1	36.0	33.1	2.0	0.0	0.0	0.0	5	138.5	126.6	9.7	0.0	0.0	0.0	20	172.1	151.1	17.7	0.0	0.0	0.0
75×75	1	38.1	35.1	3.2	0.0	0.0	0.0	5	600.0(1)	421.3	107.1	1.5	0.3	0.7	20	600.0(1)	461.5	113.9	0.9	0.2	0.4
100×100	1	37.6	34.5	2.0	0.0	0.0	0.0	5	600.0(4)	590.3	21.7	2.0	0.9	0.8	20	600.0(5)	600.0	0.0	4.9	2.6	1.6
200×200	1	52.3	44.5	4.5	0.0	0.0	0.0	5	600.0(5)	600.0	0.0	23.1	21.1	2.2	20	600.0(5)	600.0	0.0	23.8	22.6	1.7
300×300	1	63.7	45.3	20.7	0.0	0.0	0.0	5	600.0(5)	600.0	0.0	37.6	34.1	5.6	20	600.0(5)	600.0	0.0	40.5	35.9	4.9
400×400	1	25.7	25.1	0.6	0.0	0.0	0.0	5	600.0(5)	600.0	0.0	41.3	33.4	4.8	20	600.0(5)	600.0	0.0	44.5	36.4	5.1
500×500	1	60.7	58.1	1.7	0.0	0.0	0.0	5	600.0(5)	600.0	0.0	58.9	57.7	1.3	20	600.0(5)	600.0	0.0	59.7	58.4	1.0