The Meal Delivery Routing Problem

Damian Reyes[1], Alan Erera[1], Martin Savelsbergh[1], Sagar Sahasrabudhe[2], and Ryan O'Neil[2]

[1]H. Milton Stewart School of Industrial Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205

[2]Decision Engineering Department, Grubhub, Chicago, IL 60602

March 1, 2018

Abstract

We introduce the *Meal Delivery Routing Problem* (MDRP) to formalize and study an important emerging class of dynamic delivery operations. We develop optimization-based algorithms tailored to solve the courier assignment (dynamic vehicle routing) and capacity management (offline shift scheduling) problems encountered in meal delivery operations. Extensive computational experiments using instances with realistic size, geography, urgency and dynamism demonstrate that our algorithmic ideas provide solid foundations for real-world implementations.

1 Introduction

Online restaurant aggregators – on-demand meal-ordering platforms where diners order their favorite cravings from an array of restaurants – are growing at a fast pace [5], and with them, the volume of meal delivery operations is rising quickly worldwide [10], increasing the potential for new economies of scope, scale, and density. According to Morgan Stanley, "online food delivery could grow by 16% annual compound rate in [the] next 5 years"[17]. Aiming to capitalize on the market opportunity, emerging providers are investing heavily [8] in the deployment of meal delivery networks that promise restaurants and diners a reliable, fast and cost-effective delivery process.

While in the short-term the transition from restaurant-owned delivery services (which for many restaurants means no deliveries at all) to integrated meal delivery networks can focus on reliability and speed, the long-term sustainability of these networks is contingent on turning the efficiency potential into actual profits. For this purpose, appropriate optimization technologies must be developed to solve increasingly large dynamic pickup and delivery problems in near-real time, and prescribe high-quality decisions able to control costs while satisfying very high service standards.

The successful deployment and operation of meal delivery networks is difficult not only due to the scale of these systems, but also due to the dynamism and urgency of arriving orders [26]. Without exaggeration, meal delivery is the ultimate challenge in last mile logistics: a typical order is expected to be delivered within an hour (much less if possible), and within minutes of the food becoming ready, thus reducing consolidation opportunities and imposing the need for more vehicles operating simultaneously and executing shorter routes. Furthermore, meal delivery networks must be able to respond to wide, and often abrupt, swings in demand both in spatial and time dimensions.

In an attempt to achieve the desired responsiveness without the costs linked to employing a sufficiently large permanent fleet of vehicles (and full-time drivers), meal delivery providers have adopted "digital marketplace" business models – where the supply of couriers, i.e., independent contractors making deliveries [21], is managed indirectly through economic incentives. This strategy, first explored in the context of taxi and ride-hailing services, externalizes fixed costs (to couriers) and enhances the ability of the system to plan and control capacity levels over time and geography in sync with demand fluctuations.

However, it is worth emphasizing how full reliance on independent-contractors establishes a fundamentally different operating environment than that of traditional vehicle routing applications. In exchange for internalizing some of the risks and costs associated with demand uncertainty, couriers are entitled to a significant degree of autonomy, thereby adding yet another layer of complexity in the design of appropriate optimization technology: company drivers *will* implement instructions from central planning, whereas independent contractors *might* do so, thus introducing uncertainty in scheduling (couriers have some freedom to choose the hours they work), dispatching (couriers can reject an offered assignment), and routing (couriers can disregard the suggested sequence of deliveries).

In synthesis, meal delivery networks have ushered in dynamic pickup and delivery problems of unprecedented scale, and meal delivery providers are spearheading the adoption of flexible business models, like crowd-sourced delivery [23], in the last-mile goods-transportation sector. In this study, we are concerned with laying out solid foundations for the design of optimization technologies that can scale up to the challenge. We have adopted a dynamic deterministic framework, even if this means that novel and interesting questions (in particular, those related to courier autonomy) are not (yet) explored.

The main contributions of this paper are: i) a definition of the Meal Delivery Routing Problem (MDRP) to model the essential structure of this emerging class of dynamic delivery systems; ii) a solution algorithm based on a rolling-horizon repeated matching approach, designed to solve large-scale instances of the MDRP; iii) an off-line decision support tool to determine a high-quality schedule of courier shifts (which has been used in the instance generation process) iv) the release to the public domain of a set of instances built from real-life historic data to facilitate benchmarking of alternative solution methodologies; v) an extensive computational study demonstrating, among others, that (a) our approach achieves high-quality solutions with respect to multiple service objectives, over a wide spectrum of instance characteristics, despite its simplicity and myopism; (b) capacity scheduling decisions have a critical impact on reliability and performance; (c) meal preparation time uncertainty has a minor impact on reliability and performance..

We close this section with a brief survey of related literature, contextualizing some of the themes of later sections. In Section 2, we define the Meal Delivery Routing Problem, and then describe our solution algorithm (together with some possible variations) in Section 3. Next, in Section 4 we present and discuss the results of a computational study. Finally, in Section 5 we provide some concluding remarks.

1.1 Related literature

The MDRP belongs to the large family of dynamic vehicle routing problems (dVRP), more specifically, to the class of dynamic pickup and delivery problems (dPDP). A vast number of researchers have studied these problems from different angles for decades, and excellent surveys have been produced by Pillac et al. [18] and Psaraftis et al. [20], in the case of dVRP, and Berbeglia et al. [6], in the case of dPDP. In the next paragraphs we will limit ourselves to mention some recent studies that share some features with our work in terms of modeling or solution methodology.

Recent research in dPDP has focused mainly on the movement of people, *e.g.*, dial-a-ride and ride-sharing applications [1, 9], yet, despite the contextual difference, these problems share some important similarities with the MDRP: the difficulties posed by the increasingly large size of fully-dynamic instances, and the high urgency and low flexibility of the tasks to be scheduled. A frequently proposed strategy to deal with this challenge is the use of a myopic rolling horizon repeated matching approach, a scalable framework that has been shown to produce high quality solutions (*e.g.*, [24, 28]), making virtue out of necessity thanks to the low visibility of future events and the tight time constraints present in taxi and ride-sharing environments.

But perhaps the most natural label under which to catalogue the MDRP is that of dynamic *delivery* problems, a type of dPDP that has just recently begun to be studied on its own right, as same-day delivery services become more and more ubiquitous, in both simplified analytic systems (Archetti et al. [2], Klapp et al. [11], Reyes et al. [22]) and more realistic settings (Azi et al. [4], Voccia et al. [27], Klapp [12], Archetti et al. [3], Dayarian and Savelsbergh [7]). In dynamic delivery problems, vehicles make multiple trips during the operating period to deliver goods locally from a depot (or a small number of depots, *i.e.*, restaurants, in the case of the MDRP) to customer locations. Due to the structure of the network (one or few depots), and also due to tight time constraints being modeled, dynamic delivery solutions have a special structure: once a vehicle is dispatched, modifying the route is highly undesirable or impractical. This structure is usually enforced directly in the model formulation, by requiring vehicles to be empty before starting additional pickup operations, which, while restrictive, is not at all unreasonable when dealing with deliveries of perishable goods, like meals. Ongoing assessments by Ulmer et al. [25] on the impact of relaxing this condition in the context of same day delivery suggest that allowing more flexible routes (preemptive returns in the single-depot case) may increase the number of customers served during the operating period, but the change on the average delivery times has not been thoroughly explored (and a trade-off, *i.e.*, an increase in delivery times, is plausible). Our proposed MDRP model definition does not impose that restriction, as in a multi-depot problem it is conceivable that routes may benefit from a vehicle making pickups while still executing deliveries from a *different* depot. However, the solution algorithm presented in this paper always uses routes that finish all pending deliveries before a new pickup.

Having contextualized the problem, let us introduce a few more works which have informed our methods. The use of assignment models in rolling horizon strategies has a long history in the realm of dynamic long-haul fleet management. Here we highlight the pivotal work by Yang et al. [29], where the authors introduce a dynamic model that captures the essential characteristic of a real-time full truckload dispatching system, and compare a series of rolling horizon policies to assign (and schedule) jobs to trucks, under different operating conditions. The value of advanced information is measured by comparing the performance of myopic assignment policies and a policy that uses some *stochastic* knowledge about the expected location of future pickup and delivery points, and uses these expectations to account for the cost of moving trucks to serve future uncertain requests in an otherwise myopic algorithm. Beyond the obvious difference in time scales, the full truckload assumption is a significant departure from the meal delivery operating environment. However, we opt for defining route delivery segments (*i.e.*, consolidation decisions) before solving the assignment problem, which effectively reinstates a similar structure. While we do not use stochastic methods, we do explore the use of uncertain information about the future through the assignment of "follow-on" pick-up and delivery chains, where the later route segment will likely still change as more orders arrive, but already contains enough information to anticipate needed vehicle movements.

An issue of central importance in the design of dynamic algorithms is how to most effectively balance between the fulfillment of current tasks and the preservation of flexibility to complete future and unknown tasks. On this question, we highlight the work of Mitrovic-Minic et al. [16], who propose a "double horizon" algorithm that evaluates the cost of actions (pickup or drop-off insertions, in this case) using different cost functions if the actions occur in the short-term (within a given horizon) or in the long-term (anything beyond the horizon), an approach motivated by applications where time windows are wide and deliveries may occur much later than their corresponding pick-up operation (which is not the case for on-demand meal delivery). The double horizon heuristic is shown to outperform traditional (single) rolling-horizon algorithms, on instances with time windows ranging between 1 hour and 8 hours from the release time, but with diminishing returns as the size of instances grows. Our solution algorithm incorporates a bi-objective mechanism with the same spirit, but operating in a different way (after all, the time constraints in meal delivery are very tight,*e.g.*, 40 minute target since announcement until drop-off, unlike the problems for which "double-horizon" was conceived): we use information from an interval possibly different to the "assignment horizon" (*i.e.*, the window of orders to include in routes) to determine how intensely should consolidation opportunities be prioritized. At or before busy periods, like lunch and dinner time, when many orders arrive in a short time-span, the algorithm attempts to create more efficient routes, even at the expense of individual order service quality, while in relatively calm periods, the emphasis switches to delivering orders fast.

Another important question in dynamic routing is when to postpone and went to commit to the execution of decisions in order to mitigate uncertainty. On this topic, the work by Mitrovic-Minic and Laporte [15] comes to the fore. In [15], the distribution of waiting time for any given route is determined through one of 4 "waiting strategies": Drive First (DF: depart as early as possible), Wait First (WF: depart as late as possible), Dynamic Wait (DF within service zones, WF between service zones), and Advanced Dynamic Wait (DF within service zones, and slightly less "lazy" than WF between zones), where "service zones" are essentially clusters of consecutive stops in a route determined dynamically. In opposition to DF, WF tends to lead to more efficient routes, but at the risk of running out of vehicles to complete all deliveries by their deadline; as expected, the more complex strategy dominates the rest. In the context of meal delivery, service constraints are so restrictive that complex waiting strategies may not have a critical impact: it makes little sense for a vehicle to wait idle after delivering an order if there is more work to do, so the main question becomes how much to postpone departures from a restaurant. Our strategy in this regard is described in detail in Section 3.3, and it can be interpreted as a restricted form of WF at restaurants.

Finally, the concepts of dynamism and urgency are fundamental for the study of dynamic delivery systems, where a large majority, if not the totality, of requests are revealed during the operating period (dynamic requests), and must be completed within a short time window (urgent requests). Naturally, the precise definition of these concepts has evolved throughout the years. Two decades ago, Lund et al. [14], proposed to measure the *degree of dynamism* as the proportion of dynamic requests, a rough measure primarily designed for situations where at least part of the requests are "advance requests". To make meaningful comparisons between scenarios with any proportion of dynamic requests, Larsen et al. [13] later refined the definition and proposed the *effective degree of dynamism*, which attempted to capture both urgency and evolution of information in a single measure. This definition is the most popular one in recent literature. However, van Lon et al. [26] have recently argued that this popular measure has significant flaws, and that dynamism and urgency should be measured separately to correct this. After defining two independent measures, they empirically observe that dynamism, urgency and "cost" are related in a non-linear way: low dynamism and high urgency lead to higher costs, but high dynamism and high urgency do not; higher urgency leads to higher costs; and that cost is largely insensitive to changes in dynamism, all else equal (except for high urgency scenarios, as previously noted). This is consistent with the observations of Lund et al. [14], who noted that "even with a large number of dynamic requests, it is possible to produce good solutions, provided that the dispatcher receives the requests way ahead of the actual service time." In this paper, we adopt the definitions of [26].

2 The meal delivery routing problem

In this section, we introduce a stylized model of meal delivery operations, with the goal of formalizing what we consider to be their main structural features: multiple pick-up points (restaurants), dynamic order arrivals, delivery capacity distributed throughout the day in the form of courier shifts, the possibility to pick up multiple orders simultaneously, among others. Being a first attempt to study such systems, we have assumed a deterministic dynamic framework. Two important real-life features that are not captures are the ability of couriers to turn down delivery assignment offers, and the ability of couriers to relocate freely when idling. Having said this, let us introduce the model.

Let R be a set of restaurants, and let each restaurant $r \in R$ have an associated location ℓ_r. Let O be a set of orders, and let each order $o \in O$ have an associated restaurant $r_o \in R$, a placement time a_o, a ready time (*i.e.* a release date at the restaurant) e_o, and an order drop-off location ℓ_o. Let C be a set of couriers, where each courier $c \in C$ has an on-time e_c (when the courier goes on duty), an on-location ℓ_c (where the courier will be at time e_c), and an off-time $l_c > e_c$ (when the courier goes off duty). Assume that all information about R and C is known *a priori*, but information about any particular order $o \in O$ is revealed only at its placement time a_o. The meal delivery routing problem consists of determining feasible routes for couriers to complete the pick-up and delivery of orders, with the objective to optimize a single or multiple performance measures. The structural assumptions that determine feasibility, as well as relevant performance criteria are defined as follows.

2.1 Structural assumptions

Orders from the same restaurant may be combined into "bundles" with multiple drop-off locations, where the ready time of a bundle is the latest ready time of the constituent orders. There is no limit to the number of orders that may be combined into a bundle.

The travel time between any pair of locations is assumed to be invariant over time. The service time associated with the pickup of an order at a restaurant, s^r, represents the time a courier needs to park his vehicle, walk to the restaurant, pick up one or more orders, and walk back to his vehicle. The value of s^r is invariant over time and independent of the number of orders being picked up. Similarly, s^o, represents the service time associated with delivering an order at a customer location, i.e., the time a courier needs to park his vehicle, walk to the customer, drop off an order, and walk back to his vehicle.

The pickup time of an order (or bundle) at a restaurant is not smaller than the maximum of a) the order ready time and b) the courier arrival time to the restaurant plus half of the service time at a restaurant. The departure time from a restaurant is the pickup time plus half of the service time. The drop-off time of an order is the arrival time of the courier at the customer location plus half of the service time at a customer location. The departure time after delivering an order is the drop-off time plus half of the service time.

Couriers cannot pick up any orders after their off-time, but are allowed to drop off orders after their off-time. More importantly, in this deterministic model, it is assumed that couriers do not execute any autonomous decisions while on duty. In particular, they always accept any instruction handed to them, and they always wait for (new) instructions at their on-location and at the last location of their active assignment. Furthermore, couriers cannot receive instructions while executing an assignment and, thus, cannot be diverted while en-route to a restaurant.

Payments to couriers follow a guaranteed minimum compensation scheme: a courier earns p_1 per delivered order, or is compensated at the rate of p_2 per hour, whichever is higher. Thus a courier collects $\max\{p_1 n,\ p_2(l_c - e_c)\}$, where n is the total number of orders served by the courier.

We consider the following variations in the operating environment:

- *Prepositioning.* Without prepositioning, the only instruction that can be sent to a courier is to pick up and deliver an order (possibly a bundled-order). With prepositioning, either the instruction to move to a restaurant (a prepositioning move) or the instruction to pick up and deliver an order can be sent to a courier.
- *Assignment updates.* Without assignment updates, once the instruction to pick up and deliver an order (possibly a bundled-order) has been communicated to a courier, this assignment has to be completed before a courier can receive a new instruction. With assignment updates, when a courier arrives at a restaurant to pick up an order (possibly a bundled-order), an instruction can be sent to the courier updating the order to be picked up. For example, the initial instruction may have indicated that order o_1 had to be picked up and delivered, and the update instruction may indicate that order o_1 and order o_2 have to be picked up and delivered (a bundled-order).

Note that the only assignment update allowed is an update to the set of orders picked up at a restaurant.

2.2 Performance metrics

Given that meal delivery providers must bring together diners, restaurants and couriers, each with their own concerns, a number of performance measures are of interest. Some, like click-to-door, which is the difference between the drop-off time of an order and the placement time of an order, involve a target value, τ, and a maximum allowed value, $\tau_{\max}$. The primary performance measures for the meal delivery routing problem are:

1. Number of orders delivered.
2. Total courier compensation.
3. Cost per order: total courier compensation divided by number of orders delivered.
4. Fraction of couriers receiving guaranteed minimum compensation.
5. Click-to-door time: the difference between the drop-off time and the placement time of an order.
6. Click-to-door time overage: the difference between the drop-off time of an order and the placement time of an order plus the target click-to-door time.
7. Ready-to-door time: the difference between the drop-off time of an order and the ready time of an order.
8. Ready-to-pickup time: the difference between the pickup time of an order and its ready time.
9. Courier utilization: the fraction of the courier duty time that is devoted to driving, pickup service, and drop-off service (as opposed to time spent waiting).
10. Courier delivery earnings: courier earnings when considering only the number of orders served.
11. Courier compensation: the maximum of the guaranteed minimum compensation (based on the length of the duty period) and the delivery earnings.
12. Orders delivered per hour: for each courier, the number of orders delivered divided by the length of the shift.
13. Bundles picked up per hour: for each courier, the number of order bundles assigned divided by the length of the shift.
14. Orders per bundle: for each assignment, the number of orders to be picked up.

Relevant summary statistics for measures 5-13 are: average, standard deviation, minimum, 10-th percentile, median, 90-th percentile, and maximum.

3 A rolling horizon algorithm for the MDRP

In light of the highly dynamic and urgent nature of meal delivery operations, making a detailed delivery plan for orders that will not be ready in the near future is unlikely to be of much advantage. For this reason, we propose to solve the MDRP using a rolling horizon matching-based algorithm that every f minutes solves a matching problem to prescribe only the next pick-up and delivery assignment for each courier. Individual orders may be bundled to be picked-up together and then delivered by a single courier following a specified route. After defining tentative courier - order assignments, a commitment strategy dictates which of these decisions are postponed and which are communicated to couriers.

Furthermore, since a bundle cannot be picked up before all orders are ready, the assignment of a bundle with a ready time far into the future is likely to be postponed (the most likely outcome of the commitment strategy), our algorithm focuses on finding assignments only for the subset of known upcoming orders whose ready time falls within $t + \Delta_u$, the *assignment horizon*, $U_t = \{o \in O_t : \ e_o \leq t + \Delta_u\}$. So, at optimization time, t, the algorithm determines the best assignment of orders in U_t to the couriers on duty.

3.1 Bundles and routes

While ideally every order should be picked up at its ready time and delivered by its target drop-off time (derived from target click-to-door time), this goal must be reconciled with the reality of a limited set of couriers. Especially during busy periods of the day, it may not always be possible to pick up orders as they leave the kitchen and deliver them individually. Therefore, to better utilize capacity, couriers may pick-up and deliver multiple orders, increasing the utilization of couriers at the expense of some freshness loss. At optimization time t, the algorithm determines how many orders should be in a bundle (defining a target bundle size), and then defines a satisfactory grouping and sequencing of the individual orders that will be assigned to couriers. Since we always define bundles with a unique route associated to them, from now on we may refer to bundles as routes indiscriminately.

System intensity and a target bundle size. A target bundle size may simply be defined as a fixed number throughout the whole operating period, or throughout predefined intervals, such as “lunch” and “dinner” times. However, to induce robustness and responsiveness in our solution method, we define such target in a dynamic fashion, in direct relation to a fraction of the form ${}^{(\#\text{orders ready})}/_{(\#\text{ couriers available})}$, a rough measure of the amount of work that must be completed with the available resources during a given period of time. By doing this, we intend to encourage single-order bundles when there are fewer orders than couriers, and favor larger bundles when there are more orders than couriers and the system is under pressure. A parametric definition of the target bundle size at optimization time t is:

$$Z_t = \left\lceil \frac{|\{o \in O_t : \ e_o \leq t + \Delta_1\}|}{|\{d \in D_t : \ e_d \leq t + \Delta_2\}|} \right\rceil, \quad \Delta_1 > 0, \Delta_2 > 0,$$

where e_o is the ready time of order o and e_d is the time when courier d becomes available for a new assignment. Note that it is possible that there are no couriers available before $t + \Delta_2$, in which case Z_t is set to some default value. Specific values for Δ_u, Δ_1 and Δ_2 are set through a tuning procedure, but, heuristically, for the algorithm to have an anticipatory character, $f < \Delta_u \leqslant \Delta_1$ and $f < \Delta_u \leqslant \Delta_2$ should hold

Creation of bundles and delivery routes. Once a system-wide target bundle size Z_t has been determined, the set $S = \bigcup_{r \in R} S_r$ of bundles to be assigned is built by processing the upcoming orders at each restaurant r, $U_{t,r}$, following the steps described in Procedure 1.

Note that once a bundle reaches its target size, an additional order is inserted only if this increases route efficiency, i.e., the time per order delivered in that bundle decreases. Also note that the parameter β, which controls the importance of freshness in the construction of bundles, is given beforehand (*i.e.*, it should be tuned off-line).

3.2 Assignment logic

At the core of our algorithm, the solution of a bipartite matching problem assigns order bundles to couriers, dictating the next delivery route to be executed by each courier. Such a simple model is able to capture and balance the trade-off between short-term efficiency and service quality, while ensuring that, in practical implementations, optimization is not a significant bottleneck in the overall solution process. But, of course, simplicity comes at the cost of expressiveness: the only levers to guide decisions in the matching are the weights and feasibility conditions of order-courier pairs, and only so many aspects of the problem can be captured by these. In an attempt to retain some more control in the assignment process, specifically around the issue of how to avoid service delays for bundles that are already late, we introduce a priority scheme: assignments are built by first partitioning the set of relevant bundles in priority groups based on their “urgency”, and then finding optimal assignments sequentially for each group.

Procedure 1: Bundle generation using parallel-insertion

```
Input: $U_{t,r}$, set of upcoming orders at restaurant $r$,
       $Z_t$, target bundle size,
       and $k^r_t$, number of couriers that available at $r$ (see Section 3.3).
Output: $S_r$, set of bundles from restaurant $r$ to be assigned to couriers.
/* Initial construction                                                    */
Sort the orders in $U_{t,r}$ by non-decreasing ready time;
Compute the target number of bundles to create, $m_r = \max\left(k^r_t, \lceil |U_{t,r}|/Z_t \rceil\right)$;
Initialize empty routes (bundles) $s_1, s_2, \ldots, s_{m_r}$ in $S_r$;
for $o \in U_{t,r}$ do
    Find the route $s \in S_r$ and the insertion position $i_s$ for order $o$ into route $s$ which results in
      the minimum increase in route cost, where the route cost of $s$ is
      $\sum_{(p,q)\in s} TravelTime(p,q) + \beta \sum_{p \in s} ServiceDelay(p)$;
    if $|s| > Z_t$ and insertion decreases route efficiency then
        Disregard $s$ for order $o$ and find the next best route and insertion position (repeat if
          necessary)
    end
    Insert $o$ in route $s$ at position $i_s$;
end
/* Improvement by ``remove-reinsert'' local-search                         */
for $o \in U_{t,r}$ do
    Remove $o$ from its current route;
    Given the existing routes in $S_r$, find route $s$ and position $i_s$ to re-insert $o$ at minimum cost;
    Re-insert $o$ into route $s$ at position $i_s$;
end
```

Priority scheme. Orders are classified according to their unavoidable delays in drop-off and pick-up:

- **Group I**: orders whose target drop-off time is impossible to achieve.
- **Group II**: orders not in Group I which can no longer be picked up at their ready time.
- **Group III**: orders not in Group I or II.

A bundle is assigned the highest priority of any of its constituent individual orders.

By creating assignments sequentially for these priority groups, urgent deliveries are more likely to achieve an earlier pickup time than if all orders were included in one big matching problem, which may make a difference at the time of commitment, preventing the postponement of delivery for orders that are already late. Similarly, in matchings where there are fewer couriers available than bundles to be assigned, a solution obtained with this priority scheme averts situations where orders that are already late do not get a courier assigned.

Three alternative optimization models, with increasing degrees of complexity, are considered.

3.2.1 Linear assignment model

The simplest assignment model, and the one used by default in our solution algorithm, is a bipartite matching with no side constraints, *i.e.*, a linear program. To define it, we introduce the following notation:

- N_s: number of individual orders in bundle s,
- θ: constant "penalty" for delays in the pick-up of an order or bundle of orders,
- $\pi_{s,d}$: pick-up time of bundle s if assigned to courier d. Note that, by definition, $\pi_{s,d} \geq \max_{o\in s}\{e_o\}$.
- $\delta^s_{o,d}$: drop-off time of order o in bundle s if s is assigned to courier d (note that this value depends on the time that d can pick up s).
- $x_{s,d}$: 0-1 decision variable for the assignment of bundle s to courier d.

Using these definitions, the problem is:

$$\max_x \sum_{d\in D_t}\sum_{s\in S}\left(\frac{N_s}{\max_{o\in s}\{\delta^s_{o,d}\} - e_d} - \theta\left(\pi_{s,d} - \max_{o\in s}\{e_o\}\right)\right)x_{s,d} \tag{1a}$$

$$\text{s.t.} \sum_{s\in S} x_{s,d} \leqslant 1,\ \forall\, d \in D \tag{1b}$$

$$\sum_{d\in D\cup\{0\}} x_{s,d} = 1,\ \forall\, s \in S \tag{1c}$$

$$x_{s,d} \in \{0,1\},\ \forall\, s \in S, d \in D \cup \{0\} \tag{1d}$$

Observe that the first term in the matching weights captures a "throughput" value, dividing the number of orders in a delivery route, N_s, by the time required to execute the assignment, $\max_{o\in s}\{\delta^s_{o,d}\} - e_d$. Meanwhile, the second term relates to the pick-up delay, which causes a loss of freshness in meals, $\pi_{s,d} - \max_{o\in s}\{e_o\}$. The model formulation makes use of an artificial courier collecting excess bundles (by defining these artificial assignments to have a value strictly lower than the value of any feasible assignments).

3.2.2 More complex integer programming assignment models

A relatively simple but powerful departure from the previous formulation is to drop the requirement that the set of bundles in the assignment problem be pairwise disjoint with respect to the set U of orders to be assigned. Instead, that condition can be enforced as a constraint in the optimization problem. The freedom thus gained can be leveraged to mitigate some potential pitfalls inherent in the process of assigning (at most) one bundle to each courier in every optimization run.

Let $\mathbb{S}$ be the set of bundles considered for assignment, which can simply be the S of bundles generated by Procedure 1, or it may be larger. A simple and effective way to create more routes is to break each bundle of S into two pieces, as described in Procedure 2:

```
Procedure 2: Split original bundles by their ready times
Input: Set of bundles $S$.
Output: $\mathbb{S}$, ground set of not necessarily order-disjoint bundles.
Initialize $\mathcal{S}^1 \leftarrow \emptyset$, $\mathcal{S}^2 \leftarrow \emptyset$;
for $s \in S$ do
    Sort orders in $s$ by increasing ready time, and relabel as $o_1, \ldots, o_k$ ($o_k$ is ready the latest);
    for $j \in \{1, \ldots, k-1\}$ do
        Create bundle $s_j^1$, containing $o_1, \ldots, o_j$, the first $j$ orders to be ready from $s$. Preserve
         sequence of deliveries originally prescribed in $s$ for these orders;
        $\mathcal{S}^1 \leftarrow \mathcal{S}^1 \cup s_j^1$;
        Create bundle $s_j^2$, containing $o_{j+1}, \ldots, o_k$, which are ready after $o_j$. Preserve the
         sequence of deliveries originally prescribed in $s$ for these orders;
        $\mathcal{S}^2 \leftarrow \mathcal{S}^2 \cup s_j^2$;
    end
end
$\mathbb{S} \leftarrow S \cup \mathcal{S}^1 \cup \mathcal{S}^2$;
```

The motivation of Procedure 2 is to prevent the assignment of large bundles in S (which are less likely to be ready for immediate pick-up) from unnecessarily delaying the dispatch of orders that could be delivered soon (bundles in $\mathcal{S}^1$), while at the same time attempting to find a courier (and, if necessary, make a partial commitment) for orders that will not be ready as soon (bundles in $\mathcal{S}^2$).

Another extension is to allow two bundles to be assigned to a courier, in which case the courier completes the bundles one after the other in a prescribed sequence. This can mitigate a potential pitfall of the myopic approach, arising when two bundles are assigned to different couriers even though they could have been delivered more efficiently by a single courier without degrading service quality. Define $\mathcal{Q} \subseteq \mathbb{S} \cup \mathbb{S} \times \mathbb{S}$ as the set of one or two bundles that can be assigned to a courier. $\mathcal{Q}$ is constructed by finding all pairs of bundles in $\mathbb{S}$ that can be concatenated in such a way that second bundle does not suffer an excessive freshness loss. We define a freshness loss tolerance $\lambda \geqslant 0$, and follow Procedure 3:

```
Procedure 3: Find compatible "follow-on" bundle pairs
Input: Ground set of bundles $\mathbb{S}$, freshness loss tolerance $\lambda \geqslant 0$.
Output: Set $\mathcal{Q} \subseteq \mathbb{S} \cup \mathbb{S} \times \mathbb{S}$ of routes to be assigned to couriers.
Initialize $\mathcal{Q} \leftarrow \mathbb{S}$;
for $s \in \mathbb{S}$ do
    for $s' \in \mathbb{S} \setminus \{s\}$ do
        if $s \cap s' = \emptyset$ then
            Compute $\pi_{s,s',d}$;
            if $r_{s'} + \lambda \geqslant \pi_{s,s',d}$ then
                $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(s, s')\}$;
            end
        end
    end
end
```

Note that if $\lambda = 0$, only pairs of bundles where the second pick-up can be done at the earliest possible time are considered compatible. In contrast, if λ is large enough, any pair of bundles with no orders in common is a compatible "follow-on" pair.

Assignment model variations. For any order o, denote the set of routes that contain o by $\mathcal{Q}(o)$ and define variables:

$$x_{q,d} \in \{0,1\} : \text{indicates whether route } q \text{ is assigned to courier } d, \text{ for } q \in \mathcal{Q},\ d \in D \cup \{0\}$$
$$y_o \in \{0,1\} : \text{indicates whether order } o \text{ is assigned to any courier, for } o \in U$$

Using these definitions, the assignment problem can be formulated as:

$$\max_x \sum_{d\in D}\sum_{q\in\mathcal{Q}} w_{q,d}x_{q,d} + \sum_{o\in O} p(1-y_o) \tag{2a}$$
$$\text{s.t.} \sum_{q\in\mathcal{Q}} x_{q,d} \leqslant 1 \qquad \forall\, d \in D \tag{2b}$$
$$\sum_{d\in D\cup\{0\}} x_{q,d} = 1 \qquad \forall\, q \in \mathcal{Q} \tag{2c}$$
$$\sum_{d\in D}\sum_{q\in\mathcal{Q}(o)} x_{q,d} \leqslant 1 \qquad \forall\, o \in U \tag{2d}$$
$$\sum_{d\in D}\sum_{q\in\mathcal{Q}(o)} x_{q,d} \geqslant y_o \qquad \forall\, o \in U \tag{2e}$$
$$x_{q,d} \in \{0,1\} \qquad \forall\, q \in \mathcal{Q}, d \in D\cup\{0\} \tag{2f}$$
$$y_o \in \{0,1\} \qquad \forall\, o \in U \tag{2g}$$

Constraints (2b) guarantee that each courier is assigned at most one route, while (2c) guarantee that each route is assigned to one courier (it may be 0, a "dummy" courier). Constraints (2d) guarantee that each order is in exactly one of the routes assigned. If an order $o \in O$ is not included in any of the routes assigned to real couriers, a value p is added to the objective value (with $p < \min_{q,d}\{w_{q,d}\}$, to discourage null order assignments if at all possible, *i.e.* to assign as many orders as possible to real couriers). The weight of a (q,d) assignment is $w_{q,d} = u_{s,d}$ if q consists of only one bundle s, or $w_{q,d} = u_{s,d} + v_{s,s',d}$ if q consists of a sequence of bundles (s,s'), where u and v are defined as:

$$u_{s,d} = \frac{N_s}{\max_{o\in s}\{\delta^s_{o,d}\} - e_d} + \theta(\pi_{s,d} - \max_{o\in s}\{e_o\}) \qquad \forall s \in \mathcal{S},\ d \in D \tag{3}$$
$$v_{s,s',d} = \frac{N_{s'}}{\max_{o\in s'}\{\delta^{s,s'}_{o,d}\} - \max_{o\in s}\{\delta^s_{o,d}\}} + \theta(\pi_{s,s',d} - \max_{o\in s'}\{e_o\}) \qquad \forall (s,s') \in \mathcal{Q},\ d \in D \tag{4}$$

Here N_s and $N_{s'}$ denote the number of orders in bundles s and s', respectively; e_d denotes the time that courier d is available to begin a new route; $\max_{o\in s}\{\delta^s_{o,d}\}$ is the last drop-off time in s if d is assigned to s; $\max_{o\in s'}\{\delta^{s,s'}_{o,d}\}$ is the last drop-off time in s' if this bundle is served by d immediately after completion of deliveries of s; $\max_{o\in s}\{e_o\}$ is the time that s is ready for pick-up; $\pi_{s,d}$ is the earliest possible time that d can pick up s; and $\pi_{s,s',d}$ is the earliest possible time that d can pick up s' after having picked up and delivered s.

We have explored two alternative models in addition to the (basic) linear model:

(i) A "medium complexity" model, consisting of integer program (2) and $\mathcal{Q}$ defined by Procedure 3 acting on S, the bundles created by Procedure 1.

(ii) A "high complexity" model, consisting of integer program (2) and $\mathcal{Q}$ defined by Procedure 3 acting on $\mathbb{S}$, the set of bundles resulting from Procedure 2.

3.3 Commitment strategy

One of the ways through which a rolling horizon algorithm attempts to mitigate uncertainty is the postponement of decisions that are not time-critical. In light of the characteristics of the MDRP, we adopt a "lazy" strategy that decomposes assignments into two travel segments: "inbound" travel to the restaurant, and an "outbound" delivery trip (*i.e.*, an open route).

Two-stage additive commitment. For each tentative assignment (s, d), of order bundle s and courier d, in the solution of a matching problem, the commitment strategy dictates:

1. If d can reach restaurant r_s before $t + f$ and all orders in s are estimated to be ready by $t + f$, make a *final commitment* of d to s: instruct d to travel to r_s, pick up and deliver orders in s.
2. If d cannot reach r_s by $t + f$, but completes its last assignment before $t + f$, make a *partial commitment*: instruct d to travel to r_s and wait there for a finalized order assignment, which is guaranteed to include orders in s, and possibly more.
3. If d cannot start a new assignment by $t + f$, ignore the assignment.
4. *Exception:* If any order in s has been ready for more than x minutes, override the rule and make a final commitment.

The motivation to send a courier to a restaurant without committing the delivery of a specific bundle – as would be the case in a simpler single-stage commitment rule – is that, even if travel should begin without delay, the courier can be matched again in the next optimization, while en-route, and the composition of the bundle to be picked up at the restaurant may change. On the other hand, if the courier is busy until $t + f$, waiting for the next optimization will not delay the pick-up or drop-off of any order.

We call this strategy "additive" because if s is partially committed to d at time t, then at optimization time $t + f$, we force the bundle assigned to d to include s. Hence, orders in a bundle partially assigned to d are guaranteed to be the final assignment to d. While this is not the most flexible policy that can be conceived (as one could completely decouple inbound and outbound assignments), there are good practical arguments for such consistency (*e.g.*, to mitigate the risk of couriers rejecting time-critical assignments).

Two-stage additive commitment logic for assignments with more than one bundle. In the more complex assignment models, when dealing with the assignment of courier d to route $q = (s_1, s_2)$, the commitment rule is generalized in a way that delays final decisions on the actions to be taken after the completion of s_1, so long as this does not generate a decrease in the service quality of orders in s_2. Concretely, given a tentative assignment of order batch $q = (s_1, s_2)$ to courier d, our commitment strategy dictates:

1. If d can complete the delivery of s_1 *and* pick up s_2 by time $t + f$, make a final commitment to both s_1 and s_2.
2. If d can complete the delivery of s_1 by $t+f$ but cannot pick up s_2 by $t+f$, make a final commitment to s_1 and a partial commitment to s_2.
3. If d can only pick up s_1 by $t + f$, *i.e.*, if d can reach restaurant r_{s_1} by $t + f$ and all orders in s are estimated to be ready by $t + f$, make a *final commitment* of d to s_1: instruct d to travel to r_{s_1}, pick up and deliver orders in s_1.
4. If d cannot pick up s_1 by $t + f$, but completes its last assignment before $t + f$, make a *partial commitment* for d: instruct d to travel to r_{s_1} and wait there for a finalized order assignment (guaranteed to include orders in s_1.
5. If d cannot start a new assignment by $t + f$, ignore the assignment completely.
6. *Exception 1:* If any order in s_1 has been ready for more than x minutes, make a final commitment to s_1.

7. *Exception 2:* If any order in s_2 has been ready for more than x minutes, make a final commitment to s_1 and s_2.

It is worth noting that our two-stage additive commitment strategy is consistent with the MDRP operating environment that allows assignment updates: a courier may be instructed to move to a restaurant as soon as there is at least one order for the courier to pick up, but before a final determination is made about the exact set or sequence of orders to be assigned. Furthermore, a single pre-positioning move is used by our algorithm: at the beginning of each courier duty time, a randomized instruction is handed to the courier, telling him to move to a nearby restaurant (the rest of instructions are deterministic, only making use of information already revealed.)

4 Computational study

In this section, we describe the design and results of a computational study to assess the quality of solutions produced by our algorithm (in its different variations), and how this performance relates to instance characteristics and key algorithmic features. In a set of appendices, we present visualizations of our instances as well as a cache of experimental results.

4.1 MDRP instances

Instances have been crafted to resemble realistic day-long order and courier patterns in metropolitan areas, based on real-life historical data from different cities and days. We create a total of 240 instances of varying sizes, ranging from 242 to 3213 orders, 54 to 323 restaurants, 53 to 457 couriers, and 123 to 1542 courier hours. Times are represented as non-negative integers (with zero representing the start of business hours). Locations have been anonymized while preserving their overall distribution, and are represented as (x, y) coordinates in meters from a reference point. The travel time from $\ell_1 = (x_1, y_1)$ to $\ell_2 = (x_2, y_2)$ is the product of their Euclidean distance and a multiplier γ, rounded up to the next integer: $t_{\ell_1,\ell_2} = \left\lceil \gamma\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \right\rceil$ (*i.e.*, $\frac{1}{\gamma}$ is the speed of vehicles in the instance). Note that rounding may lead to cases where travel times do not satisfy the triangle inequality.

Service times at all restaurants are set to 4 minutes, and service times at all order delivery locations are likewise set to 4 minutes. The click-to-door target is 40 minutes in all instances, and the maximum allowed click-to-door is 90 minutes. Couriers are paid \$10 per order delivered and are promised a \$ 15 minimum compensation per hour.

For each "seed" instance, we have prepared a set of 24 variations:

- **Varying size of order and courier sets.** Given an order set, two instance variations are obtained by sampling 50% of the orders directly (uniform sampling with repetitions) or indirectly, through sampling of restaurants (where including a restaurant means including all the orders placed at it). Courier sets are reduced in a similar fashion, sampling roughly half of the shifts at each start time (rounding up to the nearest integer number of shifts when necessary), while ensuring that the scheduled courier hours are reduced by 50%. These variations preserve some of the geographical and temporal distribution of orders. A detailed exposition about the reduction procedures for order and courier sets is available in Appendix A.
- **Varying travel times.** Different location densities can be induced by using different γ multipliers. This preserves the relative spatial distribution of orders but, of course, impacts the nature of the solutions, as travel time changes (*e.g.*, orders served per courier hour, number of bundled orders, etc.). Concretely, to build these instance variations, the original multiplier of each instance is reduced by 25%, producing a short travel time version.
- **Varying structure of courier schedules.** Schedules can resemble the patterns observed historically, or follow an "optimized" distribution. To ensure fairness in comparisons, optimized schedules preserve the total number of courier-hours of their historical counterparts. Details about the con-

struction of optimized shift schedules are available in Appendix B.

- **Varying preparation times.** We can modify the degree of flexibility in the dispatch of upcoming orders in an instance by changing the time it takes to prepare each meal, i.e. the difference between the time when an order is ready and its placement time: given a target click-to-door time, longer preparation times imply that less time is available for a timely pickup once the order becomes ready In our instance set, these instance variations are built by increasing the original preparation times by 25%.

To facilitate the study of different algorithms and solution strategies, the MDRP instances, together with a script to check feasibility and evaluate performance metrics of a solution, are available online at:

`https://github.com/grubhub/mdrplib`

Details about instance and solution encoding can be found in Appendices C and D.

4.2 Uncontrolled instance features

Beyond the features controlled during the instance generation process, we have focused our analysis on a set of structural properties whose importance has been long recognized in the dynamic routing literature [26]. We measure four features for each instance: geographic dispersion, dynamism, urgency and flexibility.

- **Geographic dispersion** captures the separation between restaurant and delivery locations over the instance geography. Other things equal, the more disperse locations are, the longer it takes to complete an average route, and the harder it becomes to build routes that achieve an acceptable performance (in order click-to-door, courier utilization, etc.). We measure geographic dispersion in terms of the travel time from restaurant to delivery location of any order, and the travel time between any pair of restaurants, using the instance mean and standard deviation across all relevant distance travel time measures. More precisely, we define the dispersion of an instance as:

$$dispersion = \frac{\sum\limits_{a\in R, b\in R} t_{\ell_a,\ell_b}}{|R \times R|} + \frac{\sum\limits_{o\in O} t_{\ell_{r_o},\ell_o}}{|O|}$$

- **Dynamism** captures the continuity of change in the information available over the planning period. Under this definition, an instance where orders arrive in a few "bursts" is considered less dynamic than an instance where orders arrive at an even rate over time. For simplicity, we measure only the degree of dynamism of the arrival stream of orders (disregarding information from courier sign-on and sign-off events), using the method introduced in Van Lon et al. [26], which we reproduce below for the sake of completeness. The degree of dynamism of an instance is always a number between 0 (no dynamism, *i.e.*, all orders are simultaneously revealed) and 1 ("maximal" dynamism), computed as follows.

  Let n be the number of orders placed and A be the non-decreasing sequence of order placement times. Let H be the sequence of inter-arrival times, $H = (\eta_1, \eta_2, \ldots, \eta_{n-1}) = (a_2 - a_1, \ldots, a_n - a_{n-1})$. Let $T = \min\{\max\limits_{c\in C}\{l_c\}, a_n\} + \tau_{\max}$ be regarded as the operating period, and define $\varphi = \frac{T}{n}$ as the "perfect inter-arrival time" (corresponding to arrivals with maximum dynamism). Define the sequence of penalized inter-arrival deviations as

$$\sigma_0 = 0, \quad \sigma_i = (\varphi - \eta_i + \frac{\varphi - \eta_i}{\varphi}\sigma_{i-1})^+ \text{ for } i = 1, \ldots, n-1$$

  Note that this measure penalizes "bursts" of arrivals over a short period. Then, the instance degree of dynamism is defined as:

$$dod = 1 - \frac{\sum\limits_{i=1}^{n-1} \sigma_i}{\sum\limits_{i=1}^{n-1} \bar{\sigma}_i}$$

where $\sum\limits_{i=1}^{n-1} \bar{\sigma}_i$ is a normalizing constant defined by $\bar{\sigma}_i = \varphi + (\frac{\varphi - \eta_i}{\varphi}\sigma_i)^+$ (capturing the maximum possible penalized deviations).

- **Urgency** captures the range of time available to complete the delivery of orders in a satisfactory way. We define "soft" and "hard" measures, relative to target and maximum click-to-door values, respectively. For clarity, we use "reaction time" measures, negatively related to urgency: the higher the reaction time of orders, the less urgency in the instance. As in [26], we summarize the urgency level of an instance by the mean of individual order reaction times:

$$react_{soft} = \frac{\sum\limits_{o \in O} \left(\tau - t_{\ell_{r_o}, \ell_o}\right)^+}{|O|}$$

$$react_{hard} = \frac{\sum\limits_{o \in O} \left(\tau_{\max} - t_{\ell_{r_o}, \ell_o}\right)^+}{|O|}$$

- **Flexibility** is closely related to urgency, but not completely equivalent. It captures the effective range of time available to dispatch an order, if this is going to be delivered in a satisfactory way. Measuring flexibility is important in meal delivery, because relatively long preparation times (with respect to service targets and guarantees) can make it hard to deliver an order on time, even if there is a large reaction time: if the order is not ready, a part of this reaction time is useless. We measure flexibility as follows:

$$flex_{soft} = \frac{\sum\limits_{o \in O} \left[(a_o + \tau) - (e_o + t_{\ell_{r_o}, \ell_o})\right]^+}{|O|}$$

$$flex_{hard} = \frac{\sum\limits_{o \in O} \left[(a_o + \tau_{\max}) - (e_o + t_{\ell_{r_o}, \ell_o})\right]^+}{|O|}$$

4.3 Analysis variables

Apart from categorical identifiers for each set of instance variations, and in addition to dispersion and degree of dynamism, we select the most informative measures of urgency, flexibility, and size by exploring the correlations in the instance dataset.

It is not surprising that the soft and hard versions of urgency and flexibility measures are correlated, as shown in Table 1. Therefore, to simplify the analysis, we focus only on hard reaction time and soft pickup flexibility, the pair of urgency and flexibility measures with the smallest correlation. Similarly, number of orders and total courier hours are highly correlated, as shown in Table 2. Since courier hours in an instances have been decided in advance, based on forecasts about the number and distribution of orders, we decide to measure instance size by the number of orders, discard total courier hours from the analysis variables, and include the ratio of orders to courier hours, which indicates how "busy" or congested an instance turns out to be (instances with a unusually high ratio suggest that the forecast might have been an underestimation, or perhaps there were not enough couriers to keep up with expected demand).

measure	soft reaction time	hard reaction time	soft pickup flex	hard pickup flex
soft reaction time	1.00	0.62	0.89	0.94
hard reaction time	0.62	1.00	0.41	0.46
soft pickup flex	0.89	0.41	1.00	0.96
hard pickup flex	0.94	0.46	0.96	1.00

Table 1: Correlation matrix of urgency and flexibility measures

measure	orders	courier hours	orders/courier hour
orders	1.00	0.90	0.31
courier hours	0.90	1.00	-0.06
orders per courier hour	0.31	-0.06	1.00

Table 2: Correlation matrix of orders, courier hours, and orders per courier hour

Having defined the analysis variables, let us briefly summarize the structural properties of the instance set. The histograms in the diagonal of Figure 1 illustrate the marginal distribution of the adopted key instance characteristic measures (aggregated in 8 buckets), and the hexagonal bins in the lower triangle of Figure 1 illustrate their pairwise-joint distribution.

Instances have degrees of dynamism ranging between 0.3 and 0.6, more frequently in the 0.4-0.55 range, and the most significant interactions of dynamism are with size (largest instances are also the least dynamic) and geographic dispersion (less disperse instances tend to be less dynamic). Most instances in the set have less than 1000 orders and a relatively dense geography (in most instances, individual-order inbound and outbound trips would on average total less than 20 minutes). Hard reaction times values range from 80.7 to 84.5 minutes on average, and shows some correlation only with geographic dispersion: this is not unexpected, as both measures are a function of direct travel times from restaurant to delivery location. Soft pickup flexibility, ranging from 10.4 to 19.4 minutes, exhibits no strong correlation to any other feature. The ratio of orders placed to courier hours, ranging from 1.2 to 2.5, shows little correlation with all other instance features, but it is important to note that the largest instances (2500 or more orders) all have a very high ratio.

4.4 Algorithm configurations

For each instance, we obtain 21 solutions by running slightly different versions of the rolling horizon algorithm: the default algorithm, 5 variations differing in one feature only, and 15 variations with alternative values of optimization frequency, assignment horizon, order and courier bundling intensity lookaheads. In detail, the algorithm variations explored are defined as follows:

Different optimization frequencies. Set the time between successive optimizations to $f = 5$ or $f = 2$ minutes (all other settings at their default values).

Different order-assignment horizon and myopic lookaheads. Experiment setting a lookahead of $2f$ or $4f$ (default) minutes for the numerator (orders) of the target size ratio calculation; setting a lookahead of $2f$ or $4f$ (default) minutes for the denominator (couriers) of the target size ratio calculation; and setting a horizon of $2f$ (default) or $4f$ minutes to limit the set of orders open for assignment. In sum, for each value of f, 8 configurations are considered.

Bundling intensity rules. Explore two algorithm variations, one using dynamic target bundle sizes (default), and the other explicitly forbidding any consolidation (*i.e.*, always assign couriers to single orders).

Assignment prioritization. Enforce three order priority groups in the assignment process (default), or find assignments for all orders in a single matching problem per optimization run.

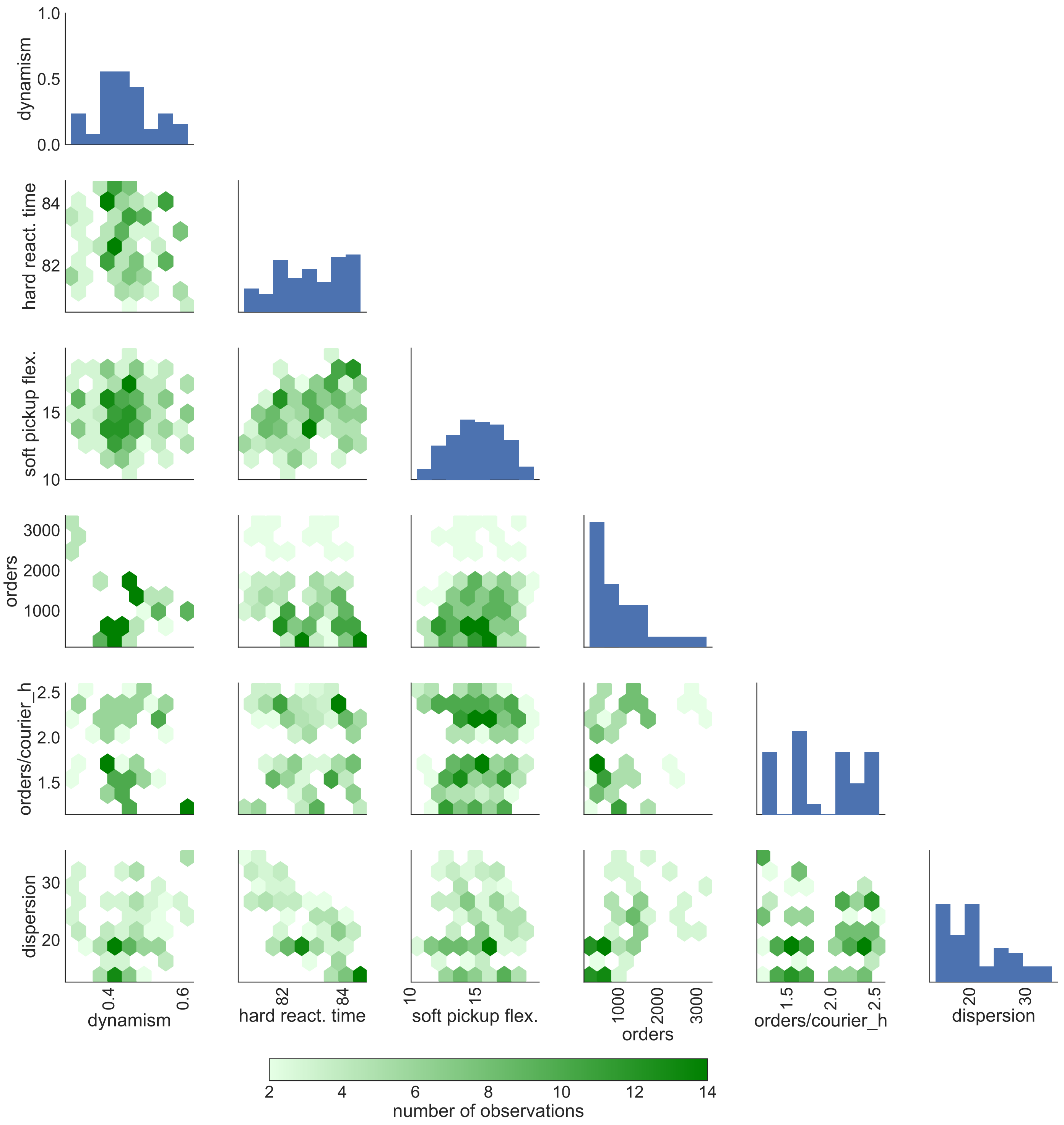

Figure 1: Distribution of key instance features

Commitment. Follow the two-stage additive commitment strategy (default), or use a single-stage lazy strategy, where no more changes can be made in the "outbound" portion of the assignment as soon as couriers begin the"inbound" movement to the restaurant.

Complexity of assignment model. Consider the three different models described in Section 3.2 to govern the assignment process.

Before conducting the experiments, reasonable default values for a series of secondary parameters in the algorithm must be found. We tune the algorithm with a hierarchical service objective: first ensure the

Variation id	Description
0	Default ($f = 5, \Delta_u = 10, \Delta_1 = 10, \Delta_2 = 10$, two-stage commitment, 3 priority groups, dynamic bundling, LP assignment model)
1	$f = 5, \Delta_u = 10, \Delta_1 = 10, \Delta_2 = 20$
2	$f = 5, \Delta_u = 10, \Delta_1 = 20, \Delta_2 = 10$
3	$f = 5, \Delta_u = 10, \Delta_1 = 20, \Delta_2 = 20$
4	$f = 5, \Delta_u = 20, \Delta_1 = 10, \Delta_2 = 10$
5	$f = 5, \Delta_u = 20, \Delta_1 = 10, \Delta_2 = 20$
6	$f = 5, \Delta_u = 20, \Delta_1 = 20, \Delta_2 = 10$
7	$f = 5, \Delta_u = 20, \Delta_1 = 20, \Delta_2 = 20$
8	$f = 2, \Delta_u = 4, \Delta_1 = 4, \Delta_2 = 4$
9	$f = 2, \Delta_u = 4, \Delta_1 = 4, \Delta_2 = 8$
10	$f = 2, \Delta_u = 4, \Delta_1 = 8, \Delta_2 = 4$
11	$f = 2, \Delta_u = 4, \Delta_1 = 8, \Delta_2 = 8$
12	$f = 2, \Delta_u = 8, \Delta_1 = 4, \Delta_2 = 4$
13	$f = 2, \Delta_u = 8, \Delta_1 = 4, \Delta_2 = 8$
14	$f = 2, \Delta_u = 8, \Delta_1 = 8, \Delta_2 = 4$
15	$f = 2, \Delta_u = 8, \Delta_1 = 8, \Delta_2 = 8$
16	single-stage commitment
17	no priority groups
18	no bundling
19	medium complexity assignment model
20	high complexity assignment model

Table 3: Algorithm variations

delivery of a very high proportion of the orders placed, and then minimize average click-to-door. Details about the tuning process can be found in Appendix E.

4.5 Experiment results

As a preamble to our analysis of experimental results, we benchmark the solutions obtained by our algorithm on a subset of instances with relatively few orders and couriers with the solutions obtained by the exact algorithm presented in [30], which assumes a clairvoyant decision maker, i.e., all information about orders and couriers is known in advance.

4.5.1 Comparison of algorithm with exact method

The solutions reported in [30] have been obtained by solving an integer program with minimization of the click-to-door time as objective. They are all provably optimal or near-optimal (maximum optimality gap is 0.16%, *i.e.*, a couple of seconds away from the lower bound). In the integer program, delivery of all orders is enforced as a feasibility condition. It must be noted that the formulation in [30] restricts bundle sizes to be at most 2, however, we have reason to believe that the effect of this restriction is very small: in the solutions produced by our algorithm for these instances only an average of 3% of the total number of orders are part of bundles of size greater than 2.

As shown in Table 4, on this set of small instances, our algorithm produces high-quality solutions with respect to mean click-to-door time, on average only 4% above the optimal value (about 70 seconds, in absolute terms). At the same time, mean ready-to-pickup time performance of our algorithm is on average 112% (45 seconds) above the value of the exact solutions, even though this measure has not

instance	orders	restaurants	couriers	courier hours	Rolling Horizon Heuristic		Exact Algorithm (minimizing total CTD)		Heur/Exact	
					CtD	RtP	CtD	RtP	CtD	RtP
0o50t100s1p100	252	93	61	151	31.19	2.52	29.81	1.46	1.05	1.73
0o50t100s1p125					34.67	2.27	33.40	1.23	1.04	1.85
0o50t100s2p100			72	146	29.79	1.22	28.96	0.64	1.03	1.91
0o50t100s2p125					34.18	1.85	32.60	0.53	1.05	3.49
0o50t75s1p100			61	151	28.4	1.65	27.49	0.96	1.03	1.72
0o50t75s1p125					31.62	1.19	31.10	0.78	1.02	1.53
0o50t75s2p100			72	146	27.29	0.58	26.83	0.30	1.02	1.93
0o50t75s2p125					31.19	0.7	30.52	0.24	1.02	2.92
0r50t100s1p100	242	54	61	151	32.46	2.14	30.76	1.13	1.06	1.89
0r50t100s1p125					36.75	2.16	34.66	0.97	1.06	2.23
0r50t100s2p100			73	146	31.21	1.11	29.93	0.43	1.04	2.58
0r50t100s2p125					35.6	1.22	33.86	0.26	1.05	4.69
0r50t75s1p100			61	151	29.57	1.04	28.47	0.67	1.04	1.55
0r50t75s1p125					33.71	1.19	32.45	0.55	1.04	2.16
0r50t75s2p100			73	146	29.03	0.64	27.95	0.21	1.04	3.05
0r50t75s2p125					33.41	0.84	32.00	0.16	1.04	5.25
average					31.879	1.40	30.674	0.66	1.04	2.53

Table 4: Comparison of solutions obtained by our algorithm and exact solutions

been directly optimized.

More challenging large-scale instances may exhibit important differences from the small instances in this benchmark (*e.g.*, more bundling opportunities), and there is no guarantee that performance will follow a similar trend. That being said, these preliminary results are certainly encouraging, as they are consistent with the heuristic argument for a rolling-horizon approach: given that dispatching decisions are heavily time-constrained, information about the distant future tends to be of little value.

4.5.2 Performance effects of controlled instance characteristics

Tables 5-8 summarize the impact of each of the controlled instance variations on performance metrics over all experiment runs (*i.e.*, over all algorithm variations). Figures 11-24 in Appendix F illustrate these controlled effects in more detail.

Table 5 shows that leveraging accurate predictive information with off-line optimization techniques, in order to schedule the right capacity at the right time, can yield significant improvements in all service metrics, while reducing performance variability, and even reducing costs. Most dramatically, optimizing shifts can reduce the number of orders undelivered by 60% and ready-to-pickup mean by 40%. We also note that with historical courier shifts the number of orders per bundle increases, which is consistent with the algorithm using bundles as a recourse to mitigate capacity shortages: the more couriers signed on at productive times of the day, the more effective capacity over time, and the smaller the target bundle sizes.

Table 6 summarizes the performance sensitivity of the algorithm to vehicle speed. If travel times are 25% faster, orders undelivered can be reduced by more than half, while improving mean click-to-door times by 10%, and slashing down the overage mean by more than a third and the ready-to-pickup mean

	courier shift schedules				paired differences (opt. - hist.)	
	historical		optimized			
	mean	std	mean	std	mean	std
% undelivered	0.42	0.91	0.16	0.28	-0.25	0.90
CtoD mean	33.7	3.78	32.38	3.03	-1.32	1.46
CtoD overage	3.21	1.84	2.64	1.36	-0.57	0.93
RtoP mean	2.79	2.08	1.64	1.09	-1.15	1.23
utilization mean	0.63	0.12	0.64	0.12	0.01	0.02
cost per order	11.58	1.34	11.22	1.15	-0.35	0.31
orders per bundle	1.11	0.07	1.08	0.06	-0.02	0.03

Table 5: Differences in performance of instances with optimized courier schedules vs. historical schedules

	travel time multiplier				paired diff. (75% tt. -100% tt.)	
	100%		75%			
	mean	std	mean	std	mean	std
% undelivered	0.36	0.88	0.22	0.4	-0.14	0.66
CtoD mean	34.79	3.35	31.29	2.64	-3.51	1.33
CtoD overage	3.54	1.8	2.31	1.18	-1.23	0.82
RtoP mean	2.95	2	1.48	1.04	-1.47	1.18
utilization mean	0.68	0.12	0.6	0.11	-0.08	0.02
cost per order	11.34	1.25	11.47	1.27	0.13	0.17
orders per bundle	1.11	0.07	1.08	0.05	-0.02	0.03

Table 6: Differences in performance of instances with travel times faster by 25% vs original

by almost half. On the other hand, however, total costs increase, as the work in the system increases (orders undelivered drop) but is distributed less evenly (utilization mean and 10th percentile drop) and the system is forced to spend more on minimum compensation complements. We believe this illustrates the downside of having "too many" drivers at the wrong times of the day: when travel times are slower, routes take longer to complete and, at any point in time, more vehicles need to be on the road (as orders have not slowed down their arrival and the same service standard must be met), thus leading the algorithm to pull couriers that are available at a remote location into the action. On the other hand, when travel times are faster, there is a relative surplus of couriers but the algorithm has little incentive to put them to work if they are not in a central location, as this may degrade service quality and potential cost-savings related to the compensation scheme are not considered by the algorithm.

Table 7 illustrates the sensitivity of performance with respect to preparation time. If orders take 25% longer to be ready, orders undelivered almost double, while click-to-door deteriorates by 12% on average, driving up the overage mean up by 80%. Interestingly, longer preparation times do not force the algorithm to sacrifice ready-to-pickup time performance significantly, which indicates that the observed increase in click-to-door time is largely unavoidable. Furthermore, despite completing fewer deliveries (thus reducing total delivery payments), cost per order improves only slightly, as a consequence of having couriers scheduled 'out-of-phase', logging-on for duty during periods of low productivity at the beginning of lunch and dinner times (thus increasing the total minimum compensation payments).

Table 8 illustrates the performance effect of halving the number of orders and courier hours, either by thinning the overall order arrival process (`o50`) or by sampling a large enough subset of restaurants (`r50`). By and large, instances of original size tend to have better service performance than their reduced counterparts (thanks to the higher location density), at a slightly higher cost (a product of

	preparation time multiplier				paired differences	
	100% prep. time		125% prep. time		(125% pt. - 100% pt.)	
	mean	std	mean	std	mean	std
% undelivered	0.2	0.6	0.38	0.76	0.18	0.31
CtoD mean	31.1	2.86	34.98	2.94	3.88	0.47
CtoD overage	2.07	1.25	3.78	1.54	1.71	0.46
RtoP mean	2.21	1.74	2.22	1.77	0.01	0.31
utilization mean	0.64	0.12	0.64	0.12	0.00	0.01
cost per order	11.41	1.25	11.39	1.27	-0.01	0.12
orders per bundle	1.09	0.06	1.1	0.06	0.00	0.01

Table 7: Differences in performance of instances with preparation times slower by 25% vs original

	instance size reduction						paired differences			
	o100		o50		r50		(o50-o100)		(r50 - o100)	
	mean	std	mean	std	mean	std	mean	std	mean	std
% undelivered	0.2	0.4	0.38	0.95	0.28	0.57	0.19	0.80	0.08	0.39
CtoD mean	32.55	3.17	33.29	3.75	33.27	3.47	0.74	1.32	0.72	1.52
CtoD overage	2.75	1.47	2.98	1.81	3.05	1.62	0.24	0.81	0.30	0.76
RtoP mean	1.68	1.28	2.61	2.13	2.36	1.61	0.94	1.16	0.69	0.66
utilization mean	0.62	0.12	0.66	0.12	0.64	0.12	0.04	0.02	0.03	0.02
cost per order	11.5	1.31	11.28	1.2	11.42	1.25	-0.22	0.19	-0.09	0.21
orders per bundle	1.1	0.06	1.06	0.05	1.12	0.07	-0.04	0.03	0.01	0.02

Table 8: Differences in performance of instance size reductions vs original

actually executing more deliveries). Furthermore, we highlight that in reduced instances, at any point in time there tend to be fewer restaurants with orders in preparation, and it takes longer on average for couriers to drive to the next pick-up (even if delivery locations are at a similar average distance from their associated restaurant). This is consistent with the results which show that, relatively, ready-to-pickup measures degrade considerably more than click-to-door measures when instances are reduced: the algorithm struggles to find couriers that can make it on time to pick-up, but can still build routes that mitigate the delay impact on click-to-door.

Focusing now on the difference between both reduction types, `r50` instances fare better than `o50` instances on orders undelivered, utilization and ready-to-pickup measures, while in the case of click-to-door, `r50` instances fare better on the average measure, but worse in worst-case measures. `o50` instances achieve a lower cost per order than `r50` instances, but only because the algorithm cannot complete as many deliveries. To interpret these trends, consider that `o50` instances tend to have less orders per restaurant than `r50` (and `o100`) instances, *i.e.*, less consolidation opportunities than `r50` instances. Consistently, the results show that `o50` instances make less use of bundling than the original instances (because it is difficult), while `r50` instances use bundling more intensely than the original instances (because it is necessary, to cope with higher geographic dispersion) and `o50` instances (because there are more opportunities.)

4.5.3 Performance effect of instance structural properties

Figure 2 summarizes the marginal effect of each instance feature on the set of performance measures over all experiment runs with the default algorithm settings (We focus on this variant in order to simplify the analysis, as we have no direct control over the structural properties being studied).

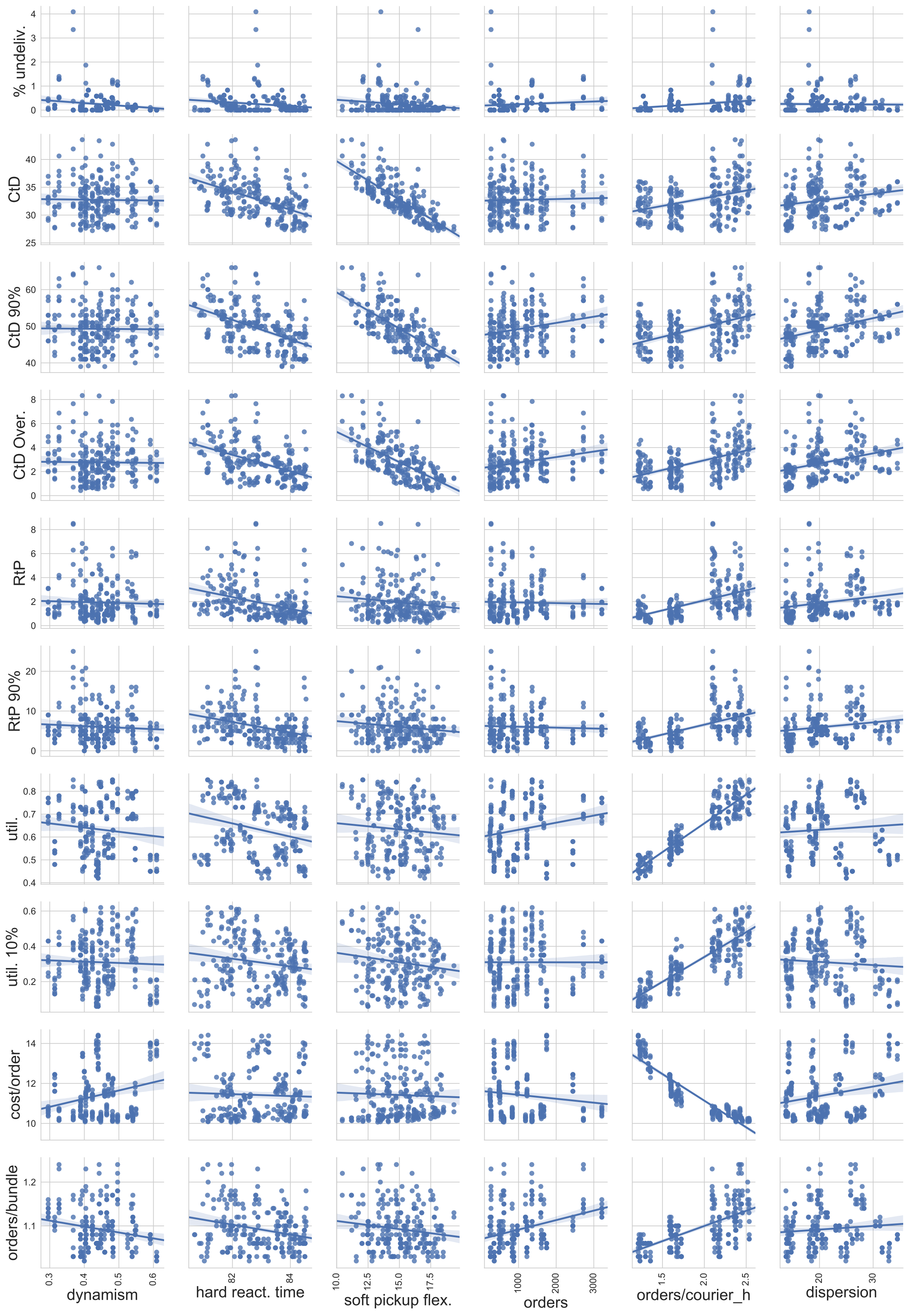

Figure 2: Effect of each instance feature on overall performance

As dynamism increases, utilization and orders per bundle tend to decrease, while cost per order tends to increase. This is expected, as consolidation opportunities that help defray delivery costs are more likely to be available when orders arrive in bursts. Except for a couple of outliers (illustrated in more depth in Table 9), the percentage of orders undelivered is always in the 0%-2% range, and tends to decrease slightly with an increasing degree of dynamism. Moreover, instances with very low and very high dynamism achieve a more consistent percentage of orders undelivered than instances with mid-range dynamism. A similar variability pattern is observable for click-to-door and ready-to-pickup measures, which suggests that that there is another factor at play, not discernible in Figure 1 and Figure 2, driving this behavior. On closer inspection, the outliers and the interaction plots in Appendix G show that a high percentage of orders undelivered always corresponds to a high ratio of orders to scheduled courier hours. While, given the available data, we can not establish the cause of the understaffed schedules, errors in forecasting or planning may be to blame.

As expected, the results show that urgency and flexibility have pronounced adverse effects in most measures, except cost per order. The linear trends in Figure 2 suggest that an additional minute of hard reaction time could reduce click-to-door roughly as much as 100 seconds in mean, 20 seconds in overage, and 2.5 minutes in the 90th percentile. Meanwhile, an additional minute of soft pickup flexibility could reduce click-to-door roughly as much as 1.25 minutes in mean, and 30 seconds in overage. Given that urgency and flexibility are defined in terms of click-to-door target and maximum, choosing the right values for these parameters in realistic applications is of utmost importance.

The ratio of orders placed to courier hours scheduled affects all performance measures as expected: higher capacity relative to demand is tied to improvements in service performance. However, this improvement comes at a steep price: a decrease of the ratio of orders per courier hour from 2.5 to 1.25 (*e.g.*, a doubling of the number of courier hours) increases costs by more than 3 cost units per order (*i.e.*, about 30%), pulling ready-to-pickup mean close to its lower limit of 0, but buying a reduction of less than 4 minutes in click-to-door mean.

Interestingly, the size of instances does not affect ready-to-pickup mean and 90th percentile values, while the percentage of orders undelivered and click-to-door mean values suffer only a slight degradation in large instances. On the other hand, overage and 90th percentile click-to-door times do increase significantly with the number of orders placed. Moreover, the number of orders placed is positively correlated with the average size of bundles and average courier utilization. All this suggests that the algorithm fends off the challenge from larger instances by assigning larger routes, even if the click-to-door of the last orders delivered will suffer greatly, in order to achieve a better average performance.

However, it must be noted that in the case of the orders/courier hours ratio, its linear relationship with these performance metrics is much stronger, and (as shown in Figure 1) the relatively few instances with very large order sets have a very high ratio of orders to courier hours. Hence, size may not be the driver of bundling intensity and courier utilization (and service performance), but instead this is the balance between orders and courier hours.

Surprisingly, dispersion is not the strongest predictor of bundling intensity. In fact, out of the structural properties considered, dispersion has the weakest linear association to orders per bundle: an increase of 20 minutes in the dispersion measure brings the average number of orders per bundle up by 0.02 units only. Cost and service metrics exhibit a clearer linear association with dispersion, but still far from strong: an increase of 20 minutes in the dispersion measure translates, on average, to an increase of about 1 unit of cost per order, and an increase of 2.5 minutes in click-to-door mean and 1 minute in ready-to-pickup mean.

From Figure 2 we have dug up the data to contextualize the abnormally high percentage of orders undelivered (beyond 3%) registered for two instances. These instances have a low degree of dynamism (in the 0.35-0.40 range) and relatively low dispersion (in lower third of the spectrum), as well as a very small size (269 orders), but a very high ratio of orders to courier hours. Their solutions achieved relatively high ready-to-pickup values, high utilization and a relatively low number of orders per bundle.

Table 9 compares their performance vis a vis the corresponding instance variations featuring optimized schedules, and reveals that the historical courier capacity was not distributed in an effective way over time, and the system could not easily overcome larger than expected bursts of order arrivals. This example serves as a warning: no matter the size of the problems, the algorithm can only do so much in the presence of poorly distributed capacity, and, by the same measure, a good forecasting and courier scheduling process can go a long way towards guaranteeing reliable performance.

instance	$\frac{\text{orders}}{\text{cour. h.}}$	% undel.	CtoD mean	CtoD 90%	CtoD overage	RtoP mean	RtoP 90%	util. mean	$\frac{\text{cost}}{\text{order}}$	$\frac{\text{orders}}{\text{bundle}}$
1o50t100s1p100	2.1	3.35	38.24	64	6.16	8.44	25	0.73	10.28	1.17
1o50t100s1p125	2.1	4.09	41.89	64.3	7.65	8.51	21	0.74	10.2	1.16
1o50t100s2p100	2.19	0	31.54	45.2	1.44	3.38	10	0.85	10.27	1.05
1o50t100s2p125	2.19	0	35.57	53	3.38	3.73	11	0.82	10.18	1.03

Table 9: Illustrating the potential consequences of inadequate courier schedules

Figures 25-28, available in appendix G, illustrate the second-degree interaction effects of key instance features with each performance measure explored in Figure 2. Most of the observable interaction trends in Figures 25 - 28 are expected, given the direction of the main effects. One result worth highlighting is the following: in scenarios with low pickup flexibility, low reaction time, or high dispersion, having higher dynamism enables a higher 10th percentile of courier utilization and a smaller percentage of orders undelivered. We conjecture that as orders are more evenly distributed over time in these difficult scenarios, the algorithm is able to find a few more assignments during non-peak times for otherwise poorly occupied couriers.

4.5.4 Performance value of algorithm features

The average performance of the different algorithm variations is summarized by a point estimate and a 95% bootstraped confidence interval (based on 1000 bootstrap iterations) in Figure 3, for variations 0-15, and Figure 4, for variations 16-21. Most variations, except 16 and 18, achieve an average percentage of orders undelivered similar to the default algorithm (0.25%). This suggests that allowing bundles and having a sensible commitment policy are the key to ensure that as many deliveries as possible are completed.

Within variations 0-15, which focus on optimization frequency, order-assignment horizon and lookahead, the average number of orders per bundle is quite sensitive to the specific configuration of parameters. Other things equal, bundling intensity tends to be higher for larger assignment horizons. Although not shown in Figure 3, the myopic lookahead configurations that lead to higher target bundle sizes (long lookahead for orders and short lookahead for couriers), can have a small but significant marginal effect on average performance (for details, see Figure 31 in Appendix H).

Compared to the default algorithm, variations 4-7, which have a longer assignment horizon, achieve significantly worse service measures, and significantly higher utilization and bundling intensity: the assignment horizon is too long, leading to larger bundles (and higher utilization) that unnecessarily hurt the last deliveries in each route (worse click-to-door) and have a longer spread in ready times (worse ready-to-pickup).

Variations 8-11, with more frequent optimization runs, are also significantly outperformed by the default algorithm in all service metrics. In this case, however, there is not even a significant gain in utilization, which suggests that these variations are just too myopic. This conclusion is confirmed by

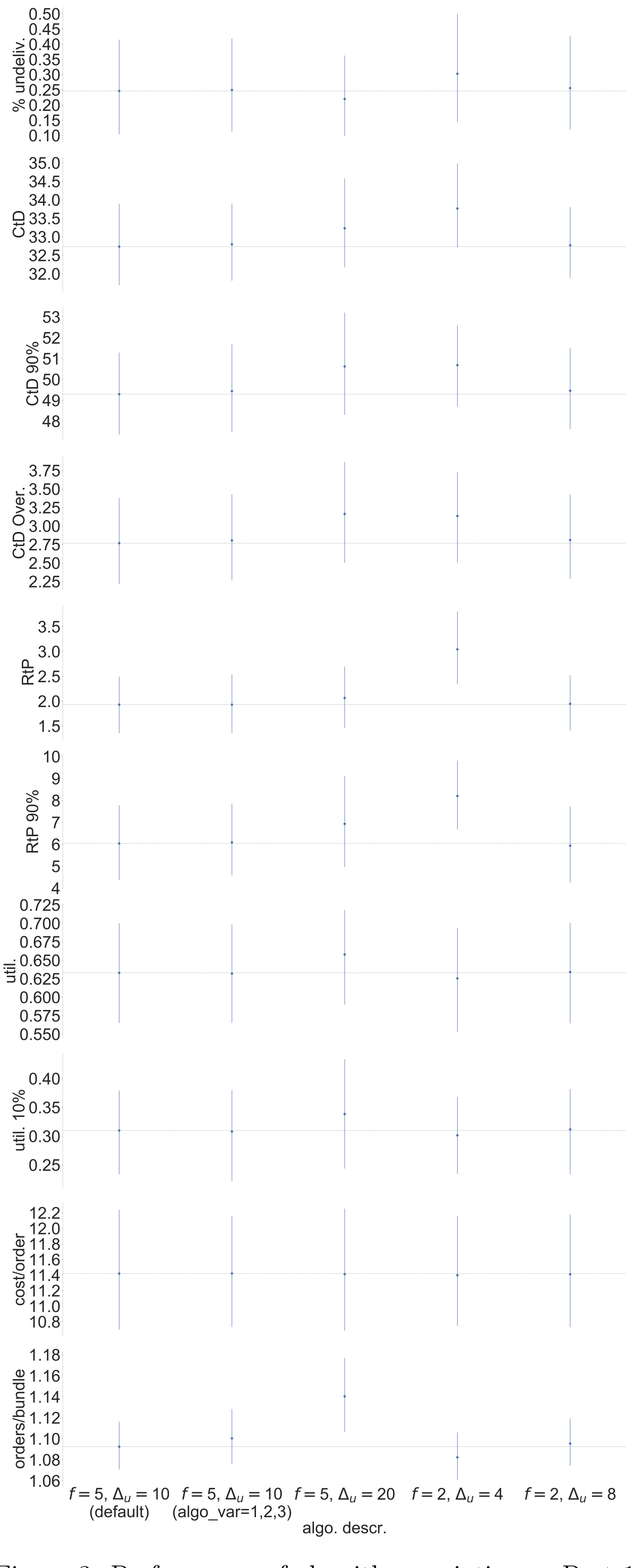

Figure 3: Performance of algorithm variations – Part 1

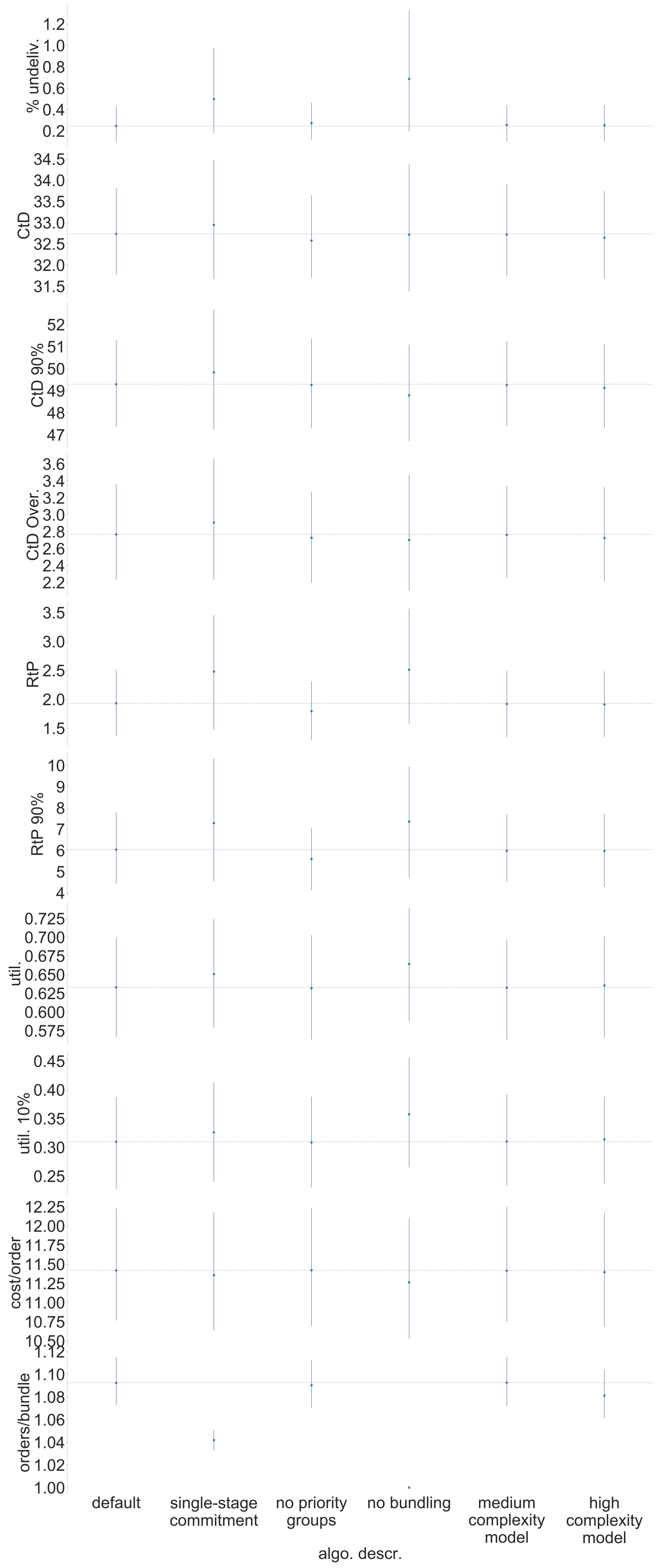

Figure 4: Performance of algorithm variations – Part 2

the fact that variations 12-15, which also have more frequent optimizations compared to the default, but are less myopic, achieve performance measures similar to the default algorithm.

As noted earlier, variations 16 (single-stage commitment strategy) and 18 (no bundling) lead to solutions with higher utilization values and less orders per bundle, due to less bundling opportunities. But this goes hand in hand with a significantly higher rate of orders undelivered, thus removing any cause for celebration in the apparent improvement in cost or the rest of service performance measures. What is worse, variation 16, with a single-stage commitment policy, also tends to increase click-to-door and ready-to-pickup measures.

Variation 17, where one matching, instead of three, is solved per optimization run, sacrifices a bit on the percentage of orders undelivered (0.27%) and cost per order (11.42), in order to outperform the default algorithm by a small margin in click-to-door mean (32.58) and overage (2.73), as well as ready-to-pickup mean (1.79).

Variation 19, featuring the assignment model of medium complexity, is able to outperform the default algorithm, however slightly, on click-to-door metrics (32.72 mean, 2.76 overage), ready-to-pickup (1.92 mean), keeping cost per order in check (11.42) and paying only a small penalty in the percentage of orders undelivered (0.26%). This is achieved with orders per bundle and utilization values similar to the default algorithm.

As expected, variation 20, featuring the assignment model of highest complexity, is able to achieve even better performance, lower click-to-door (32.64 mean, 2.72 overage) lower ready-to-pickup (1.91 mean) and lower cost per order (11.40), without increasing the percentage of orders undelivered (0.26%). This is achieved with slightly higher utilization (0.31 mean), and slightly fewer orders per bundle (1.08).

A paired differences analysis (where we focus on the differences in performance from a particular variation and the default algorithm over all instances) can reveal more clearly the value of each algorithmic feature, and allow us to assess how different instance structural properties amplify or reduce said value. Figures 29-35, available in Appendix H, summarize the distribution of paired performance differences across the instance feature space.

Impact of optimization frequency. Table 10 summarizes the results of experiment runs in algorithm variations 0-16, paired by the configuration of horizon and lookahead multipliers, to compare two settings: one where there are 2 minutes between consecutive optimizations, and one where this interval is 5 minutes (default). The results show that, by and large, 5-minute optimization intervals lead to a lower percentage of orders undelivered, lower click-to-door mean, lower ready-to-pickup mean, lower courier utilization mean, and more orders per bundle than the alternative.

	Optimization interval				paired differences	
	5 minutes		2 minutes		(f = 2) - (f = 5)	
	mean	std	mean	std	mean	std
% undelivered	0.23	0.48	0.28	0.51	0.05	0.18
CtoD mean	33.01	3.39	33.27	3.51	0.26	0.93
RtoP mean	2	1.49	2.5	1.79	0.50	0.78
cost per order	11.42	1.3	11.4	1.21	-0.01	0.18
orders per bundle	1.12	0.07	1.09	0.05	-0.03	0.03

Table 10: Differences in performance of algorithm with more frequent optimizations (2 min) vs default (5 min)

Figures 5 and 29 (in Appendix H) show the interaction of these paired differences with key structural properties. Interestingly, there is a clear trend across the urgency spectrum: as the average reaction time increases, the advantage of 5-minute optimization intervals evaporates. For instances in the upper

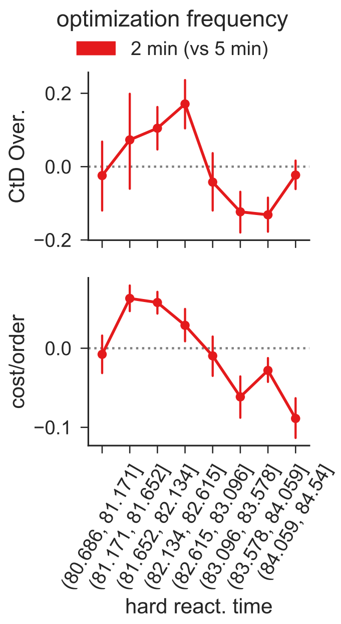

Figure 5: Performance difference of algorithm with more frequent optimizations (2 minutes, as opposed to default 5 minutes) vs instance characteristics

half of the range of reaction times, *i.e.*, for less urgent scenarios, a 2-minute optimization interval even achieves lower costs per order and lower click-to-door overage. In our view, as urgency decreases, service time windows for orders become longer, and the space of feasible bundles at any point in time grows larger, which may erode the effectiveness of the bundle insertion heuristics. Shrinking the optimization period (and the assignment horizon in the same proportion) reduces the space of feasible bundles, thus preserving the effectiveness of the heuristics in less urgent environments.

Furthermore, as the ratio of orders to courier hours increases, the default algorithm tends to have more orders per bundle than the 2-minute interval alternative, reflecting the greater ability to consolidate orders of an algorithm with less frequent assignments, which allows the default algorithm to maintain an advantage, albeit diminishing, in terms of click-to-door and ready-to-pickup. However, the trend is steady enough that, in the upper third of the range of this ratio, 2-minute optimization intervals lead to lower percentage of orders undelivered *and* lower cost per order.

Impact of horizon for assignments. Table 11, as well as Figures 6 and 30 (in Appendix H), summarize the performance difference between variation 4, which considers orders for assignment if they are ready within 20 minutes, and the default algorithm, where the assignment horizon is 10 minutes.

	Assignment horizon				paired differences	
	10 minutes		20 minutes		(u = 20) - (u = 10)	
	mean	std	mean	std	mean	std
% undelivered	0.25	0.46	0.22	0.51	-0.02	0.17
CtoD mean	32.74	3.34	33.15	3.37	0.42	0.55
RtoP mean	1.93	1.52	2.05	1.45	0.12	0.41
cost per order	11.42	1.26	11.41	1.34	-0.01	0.19
orders per bundle	1.09	0.05	1.13	0.06	0.04	0.02

Table 11: Differences in performance of algorithm with longer assignment horizon (20 min) vs default (10 min)

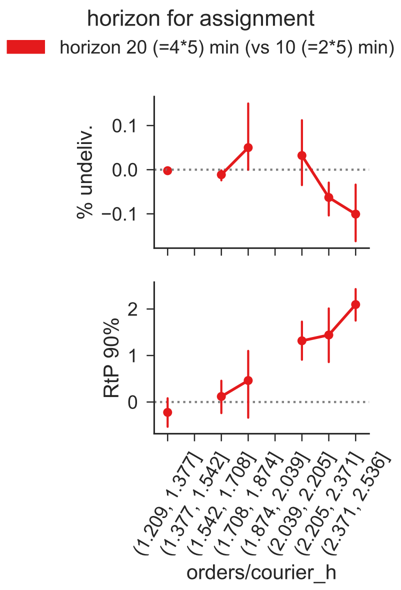

Figure 6: Performance difference of algorithm with a 20 minute horizon for assignment of orders (as opposed to default 10 minute horizon) vs instance characteristics

A longer assignment horizon leads to more orders per bundle and higher courier utilization levels. Meanwhile the default assignment horizon is preferable in terms of click-to-door and ready-to-pickup, though this advantage comes at the price of a slightly higher percentage of orders undelivered. The difference in service performance tends to become smaller for instances with low urgency, less orders placed, and low order to courier hours ratio, but its sign persists across almost all levels of all instance characteristics, with two clear exceptions: (i) the instances with low dynamism and very poor quality schedules summarized in Table 9, where the default algorithm gives up some quality of service in order to get a higher percentage of orders delivered; and (ii) a group of instances with high dispersion and high urgency, summarized in Table 12, where the default algorithm struggles to create enough bundles, sacrificing ready-to-pickup (and some order deliveries altogether) to preserve the rest of performance metrics, while variation 4 achieves the same feat with better ready-to-pickup (and more orders delivered) by virtue of its reduced myopism.

instance	resp. time	orders	disp.	algo.	% undel.	CtoD mean	RtoP mean	util. mean	$\frac{\text{cost}}{\text{order}}$	$\frac{\text{orders}}{\text{bundle}}$
5o50t100s1p100	81.6	1362	25.8	0	0.5	38.3	5.8	0.8	10.4	1.2
5o50t100s1p125	81.1	1362	25.8	0	1.3	42.7	6.4	0.9	10.3	1.2
7o50t100s1p125	81.2	1606	27	0	0.6	38	4.6	0.8	10.4	1.1
5o100t100s1p100	81.6	2724	26.7	0	0.6	37.1	4.2	0.8	10.8	1.2
5o50t100s1p100	81.6	1362	25.8	4	0.3	38	5.3	0.9	10.2	1.2
5o50t100s1p125	81.1	1362	25.8	4	1.0	42.2	5.7	0.9	10	1.2
7o50t100s1p125	81.2	1606	27	4	0.3	36.7	3.1	0.9	10.2	1.1
5o100t100s1p100	81.6	2724	26.7	4	0.3	37.1	3.7	0.8	10.5	1.3

Table 12: Illustrating the potential consequences of excessive myopism in assignment horizon

In terms of cost per order, the default assignment horizon tends to yield better results in instances with low urgency, while the opposite is true for instances with high urgency. As a tentative explanation, consider that for a given assignment horizon, higher urgency implies that each order has the chance to be matched to a courier less times before a final commitment is required: a longer assignment horizon compensates by beginning the assignment process for each order earlier in time.

The default assignment also achieves lower cost than variation 4 for low order to courier hour ratio, but the tables are turned for higher order to courier hour ratio: when there are many orders per courier and bundling becomes critical, a longer assignment horizon allows higher consolidation levels, thus driving cost down.

Impact of commitment policy. Table 13, Figure 7 and Figure 32 (in Appendix H) summarize the performance difference between algorithm variation 16, which uses a single-stage lazy commitment, and the default algorithm, which uses two-stage additive commitment.

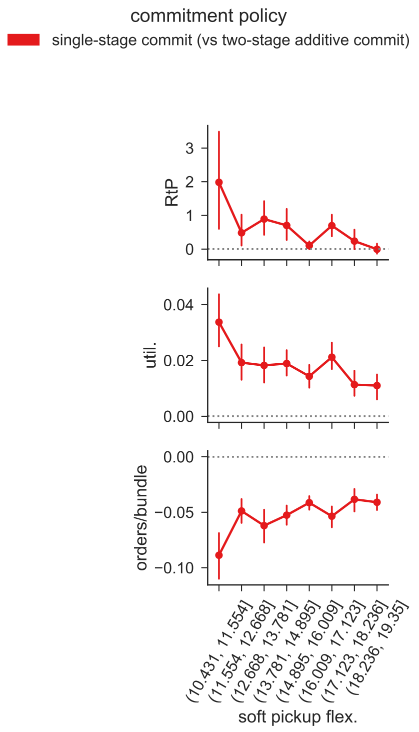

Figure 7: Performance difference of algorithm with single-stage commitment rule (as opposed to default two-stage additive rule) vs instance characteristics.

	Commitment strategy				paired differences	
	Two-stage		Single-stage		(single - two-stage)	
	mean	std	mean	std	mean	std
% undelivered	0.25	0.46	0.5	1.29	0.25	1.01
CtoD mean	32.74	3.34	32.95	4.06	0.21	1.11
RtoP mean	1.93	1.52	2.48	2.69	0.55	1.26
cost per order	11.42	1.26	11.36	1.28	-0.06	0.13
orders per bundle	1.09	0.05	1.04	0.02	-0.05	0.03

Table 13: Differences in performance of algorithm with single-stage commitment policy vs. default (two-stage additive commitment)

The evidence shows that, compared to the default, a single-stage commitment policy encourages less orders per bundle across the board. Couriers are on the road a longer proportion of time (increased utilization levels), as the algorithm struggles to find solutions that deliver all orders (percentage of orders undelivered doubles) with the same service quality. Meanwhile, mean ready-to-pickup times degrade by 28%, but click-to-door times increase at a much slower pace (about 1% in mean and about 5% in overage). We advance a tentative interpretation: if conditions of the system have changed, the two-stage commitment policy has the advantage of allowing bundles to grow in size, or to change their delivery sequence while couriers are en-route or even waiting at the restaurant (thus protecting click-to-door measures, but not ready to pickup measures).

All in all, the results reaffirm the conclusions reached in our reading of Figure 4: if the goal is to deliver all orders and deliver them quickly, using of a single-stage commitment policy is not a good idea in any scenario. In fact, the flaws of single-stage commitment are more evident (through a larger increase in the percentage of orders undelivered and all click-to-door and ready-to-pickup measures) on instances with low dynamism, low pickup flexibility, or high ratio of orders to courier hours, that is, as the dynamics of the problem become more challenging.

Impact of priority scheme. Table 14, Figure 8 and Figure 33 (in Appendix H) summarize the performance difference between algorithm variation 17, which uses a single matching per optimization run, and the default algorithm, which solves a sequence of 3 matchings based on a priority rule for orders and bundles.

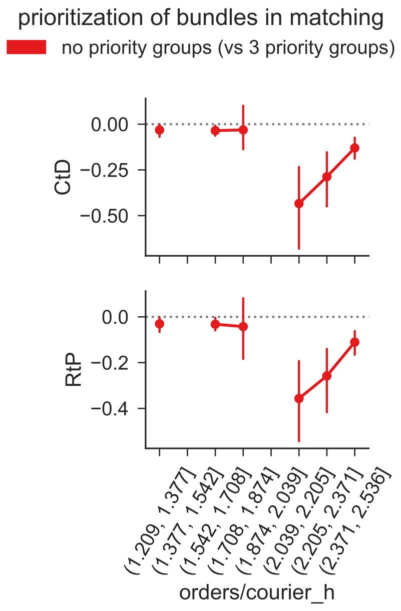

Figure 8: Performance difference of algorithm with a single matching problem per optimization run (as opposed to default sequence of 3 matchings based on priority) vs instance characteristics

For the most part, running the algorithm with or without priority groups does not make much of a difference in terms of orders undelivered, click-to-door measures, or even cost, except for a subset of poorly staffed instances, (as described in Table 9), where doing away with the priority scheme leads to a significant sacrifice in the percentage of orders undelivered, accompanied by improvements in click-to-door, ready-to-pickup and cost per order: in these instances, during moments when the system is under pressure to keep up with all deliveries, the default algorithm prioritizes delivery of orders that are already in trouble (thereby reducing the number of undelivered orders) and places the burden in orders with lower priority (worsening their click-to-door performance), while variation 17 prioritizes orders that are still on time, to achieve gains in click-to-door times, at the expense of orders that are already late (which sometimes may imply giving up on them altogether). This behavior suggests that on instances

	Priority scheme				paired differences	
	Yes		No		(No - Yes)	
	mean	std	mean	std	mean	std
% undelivered	0.25	0.46	0.27	0.61	0.03	0.26
CtoD mean	32.74	3.34	32.58	3.16	-0.16	0.42
RtoP mean	1.93	1.52	1.79	1.28	-0.14	0.35
cost per order	11.42	1.26	11.42	1.26	0.00	0.08
orders per bundle	1.09	0.05	1.09	0.05	0.00	0.01

Table 14: Differences in performance of algorithm with no priority scheme to find assignments vs. default, (three priority groups)

with a highly limited supply of couriers, demand management strategies to select which orders to "give up" on can have a big impact.

Impact of bundling. Table 15, Figure 9, and Figure 34 (in Appendix H) summarize the performance difference of variation 18, where the algorithm is forbidden from bundling orders together (*i.e.* a hard limit of 1 is imposed on the size of bundles), and the default algorithm.

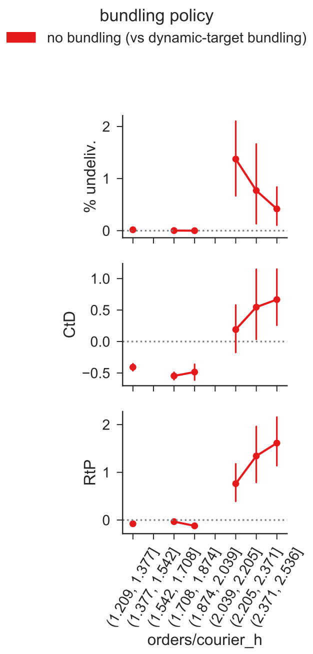

Figure 9: Performance difference of algorithm that completely forbids bundling (as opposed to default with dynamic bundling intensity) vs instance characteristics

The results show that allowing bundles is critical for the system: their prohibition almost triples the number of orders undelivered, while deteriorating the ready-to-pickup mean by about 30%, all for minuscule gains of less than 1% in click-to-door and a 1% improvement in cost per order.

Results show that only in the most dynamic, most flexible, less congested and most disperse scenarios can the absence of bundles lead to small improvements in click-to-door without sacrifices in the number of orders delivered. Coincidentally, in these extreme scenarios, the number of orders per bundle used by the default algorithm is smaller, *i.e.*, bundling opportunities are scarce or of little help anyway.

	Bundling				paired differences	
	Yes		No		(No - Yes)	
	mean	std	mean	std	mean	std
% undelivered	0.25	0.46	0.69	1.77	0.44	1.52
CtoD mean	32.74	3.34	32.72	3.93	-0.02	1.16
RtoP mean	1.93	1.52	2.51	2.53	0.58	1.35
cost per order	11.42	1.26	11.26	1.29	-0.15	0.18
orders per bundle	1.09	0.05	1	0	-0.09	0.05

Table 15: Differences in performance of algorithm with no bundling allowed vs. default (allow bundling)

Impact of complexity of the assignment model. Table 16, Figure 10, and Figure 35 (in Appendix H) summarize the performance difference of experiment runs that depart from the simple matching model, and use larger and more complex assignment procedures.

	Assignment model complexity						paired differences			
	Low		Medium		High		(Medium - Low)		(High - Low)	
	mean	std	mean	std	mean	std	mean	std	mean	std
% undelivered	0.25	0.46	0.26	0.57	0.25	0.54	0.01	0.22	0.01	0.20
CtoD mean	32.74	3.34	32.72	3.32	32.64	3.34	-0.02	0.19	-0.09	0.22
RtoP mean	1.93	1.52	1.92	1.52	1.91	1.52	-0.01	0.17	-0.02	0.17
cost per order	11.42	1.26	11.42	1.25	11.4	1.25	0.00	0.07	-0.02	0.08
orders per bundle	1.09	0.05	1.09	0.05	1.08	0.05	0.00	0.01	-0.01	0.01

Table 16: Differences in performance of algorithm with with medium and high complexity assignment models (as opposed to default linear programming model)

Overall, results show that the higher complexity assignment model dominates the medium complexity model in all performance metrics. And, compared to the default, more complex models trade off a slight increase in orders undelivered for slight improvements in the rest of service performance metrics. We note that the small magnitude of these differences is rather surprising: this suggests that the default approach, while perfectible, goes a really long way in guaranteeing satisfactory performance measures.

At greater detail, it can be noted that both medium and high complexity variations perform similarly in terms of orders delivered, matching the levels of the default algorithm, except in poorly staffed instances, where Procedure 3 seems to be counterproductive (note that both medium and high complexity versions use this Procedure), as significantly more orders are left undelivered compared to the default setting. Inspection of the simulation logs reveals that this is caused by a poor decision logic at the end of the operating period: a courier that is about to finish her shift is assigned a follow-up pair, but only commits to the first route; then her shift finishes and the second element of the pair has no courier in a good position to pick up the assignment. This unintended behavior – which may be handled as an exception at the commitment stage in a production-level application – gives the reader a flavor of the challenges faced during the implementation of fully-automated algorithmic dispatching technologies.

Moreover, the dominance of the high complexity assignment model over the medium and low complexity models is almost uniform across the space of instance characteristics, except in the case of the very poorly staffed instances. Interestingly, the high complexity model solutions are consistently less bundling-intensive than the ones produced by the default algorithm, while achieving noticeable, though small, improvements in service and cost metrics in every instance, except the same subset of poorly staffed instances.

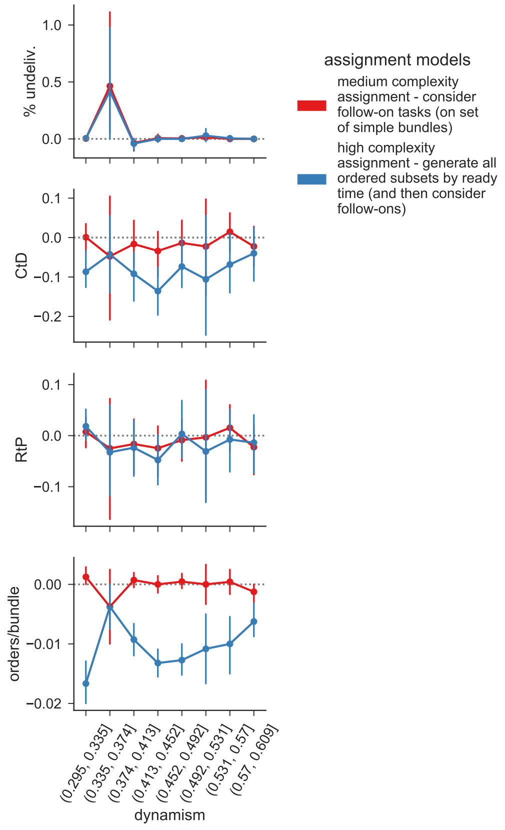

Figure 10: Performance difference of algorithm with increasingly complex assignment models (as opposed a matching linear program) vs instance characteristics

4.5.5 Performance of algorithm with imperfect information

Up to this point, we have assumed perfect information. Reality is different. Therefore, to assess the value of information, we perform one experiment (many more are possible, of course) with a slightly different information scheme: we assume that all information is perfectly known in advance, *except for the meal preparation time at a restaurant* for which only the (true) mean is known. Consequently, until an order is actually known to be ready (*i.e.*, the simulation reaches the realized ready time of the order), the algorithm uses the restaurant mean meal preparation time to *estimate* the order ready time. The standard deviation of the difference between estimated and actual meal preparation times is about 8.19 minutes (average across all 240 instances), while the mean difference is about 0.25 minutes.

Table 17 summarizes the performance effects of this form of uncertainty. Unsurprisingly, service quality degrades compared to the perfect-information baseline: in relative terms, orders undelivered more than double, click-to-door mean increases by almost 4%, and ready-to-pickup mean increases by about 50%. These results suggest that, under these conditions, the algorithm is unable to provide service to some orders (presumably due to couriers signing-off before orders are actually ready), but preserves a rather high quality of service for the remaining ones: after all, given that the imperfect estimates have a standard deviation of more than 8 minutes, 1.29 minutes in click-to-door and 0.97 minutes in ready-to-pickup may appear to be a worthwhile trade off.

5 Final remarks

In this paper, we have introduced the Meal Delivery Routing Problem, a dynamic deterministic model of the structure and functioning of meal delivery systems, and we have developed a rolling-horizon repeated-matching algorithm to solve this problem in (near) real-time. We have shown through compu-

	Information assumption				paired differences	
	Perfect		Estimate		(Estimate - Perfect)	
	mean	std	mean	std	mean	std
% undelivered	0.25	0.46	0.62	2.03	0.37	1.97
CtoD mean	32.74	3.34	34.03	4.85	1.29	3.04
CtoD 90%	49.3	5.6	50.63	7.29	1.34	3.54
CtoD overage	2.77	1.53	3.40	2.27	0.63	1.12
RtoP mean	1.93	1.52	2.90	3.53	0.97	2.88
RtoP 90%	6.04	4.24	9.94	6.14	3.90	3.46
utilization mean	0.63	0.12	0.63	0.12	0.00	0.02
utilization 10%	0.31	0.15	0.33	0.15	0.02	0.06
cost per order	11.42	1.26	11.32	1.21	-0.10	0.14
orders per bundle	1.09	0.05	1.11	0.07	0.02	0.03

Table 17: Impact of imperfect preparation time estimates (only departure from perfect information).

tational results that our methodology can be leveraged to build solutions of high quality with respect to multiple performance objectives in meal delivery systems of realistic size and difficulty levels. When confronted with scenarios of high urgency, low flexibility, and scarce capacity, the rolling-horizon assignment algorithm makes minimal sacrifices in the fulfillment of deliveries, while sacrificing click-to-door and ready-to-pickup measures in a steady fashion (linear performance degradation in all dimensions of the instance characteristic space). The issue of courier scheduling deserves a special emphasis: in practical applications, ensuring that an appropriate quantity of couriers is available at all times is often equally or more crucial than tuning specific algorithm parameters.

Furthermore, we have shown that the key algorithmic ideas embedded in our method all play an important role: optimization frequency, assignment horizon and commitment policy work together to maintain a reasonable pool of individual orders from which good bundles (and routes) can be made, if needed, without compromising service quality. Meanwhile, the intensity of bundle generation can be modulated effectively using the dynamic target bundle sizes, thus allowing the system to adapt automatically to temporal fluctuations in demand and supply. Moreover, the proposed priority scheme is critical to keep up with the pace of arriving orders in the most overwhelming scenarios.

As discussed in Section 1, meal delivery and ride-hailing systems share similarities in scale, urgency and dynamism; and our results confirm that a matching approach, which has already shown its merits in ride-hailing, is also effective in meal delivery. Our results show that a linear matching model can be expressive and powerful enough to achieve high quality solutions fast. Moreover, careful design of slightly more complex assignment models can correct some of the pitfalls stemming from an excessively myopic solution approach, enabling small but significant marginal improvements in overall performance, while keeping run-times in check.

The ideas and results reported in this paper give rise to a variety of questions for further research. In the realm of deterministic models, the development of exact algorithms for the perfect-information version of the MDRP are clearly a worthwhile endeavor. In a more practical note, restaurant-dependent dynamic target bundle sizes are a natural extension of our approach that would allow the system to compensate for *geographic* imbalances in supply and demand. Additionaly, a systematic study of the value of relocation decisions that are completely decoupled from the order commitment logic is an important question.

In the stochastic realm, opportunities are abundant. While we recognize the importance of predicting order ready times (as a function of arrival patterns and preparation times), we believe that the questions that deserve greater attention, because of their novelty, are those related to courier autonomy. For instance, independent-contractors usually work for more than one company (sometimes simultaneously), and they leverage their autonomy to reject orders in a strategic way. Developing models to understand assignment rejection behavior, and its relation to system performance, is a fundamental step in the road towards a stochastic dynamic solution algorithm.

Another interesting phenomenon related to courier autonomy is "courier drainage": couriers make waiting and relocation decisions based on incomplete local information about the system in which they operate. Over time, individual autonomous decisions may lead to too many couriers on areas that are perceived as "profitable" and too few of them in areas that are not perceived as such. The system-wide consequences emerging from courier drainage may not be in the best interest of the central provider, restaurants, and even couriers themselves. Understanding the dynamics of courier drainage, and developing strategies to mitigate it, is an exciting (and daunting) research direction.

References

[1] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295 – 303, 2012. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2012.05.028. URL http://www.sciencedirect.com/science/article/pii/S0377221712003864.

[2] C. Archetti, D. Feillet, and M.G. Speranza. Complexity of routing problems with release dates. *European Journal of Operational Research*, 247(3):797–803, 2015.

[3] Claudia Archetti, Martin Savelsbergh, and M. Grazia Speranza. The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, 254(2):472 – 480, 2016. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2016.03.049. URL http://www.sciencedirect.com/science/article/pii/S0377221716301953.

[4] N. Azi, M. Gendreau, and J.-Y. Potvin. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199(1):103–112, 2012.

[5] Evan Bakker. *The on-demand meal delivery report: Sizing the market, outlining the business models, and determining the future market leaders.* http://read.bi/2bU7EuD, 2016. Accessed: 2016-12-01.

[6] G. Berbeglia, J.F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

[7] Iman Dayarian and Martin Savelsbergh. Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Optimization Online*, 2017.

[8] Caitlin Dewey. The insane $43 billion system that gets food delivered to your door. https://www.washingtonpost.com/news/wonk/wp/2017/08/08/the-insane-43-billion-system-that-gets-food-delivered-to-your-door/, August 2017. Accessed: 2017-10-03.

[9] Masabumi Furuhata, Maged Dessouky, Fernando Ordez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57(Supplement C):28 – 46, 2013. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2013.08.012. URL http://www.sciencedirect.com/science/article/pii/S0191261513001483.

[10] Carsten Hirschberg, Alexander Rajko, Thomas Schumacher, and Martin Wrulich. The changing market for food delivery. https://www.mckinsey.com/industries/high-tech/our-insights/the-changing-market-for-food-delivery, November 2016. Accessed: 2017-05-01.

[11] M. A. Klapp, A. L. Erera, and A. Toriello. The one-dimensional dynamic dispatch waves problem. *Transportation Science*, 2016. doi: 10.1287/trsc.2016.0682.

[12] Mathias A. Klapp. *Dynamic optimization for same-day delivery operations*. PhD thesis, Georgia Institute of Technology, 2016.

[13] A. Larsen, O. Madsen, and M. Solomon. Partially dynamic vehicle routing—models and algorithms. *Journal of the Operational Research Society*, 53(6):637–646, Jun 2002. ISSN 1476-9360. doi: 10.1057/palgrave.jors.2601352. URL `https://doi.org/10.1057/palgrave.jors.2601352`.

[14] Karsten Lund, Oli B.G. Madsen, and Jens M. Rygaard. Vehicle routing problems with varying degrees of dynamism. Technical Report IMM-REP-1996-1, Department of Mathematical Modeling, The Technical University of Denmark, May 1996.

[15] Snezana Mitrovic-Minic and Gilbert Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38(7):635–655, 2004. ISSN 0191-2615.

[16] Snezana Mitrovic-Minic, Ramesh Krishnamurti, and Gilbert Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669 – 685, 2004. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2003.09.001. URL `http://www.sciencedirect.com/science/article/pii/S019126150300095X`.

[17] Morgan Stanley Research. Is online food delivery about to get 'amazoned'? `https://www.morganstanley.com/ideas/online-food-delivery-market-expands/`, sep 2017. Accessed: 2017-10-04.

[18] V. Pillac, M. Gendreau, C. Guéret, and A.L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[19] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007.

[20] H.N. Psaraftis, M. Wen, and C.A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

[21] Reuters. *How Amazon is making package delivery even cheaper*. `http://fortune.com/2016/02/18/amazon-flex-deliveries/`, 2016.

[22] Damián Reyes, Alan L. Erera, and Martin W.P. Savelsbergh. Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research*, 2017. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.09.020. URL `http://www.sciencedirect.com/science/article/pii/S037722171730841X`.

[23] Afonso Sampaio, Martin Savelsbergh, Lucas Veelenturf, and Tom Van Woensel. Crowd-based city logistics. *Optimization Online 2017-6346*, 2017.

[24] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review*, 91(Supplement C):190 – 207, 2016. ISSN 1366-5545. doi: https://doi.org/10.1016/j.tre.2016.04.010. URL `http://www.sciencedirect.com/science/article/pii/S1366554515303033`.

[25] MW Ulmer, BW Thomas, and DC Mattfeld. Preemptive depot returns for a dynamic same-day delivery problem. Working paper. TU Braunschweig, Germany, 2016.

[26] Rinde RS van Lon, Eliseo Ferrante, Ali E Turgut, Tom Wenseleers, Greet Vanden Berghe, and Tom Holvoet. Measures of dynamism and urgency in logistics. *European Journal of Operational Research*, 253(3):614–624, 2016.

[27] Stacy A. Voccia, Ann Melissa Campbell, and Barrett W. Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 0(0):null, 0. doi: 10.1287/trsc.2016.0732.

[28] Xing Wang, Niels Agatz, and Alan Erera. Stable matching for dynamic ride-sharing systems. *Transportation Science*, 0(0):null, 0. doi: 10.1287/trsc.2017.0768.

[29] Jian Yang, Patrick Jaillet, and Hani Mahmassani. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38(2):135–148, 2004. doi: 10.1287/trsc.1030.0068. URL `http://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0068`.

[30] Baris Yildiz and Martin Savelsbergh. Provably high-quality solutions for the meal delivery routing problem. Georgia Institute of Technology, 2017.

Appendix A: Instance generation - reduction of order and courier sets

Reduction of order set through randomized selection of restaurants

```
Algorithm 4: Indirect p-Reduction of order set through randomized selection of restaurants
Data: $p$, the percentage of reduction.
      $O$, original set of orders.
      $R$, original set of restaurants.
Result: $O'$, reduced set of orders.
        $R'$, reduced set of restaurants.

$O' \leftarrow \emptyset$;
$R' \leftarrow \emptyset$;
$n' \leftarrow p * |O|$;
repeat
    $r \leftarrow$ restaurant selected from $R$ (uniform random sampling without replacement);
    $O_r \leftarrow$ orders placed from $O$ at $r$;
    $O' \leftarrow O' \cup O_r$ ;
    $R' \leftarrow R' \cup \{r\}$;
until $|O'| \geqslant n'$;
if $|O'| > n'$ then
    $x \leftarrow |O'| - n'$ ;                                  /* excess from target */
    $y \leftarrow n' - |O' \setminus O_r|$ ;   /* deficit from target if last restaurant not in sample */
    if $y \leqslant x$ then
        $O' \leftarrow O' \setminus O_r$ ;                      /* backpedal... */
        $R' \leftarrow R' \setminus \{r\}$;
return $O'$,$R'$ ;
```

Reduction of courier set

Algorithm 5: p-Reduction of courier set

```
Data: $p$, the percentage of reduction.
      $S$, the set of shift start times in current schedule.
      $B_s \forall s \in S$, the set of shifts in the original schedule that share start time $s \in S$.
Result: $C_s \forall s \in S$, the set of shifts in the reduced schedule that share start time $s \in S$.

for $s \in S$ do
    /* Set target number of hours                                              */
    $c'_s \leftarrow p * c(B_s)$;                    // $c(B_s)$ is total number of courier hours in $B_s$
    /* Build $C_s$ by sampling one shift at a time                             */
    $C_s \leftarrow \emptyset$;
    repeat
        $b \leftarrow$ shift selected from $B_s$ (uniform random sampling without replacement);
        $C_s \leftarrow C_s \cup \{b\}$ ;
    until $c(C_s) \geqslant c'_s$;
    if $c(C_s) > c'_s$ then
        $b \leftarrow$ shift of maximal duration in $C_s$;
        $b' \leftarrow$ shift starting at time and location of $b$, with duration $c(b) - (c(C_s) - c'_s)$;
        $C_s \leftarrow \{b'\} \cup C_s \setminus \{b\}$;
    end
end
return $C = \bigcup_{s \in S} C_s$ ;
```

Appendix B: Instance generation - optimizing courier schedules

The shift cover problem.

Before we introduce our offline shift-optimization process, we begin by describing the optimization problem that lies at its core.

Let the planning horizon be $[0, T]$. An activity profile is a right-continuous function $a : [0, T] \to \mathbb{Z}_{\geq 0}$ indicating at each point in time t an activity level $a(t)$ that needs to be covered by resources (e.g., at each point in time it indicates the number of active drivers required). Furthermore, let there be a finite number of points $0 < t_1 < t_2 < \ldots < t_n < T$ at which the activity level changes, either up or down. We assume that $a(t) = 0$ for $0 \leq t < t_1$ and for $t_n \leq t \leq T$.

Resources can be allocated to perform activities in the form of shifts. A resource profile is a function $r : [0, T] \to \mathbb{Z}_{\geq 0}$ indicating at each point in time the number of active resources, where the number of active resources is determined by the number of active resource shifts. A resource profile is feasible if $a(t) \leq r(t)$ for $t \in [0, T]$. Each resource shift has to satisfy the following requirements:

1. The length of any shift must be a number from $L = \{l_1, \ldots, l_p\}$, where $l_1 \leqslant \ldots \leqslant l_p$.
2. A shift can start only at pre-specified shift start times $0 < s_1 < \ldots < s_m < T$, with $s_1 \leq t_1$ and $s_m + l_1 \geq t_n$.

The shift cover problem is to minimize the total shift length required to cover the activity profile, i.e., to minimize the area under a feasible resource profile, given the start times and the shift lengths allowed.

If we denote the number of shifts of length l_k that start at time s_j by $x_k^{s_j}$, then the following integer program solves the shift cover problem.

$$\min \sum_{j=1}^{m} \sum_{k=1}^{p} l_k x_k^{s_j}$$

$$s.t. \sum_{k=1}^{p} \sum_{t_i - l_k \leqslant s_j < t_i} x_k^{s_j} \geqslant a(t_{i-1}) \qquad \forall i = 1, \ldots, n$$

A concrete implementation

To decide the courier schedule to implement on a given instance, we have computed an “activity profile” based on the solution to a perfect-information pick-up and delivery problem with time windows, obtained via the Adaptive Large Neighborhood Search (ALNS) heuristic [19]. The activity profile counts the number of non-idle drivers in the perfect-information solution at regular intervals throughout the operating period(*e.g.*, every optimization time). Then, we can follow Algorithm 6.

Algorithm 6: Shift optimization procedure based on perfect-information heuristic solution

```
Data: $\mathbb{P}$, activity profile computed from ALNS solution to perfect-information PDPTW.
      $B$, the maximum total courier hours used historically.
Result: Optimized shift schedule for the dynamic problem
$a \leftarrow 1$ ;                    // value by which to re-scale the ALNS activity profile
repeat
    $\mathbb{P}' \leftarrow$ scale up $\mathbb{P}$ by $a$ ;       // multiply value counts of every period in $\mathbb{P}$ by $a$
    Solve shift cover problem on $\mathbb{P}'$;
    if shift-length of $\mathbb{P}'$ exceeds $B$ then STOP repetition loop;
    $a \leftarrow a + 0.1$;
    $\mathbb{P} \leftarrow \mathbb{P}'$;
until $B <$ total courier hours in schedule induced by $\mathbb{P}$;
Locate courier start locations randomly around the busiest restaurants in the next hour from
  the starting time of each shift;
return shift schedule induced by (re-scaled) $\mathbb{P}$, with total courier hours no larger than $B$;
```

Appendix C: Description of instance files

Information for an instance is provided in four text files, with lines formatted as follows (in all files the first line contains helpful information on its content and can be ignored):

Listing 1: Restaurant set

```
Restaurant ID, x, y
```

Listing 2: Order set

```
Order ID, x, y, placement time, Restaurant ID, ready time
```

Listing 3: Courier set

```
Courier ID, x, y, on-time, off-time
```

Listing 4: Time and compensation parameters

```
Travel time multiplier, service time at pickup,
service time at drop-off, target click-to-door time,
maximum click-to-door time, per-order pay, per-hour pay
```

Appendix D: Use of solution evaluator

A solution to an instance can be summarized by a list of assignments of orders to couriers. A full specification also includes pickup and drop-off times for individual orders, as well as the sequence of locations visited and the corresponding departure times for each courier.

Our solution evaluator receives as input three files. The first file contains one line for each pickup and delivery assignment (which represent decisions and corresponding decision times). The format of each line is as follows (in all files the first line is assumed top contain helpful information on its content and will be ignored):

Listing 5: Assignment information

```
assignment time, pickup time, Courier ID, Order ID, ..., Order ID
```

Note that if prepositioning or assignmend updates are allowed in the operating environment, only final assignments should be recorded in this file.

The second file contains one line per individual order delivered, formatted as follows:

Listing 6: Order delivery information

```
Order ID, placement time, ready time,
pickup time, delivery time, courier ID
```

The third file contains one line per movement of a courier. Moves corresponding to each courier are presented in blocs and, within each bloc, lines are written following the sequence in which the movements are executed. In general, each line is formatted as follows:

Listing 7: Courier dispatch information

```
Courier ID, departure time, origin, destination
```

where ORIGIN is either 0, indicating the courier's on-location, RESTAURANT ID, or ORDER ID, and DESTINATION is either RESTAURANT ID or ORDER ID.

The evaluator checks that the following conditions are satisfied:

1. each order is assigned at most once;
2. assignments are not made before orders are placed;
3. a courier completes all the pickup tasks before the end of his duty period.
4. the pickup time of a bundle is at or after the latest ready time of the orders in the bundle;
5. orders are delivered in the sequence prescribed by the assignment;
6. the sequence of courier movements is feasible (no tele-transporting), and all arrival and departure times are consistent.
7. at the time of the pickup of an order or a bundle, the courier is in the corresponding restaurant.
8. at the time of the drop off of an order, the courier is in the corresponding diner location.

If a given solution satisfies these conditions, *i.e.* it is feasible, the evaluator returns a file with values for the aforementioned performance metrics. If the solution is infeasible, the evaluator returns a file indicating the ID of couriers and orders involved in an infeasible action.

Appendix E: Parameter Tuning

Before starting the full-scale experiments, reasonable default values for a series of secondary parameters in the algorithm must be found. These include:

1. tolerances for service loss (click-to-door overage) and freshness loss (delay in pickup beyond ready time) before orders escalate in priority;
2. tolerance for freshness loss of individual orders before forcing the commitment of a bundle;
3. coefficient (reward) for utilization in the matching objective;
4. freshness loss coefficient (penalty) in the matching objective;
5. service loss coefficient (penalty) in the insertion cost function.

To tune these parameters, a preliminary experiment is conducted over a stratified sample of the set of instances: for each of the 24 instance variations, 4 instances are selected at random. Then, a set of 32 random parameters configurations is created by repeating the following procedure:

1. The tolerances for service loss and freshness loss before orders escalate in priority are set simultaneously by uniform-randomly choosing a pair from

   $$[(10,5),(15,5),(15,10),(20,5),(20,10),(20,15),(25,5),(25,10),(25,15),(25,20)]$$

   The first entry in the pair represents the service loss tolerance, and the second one represents the freshness loss tolerance.
2. The tolerance for freshness loss of individual orders before forcing the commitment of a bundle is set by uniform-randomly choosing a value in $[15, 20, 25]$.
3. The utilization coefficient in the matching objective is set by sampling the uniform distribution on the interval $[1, 10]$.
4. The freshness loss coefficient in the matching objective is set by sampling the uniform distribution on the interval $[0, 1]$.
5. The service loss coefficient in the insertion cost function is set by sampling the uniform distribution on the interval $[0, 1]$.

The resulting algorithm configurations are then tested on each instance. Default values are chosen for each of the 24 instance variations, with two goals in mind: virtually zero orders left-over at the end of the planning period, and minimal average click-to-door. Concretely, an algorithm run is considered 'successful' in a particular instance if no more than 0.5% of orders are left unserved, and a parameter configuration is considered 'successful' in a given instance variation sub-sample if the algorithm succeeds in at least 75% of the cases (*i.e.* in 3 out of 4 instances in the tuning sample). The selection rule is then: among the 'successful' parameter configurations choose one that minimizes average click-to-door.

Appendix F: Algorithm performance on instance variations

The figures below illustrate performance metrics for different instance subsets, as follows:

- courier schedules: historical [`cour_sched_type`=1] and optimized [`cour_sched_type`=2].
- travel times: original [`tt_multip=100`] and shorter by 25% [`tt_multip=75`].
- preparation times: original [`prep_multip=100`] and longer by 25% [`prep_multip=125`].
- size reductions: original [`size_reduction=o100`]; sampling 50% of couriers and 50% of order set [`size_reduction=o50`]; and sampling 50% of couriers and enough restaurants to obtain 50% of orders [`size_reduction=r50`].

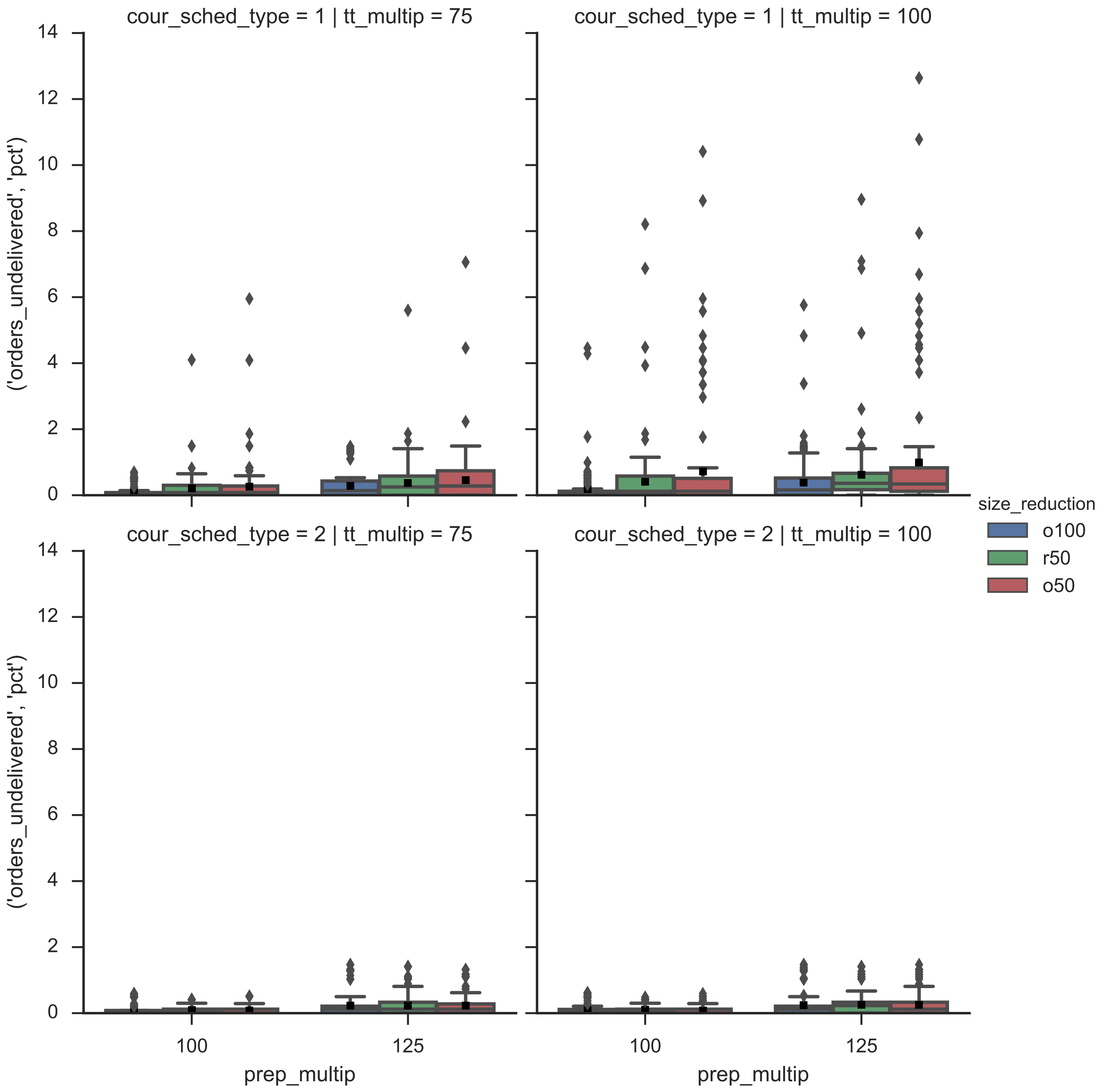

Figure 11: Percentage of orders undelivered, across different instance classes

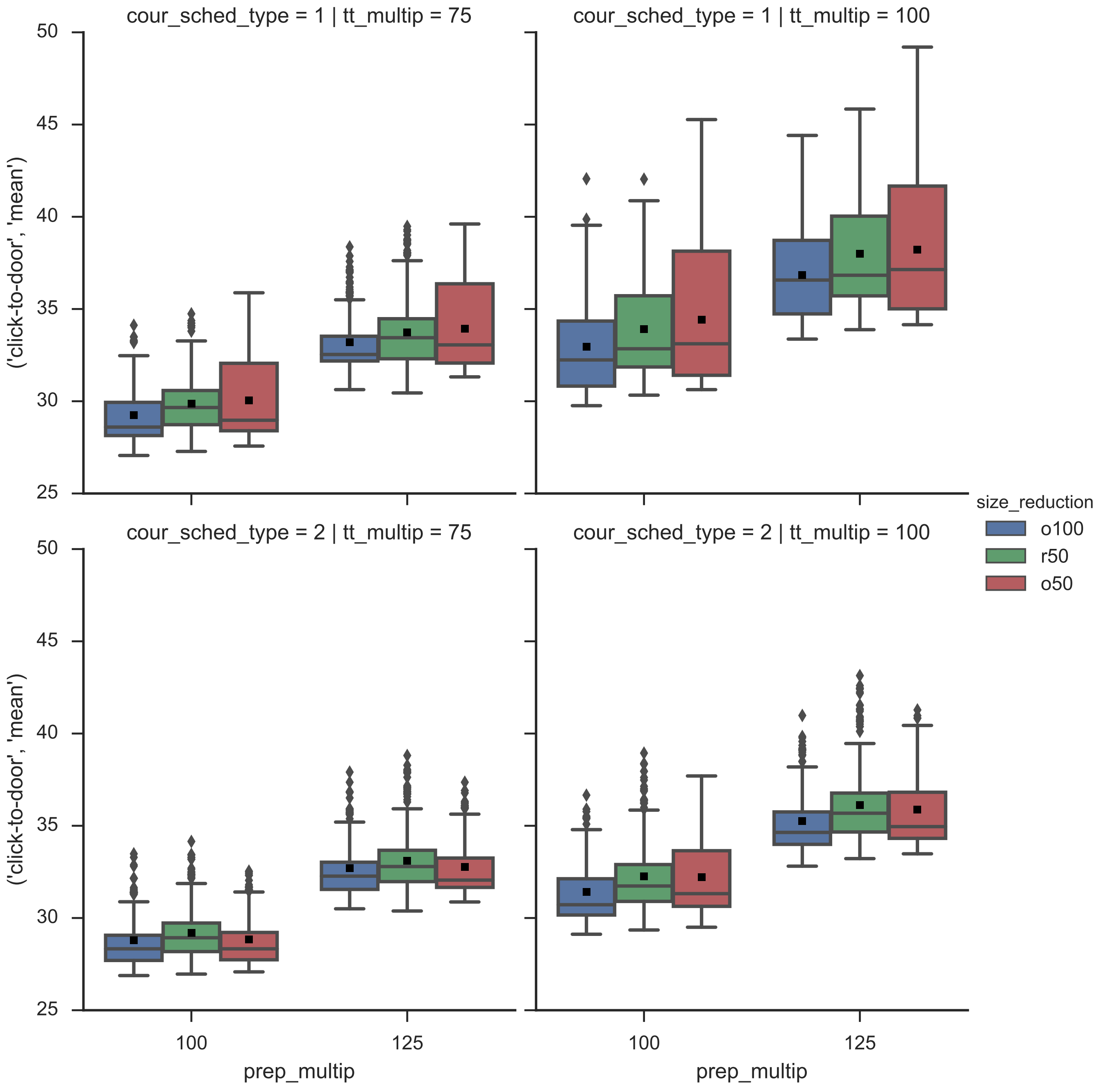

Figure 12: Click-to-door mean, across different instance classes

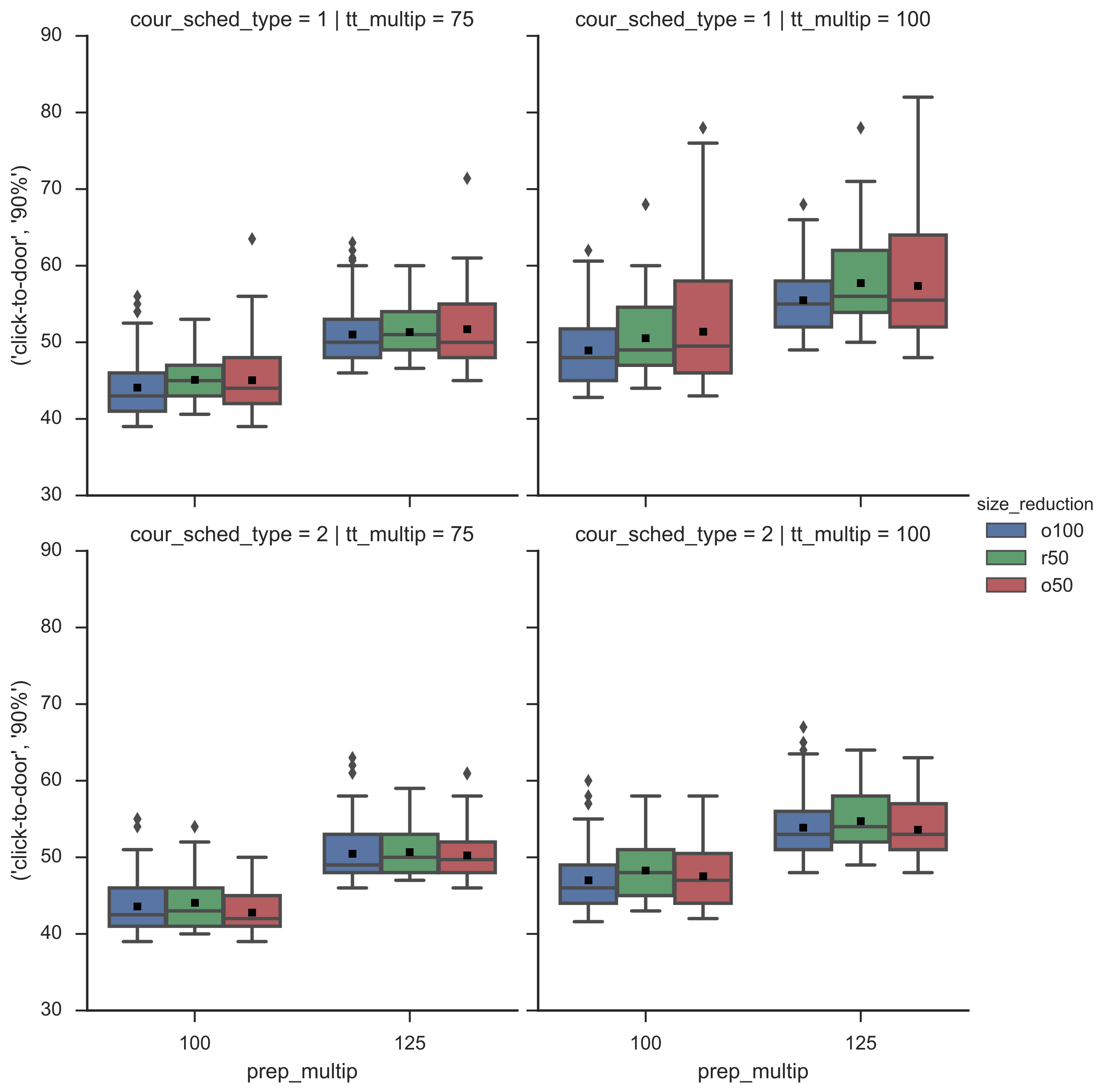

Figure 13: Click-to-door 90th percentile, across different instance classes

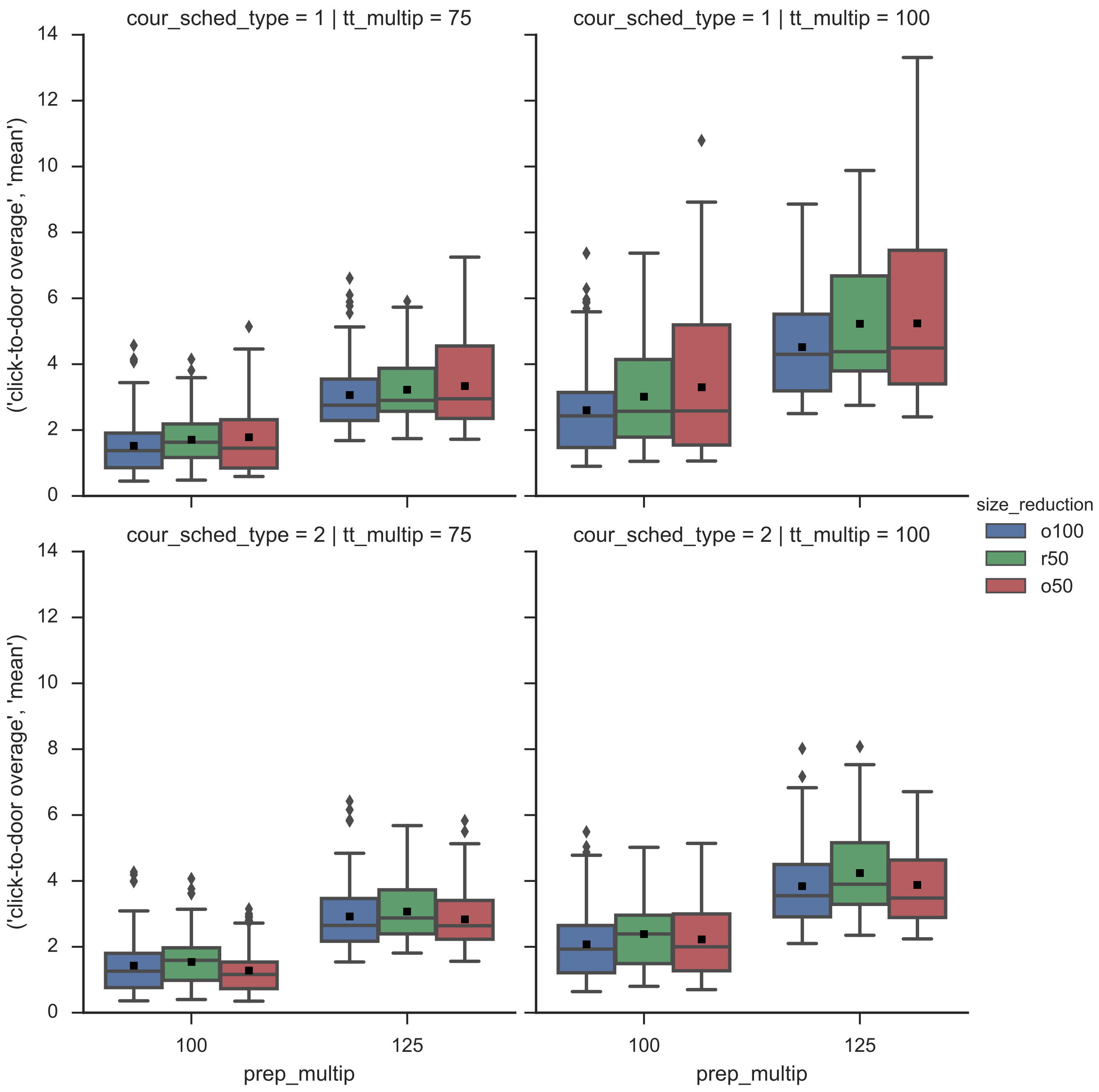

Figure 14: Click-to-door overage mean, across different instance classes

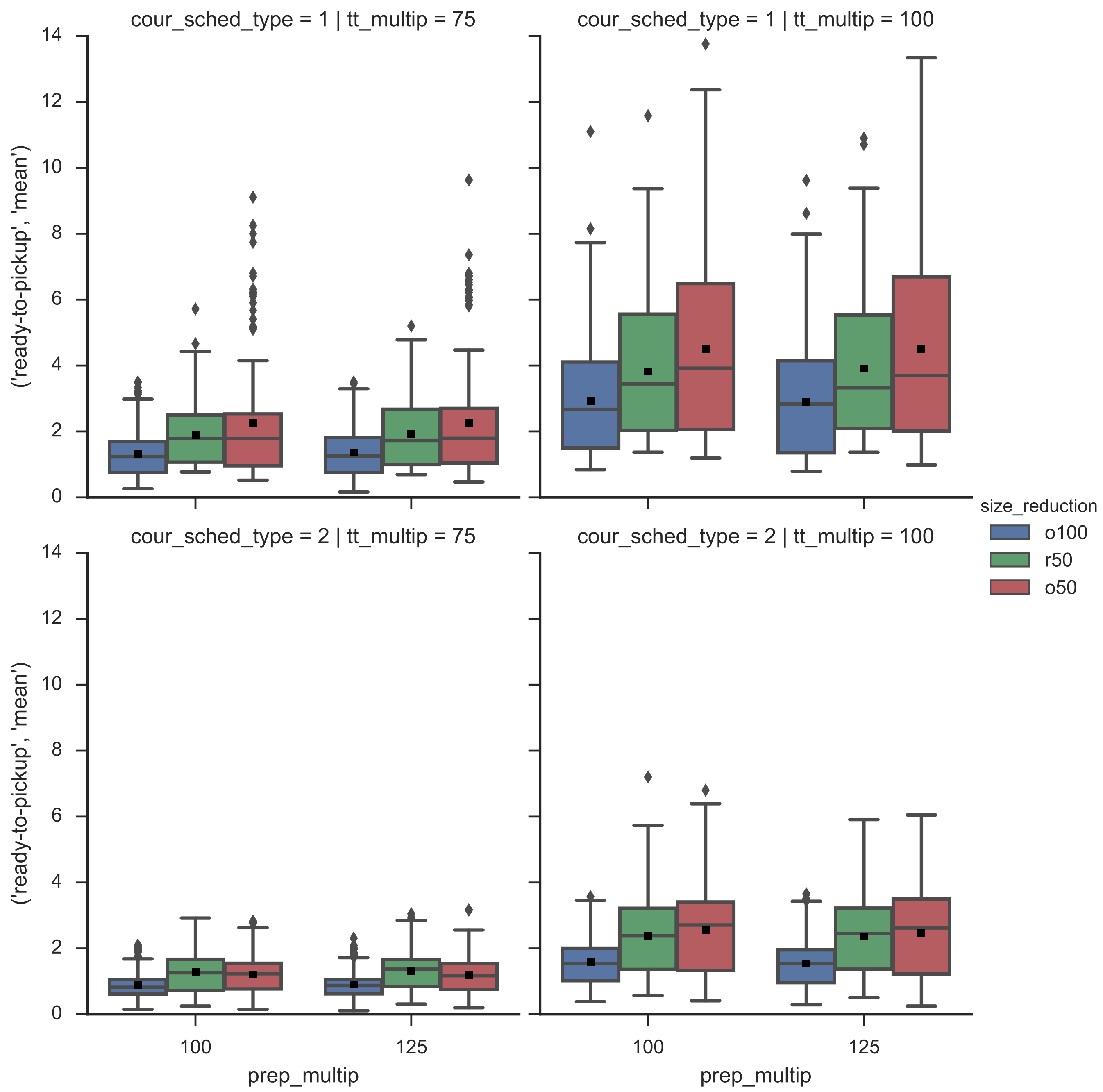

Figure 15: Ready-to-pickup mean, across different instance classes

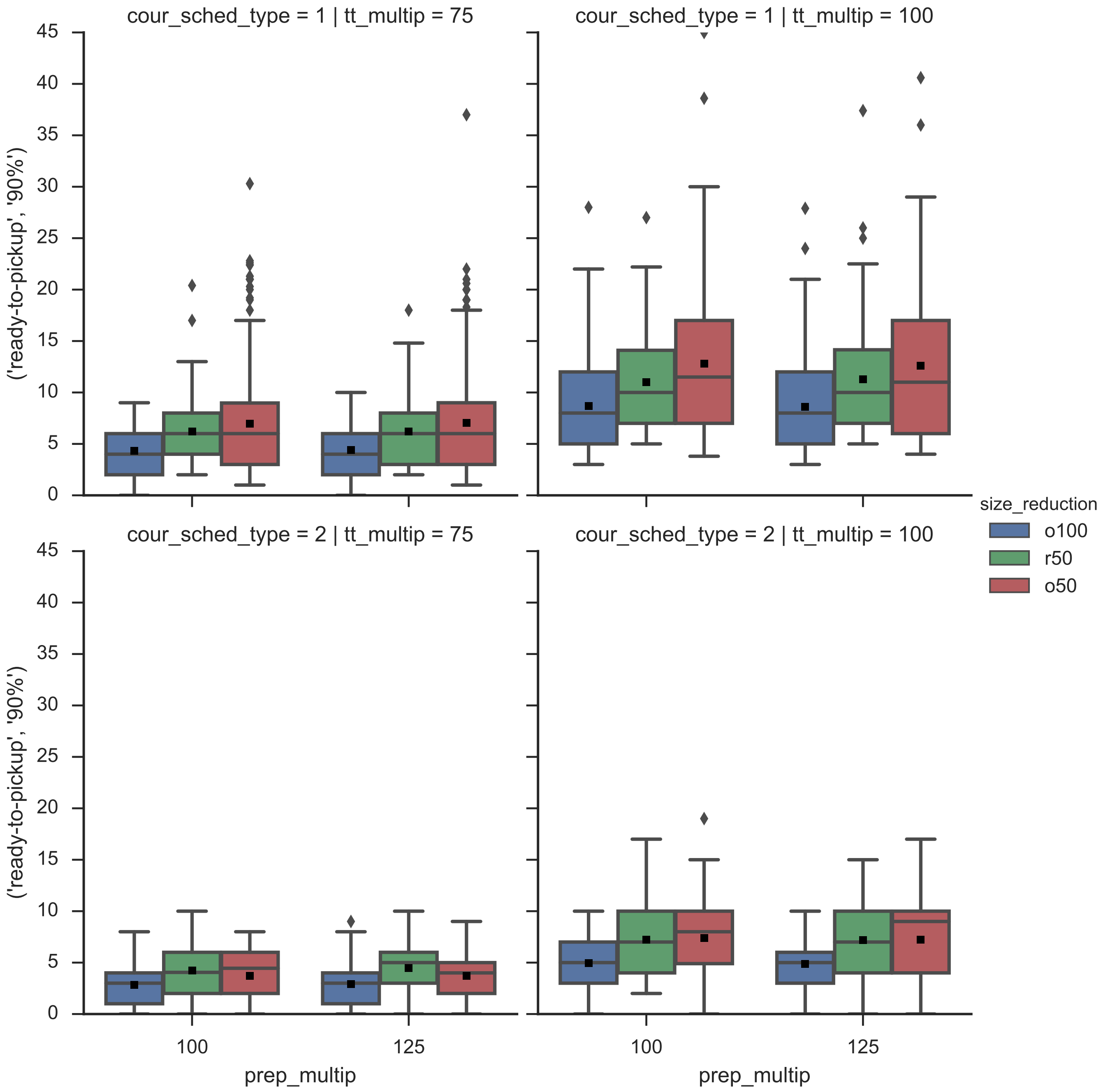

Figure 16: Ready-to-pickup 90th percentile, across different instance classes

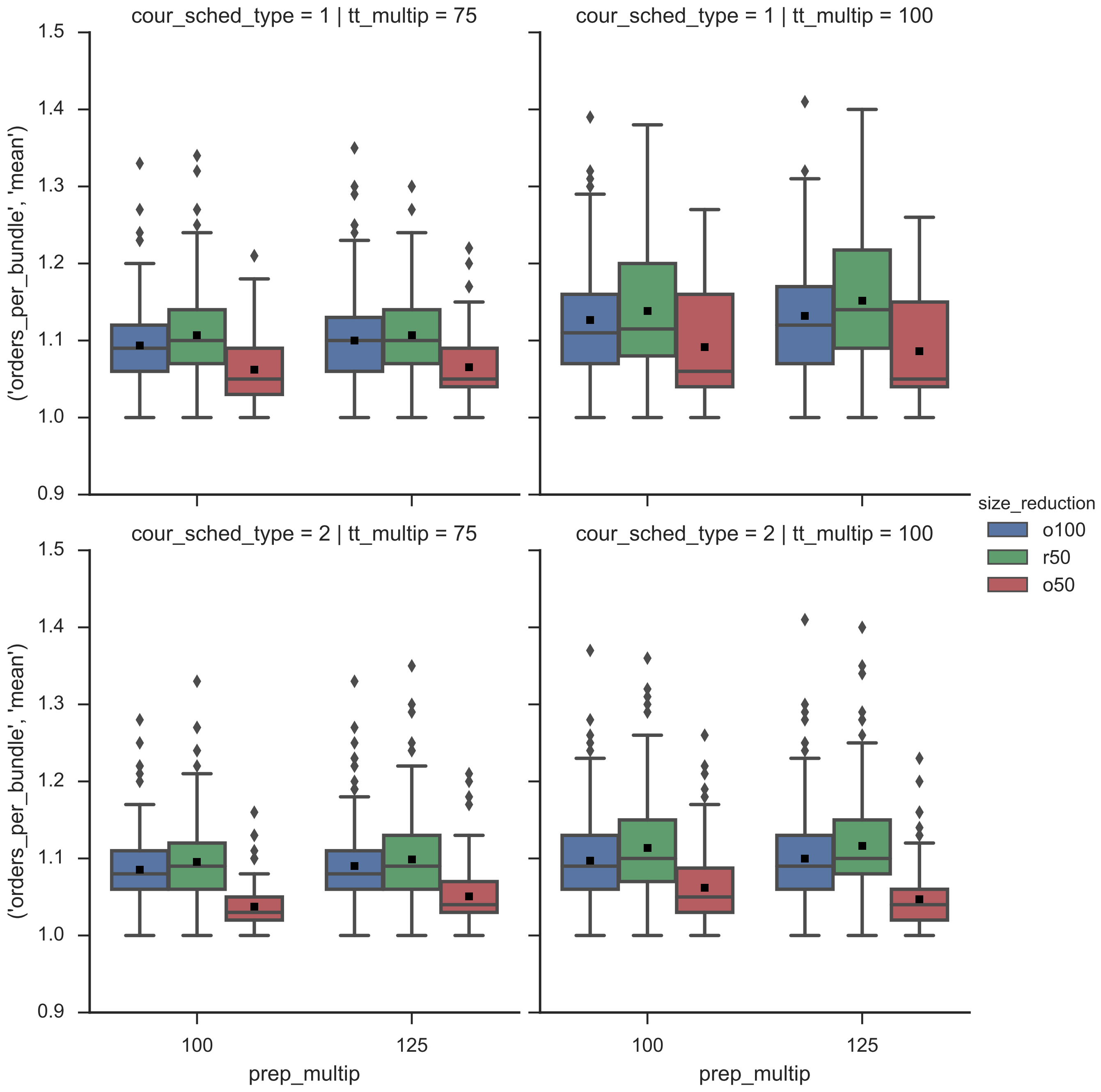

Figure 17: Mean orders per bundle, across different instance classes

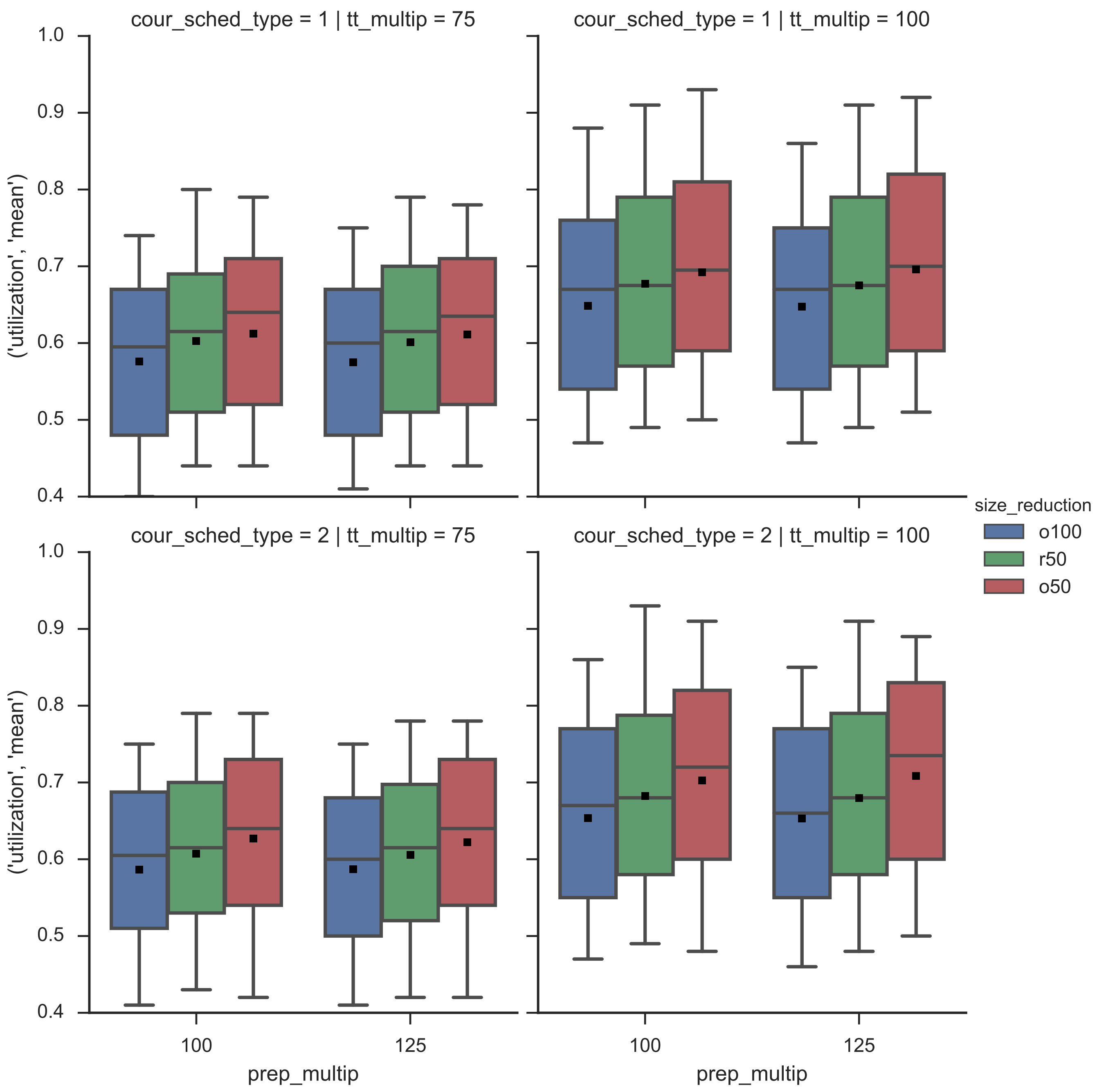

Figure 18: Courier utilization mean, across different instance classes

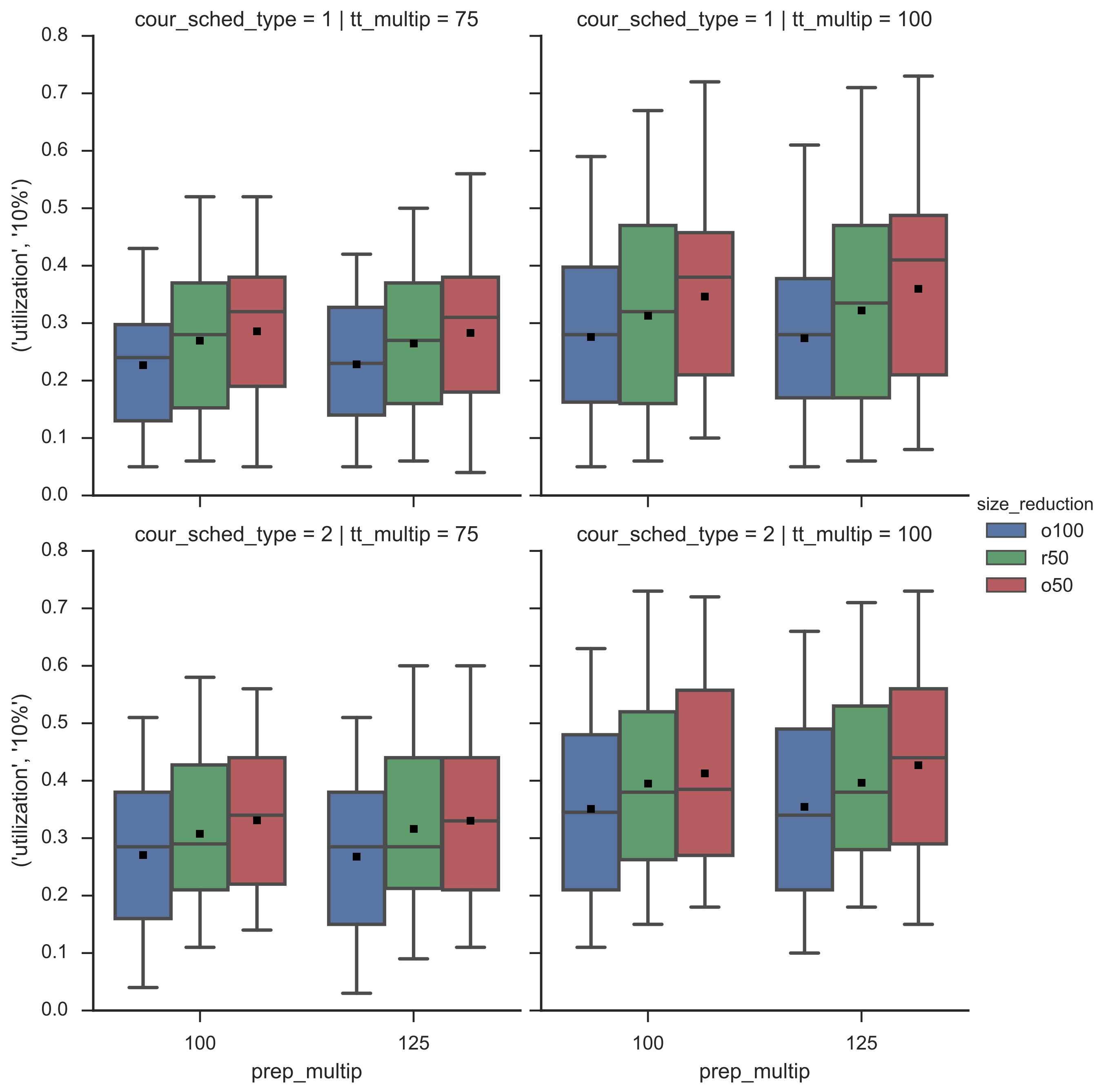

Figure 19: Courier utilization 10th percentile, across different instance classes

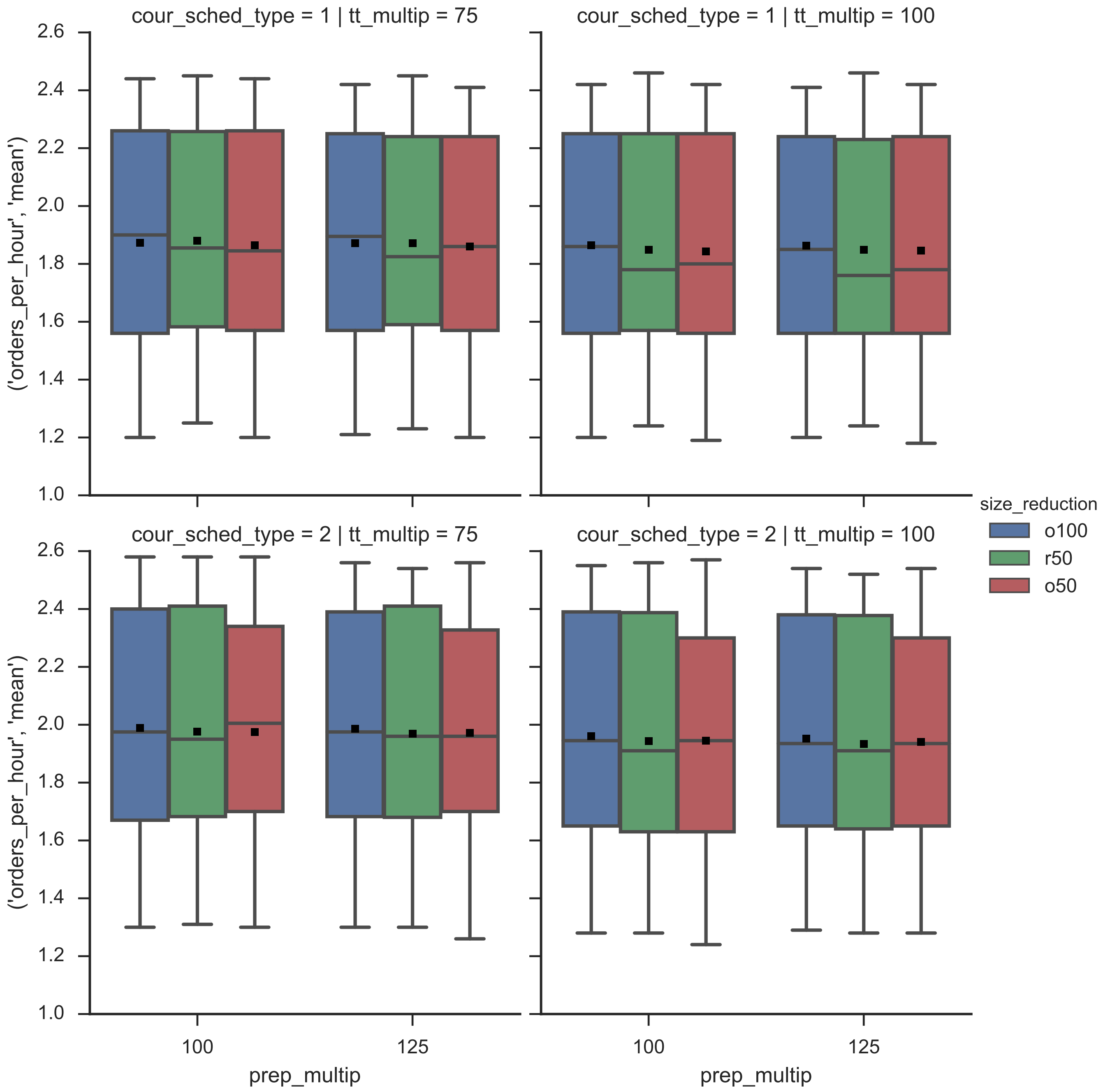

Figure 20: Mean orders delivered per courier hour, across different instance classes

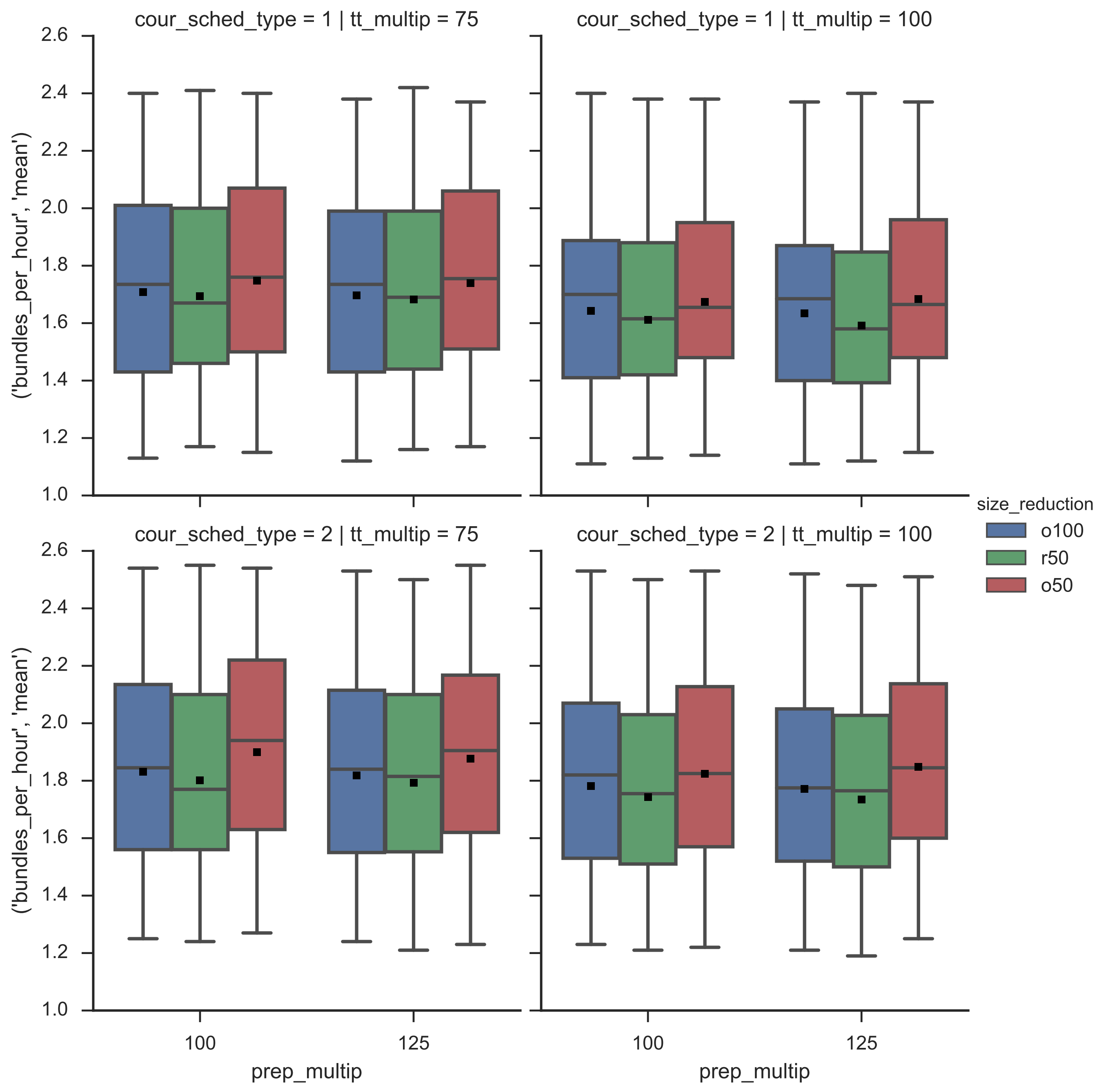

Figure 21: Mean bundles per courier hour, across different instance classes

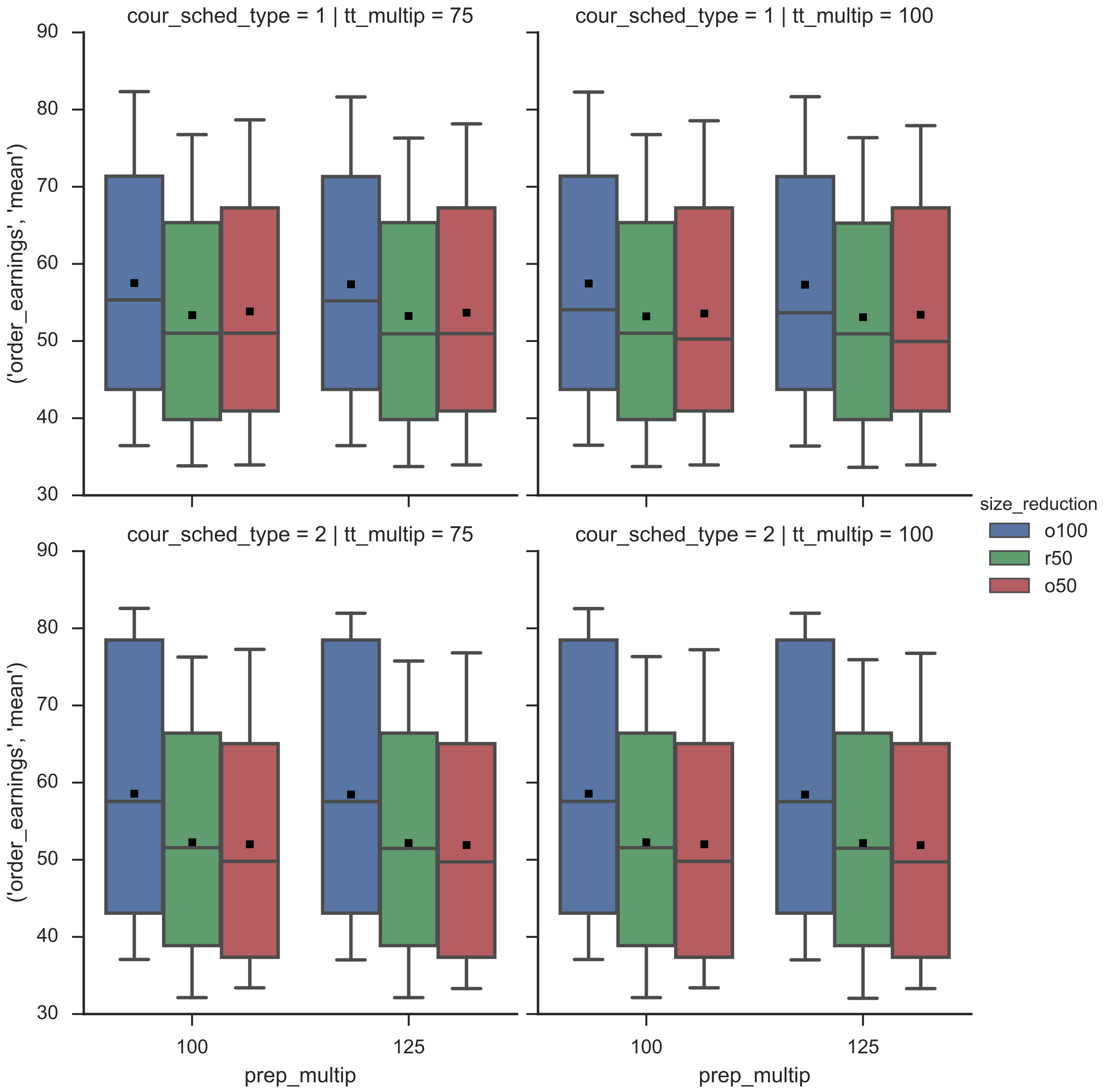

Figure 22: Average order earnings per courier, across different instance classes

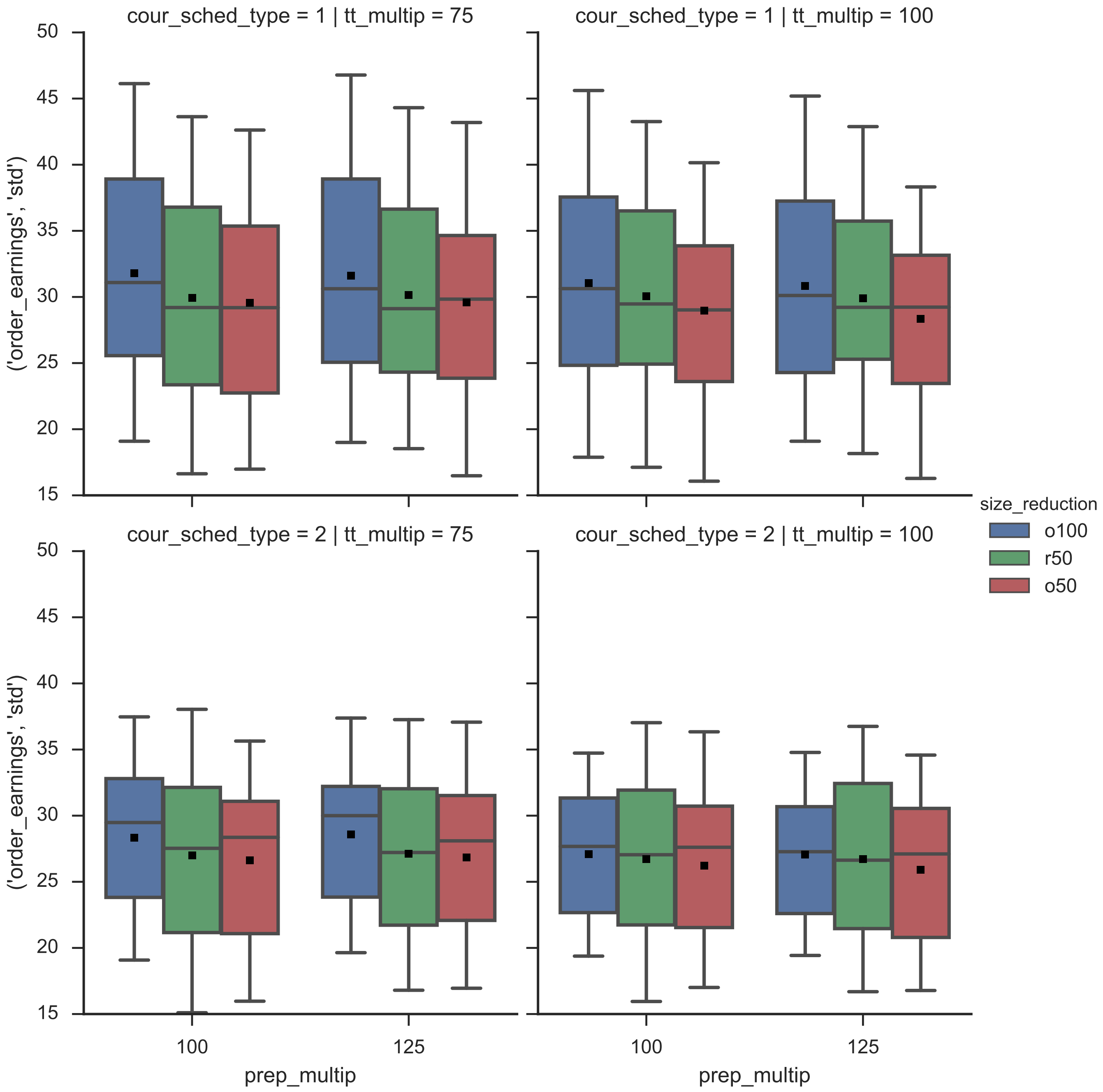

Figure 23: Standard deviation of order earnings per courier, across different instance classes

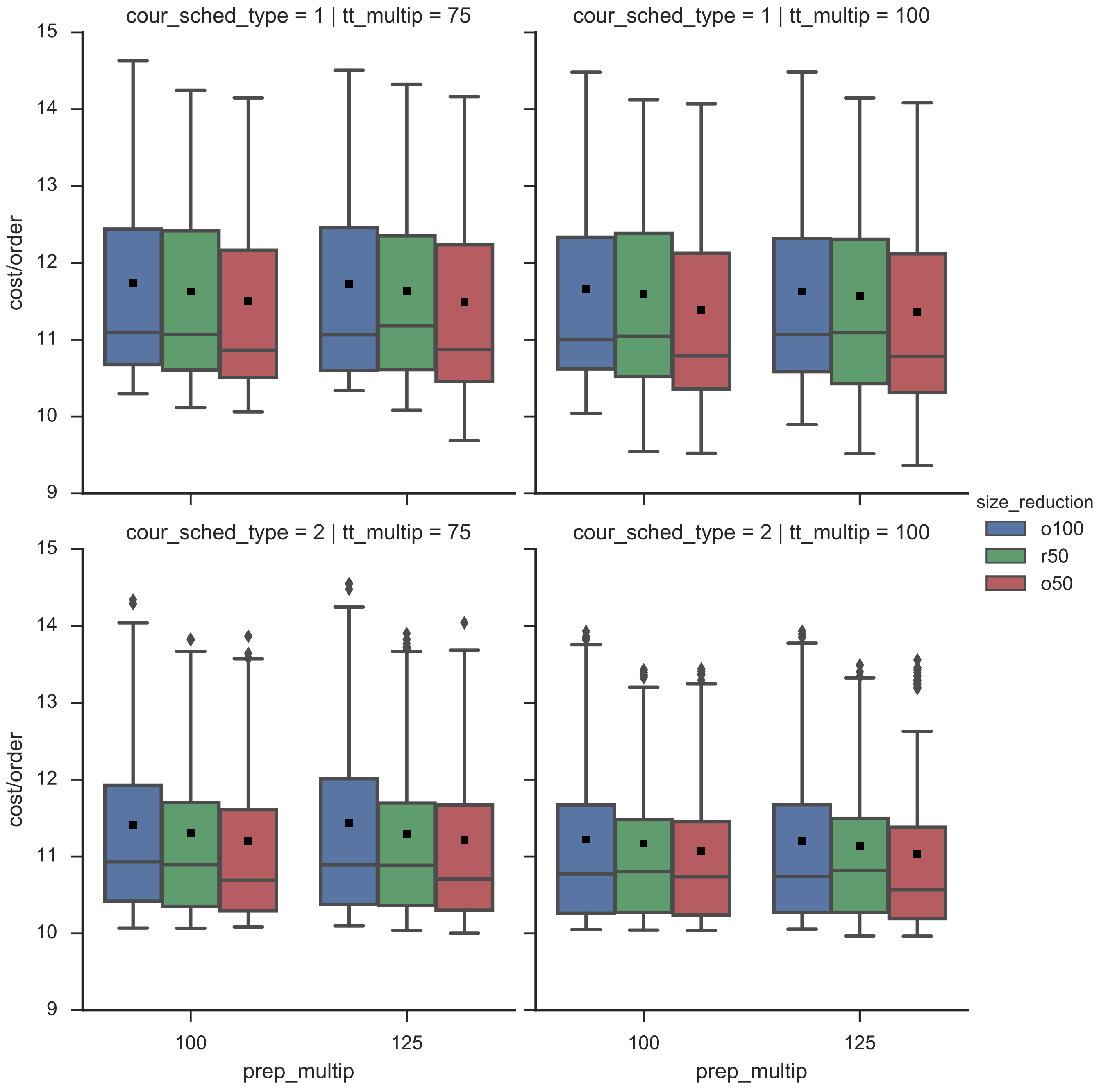

Figure 24: Average total cost per order placed, across different instance classes

Appendix G: Uncontrolled instance properties and algorithm performance

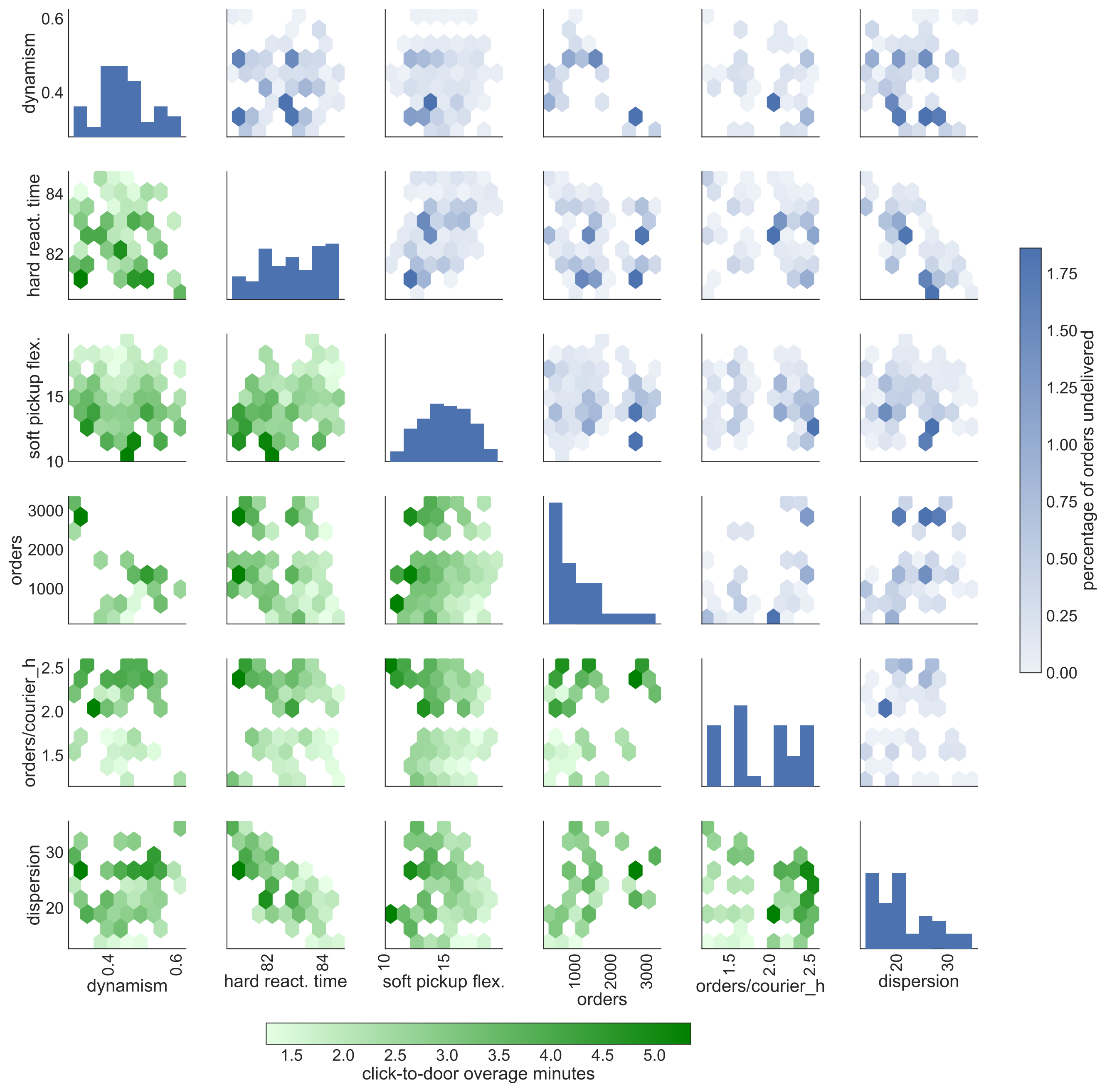

Figure 25: Percentage of orders undelivered and click-to-door overage, and their interaction with key instance features

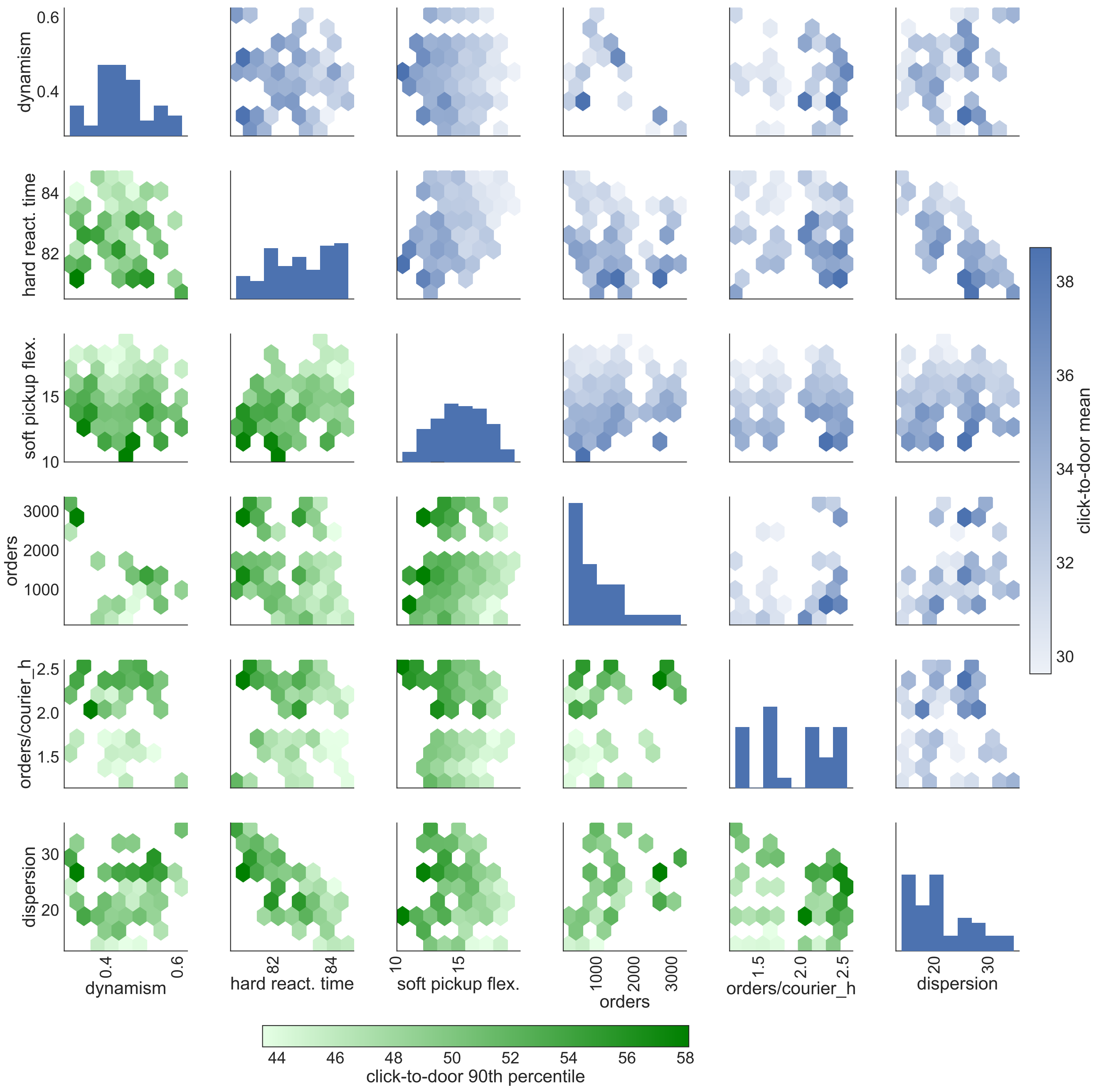

Figure 26: Click-to-door mean and 90th percentile, and their interaction with key instance features

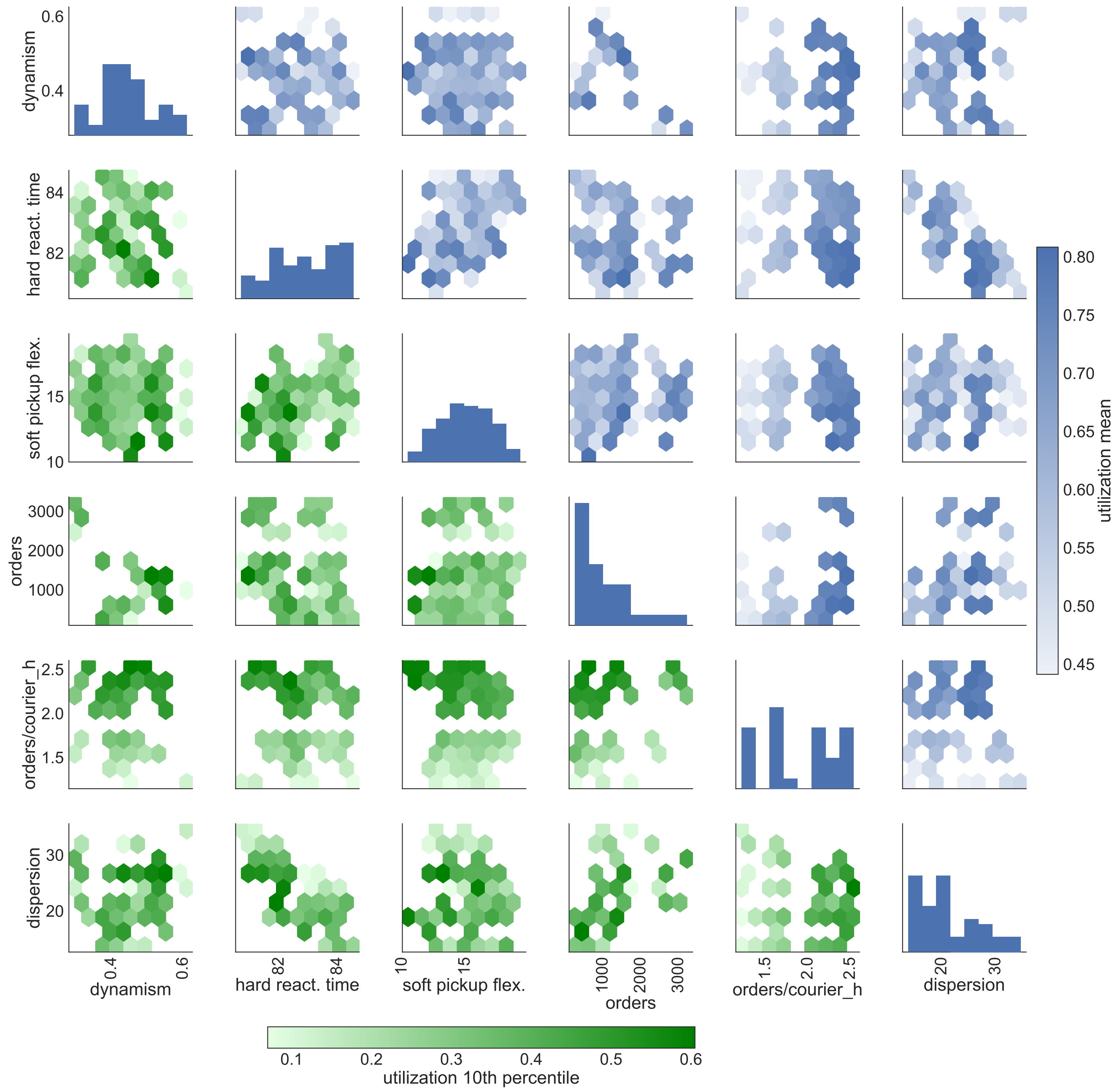

Figure 27: courier utilization mean and 10th percentile, and their interaction with key instance features

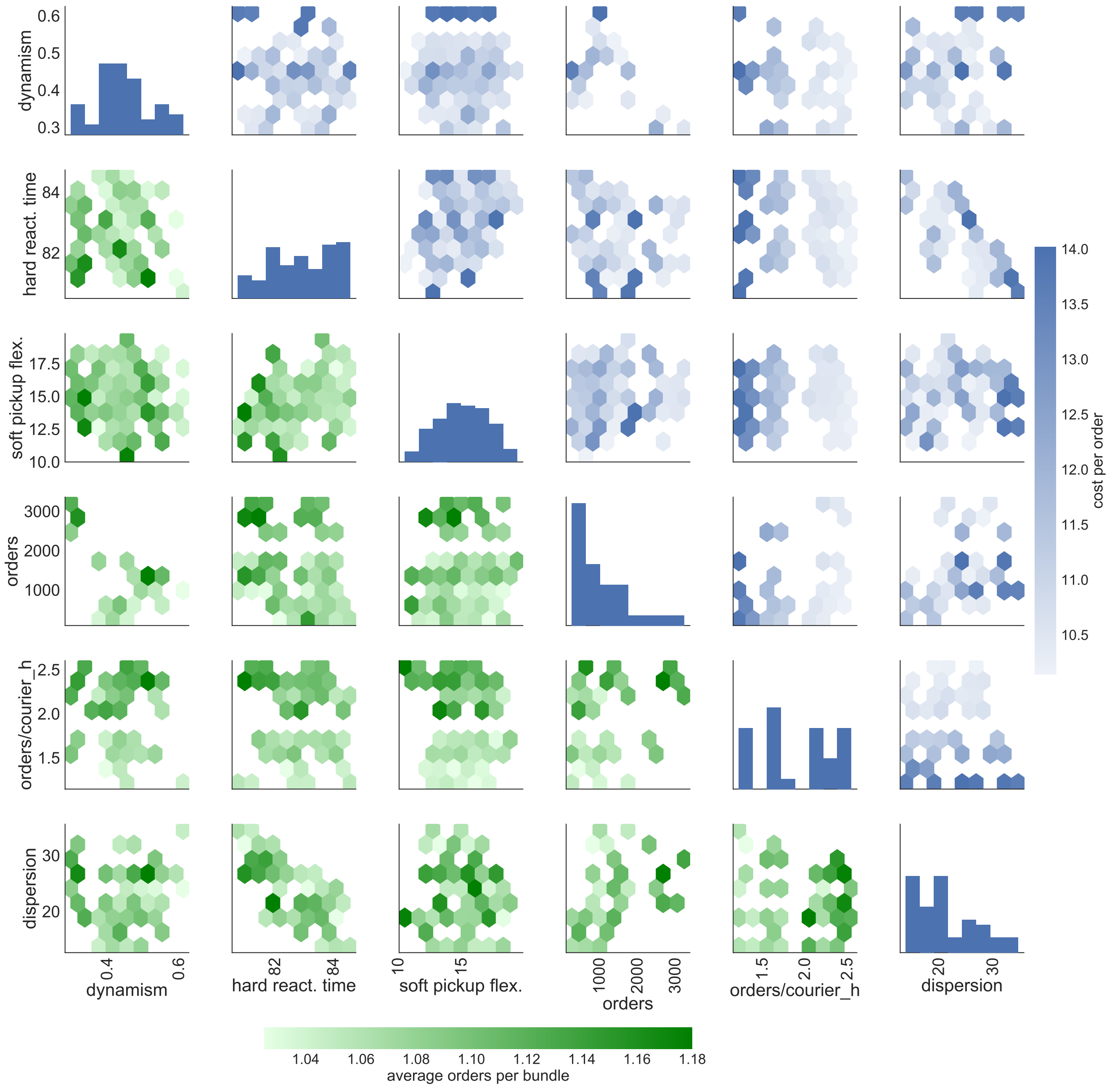

Figure 28: cost per courier hour and average number of orders per bundle, and their interaction with key instance features

Appendix H: Value of key algorithmic features

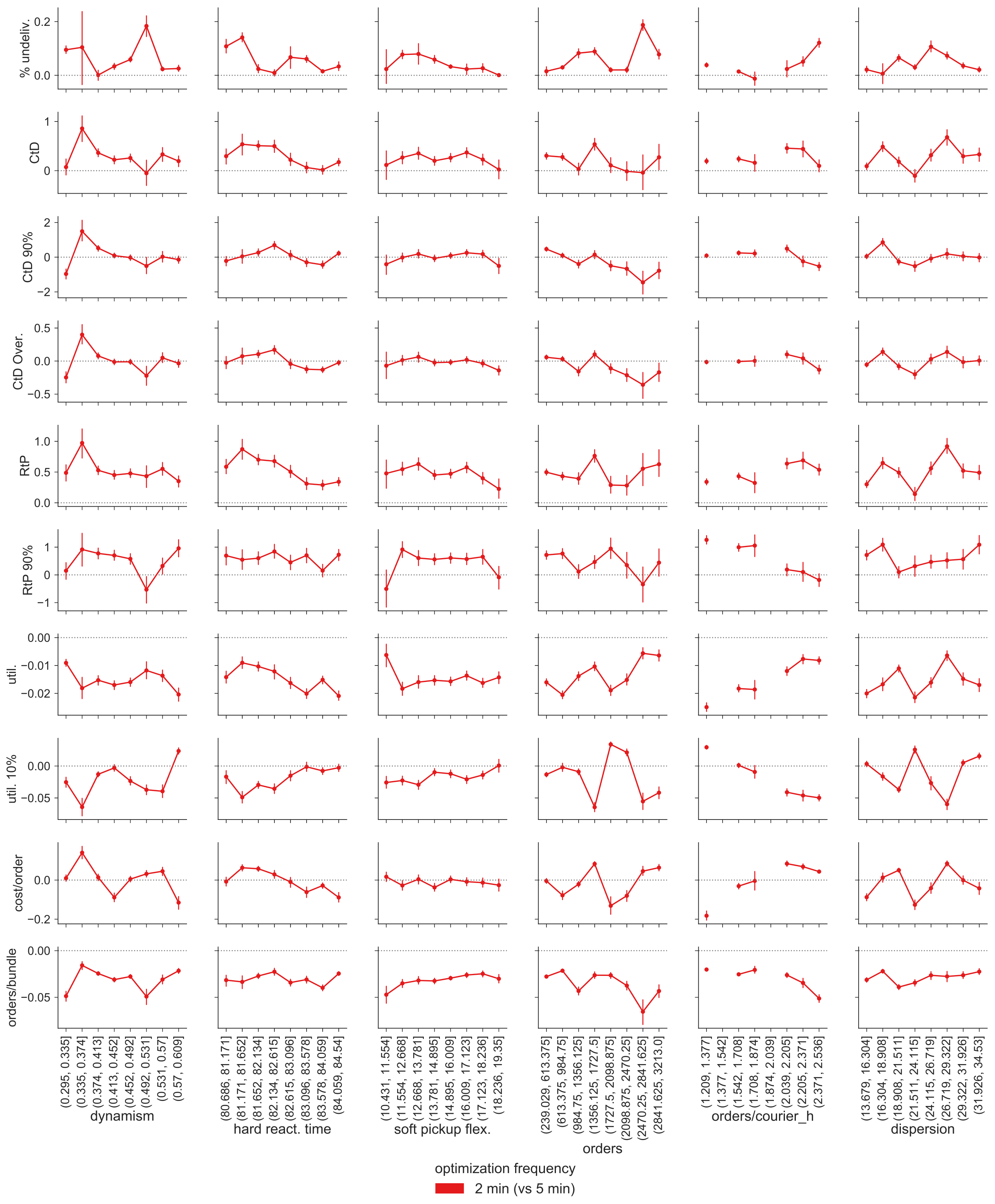

Figure 29: Performance difference of algorithm with more frequent optimizations (2 minutes, as opposed to default 5 minutes) vs instance characteristics

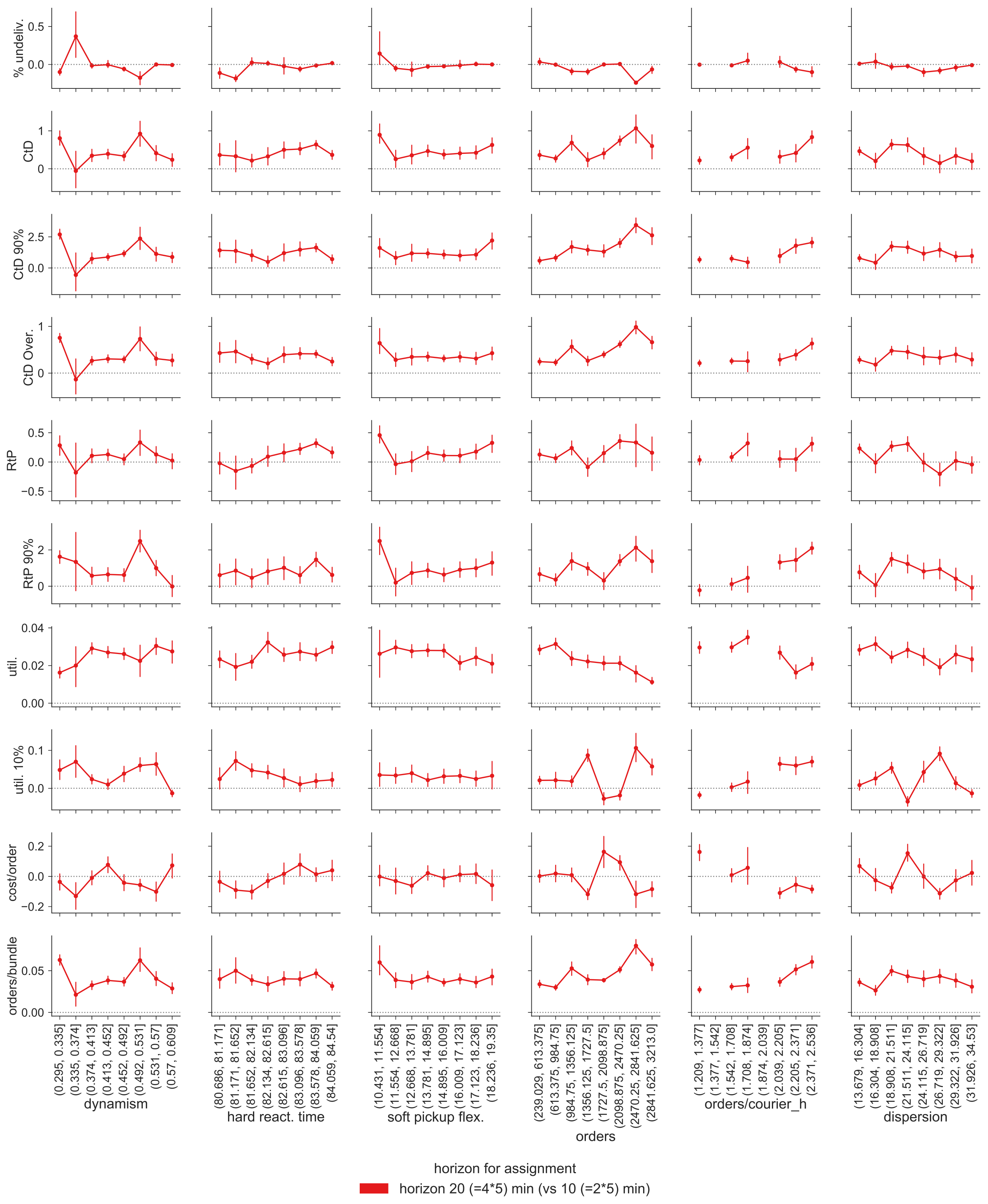

Figure 30: Performance difference of algorithm with a 20 minute horizon for assignment of orders (as opposed to default 10 minute horizon) vs instance characteristics

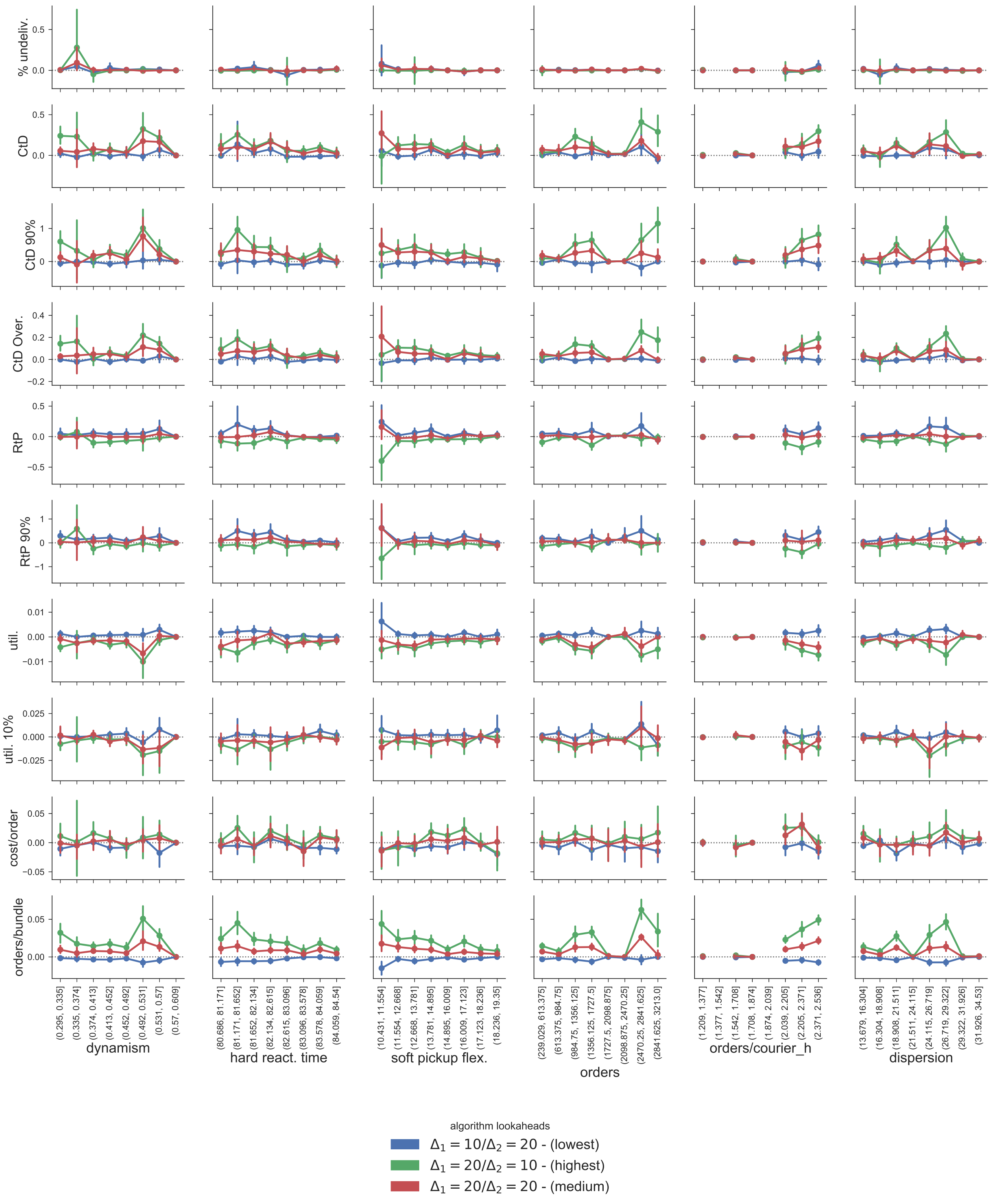

Figure 31: Performance difference of algorithms with various lookaheads for calculation of bundle target sizes (against default, which uses 10 minutes for orders and 10 minutes for couriers) vs instance characteristics

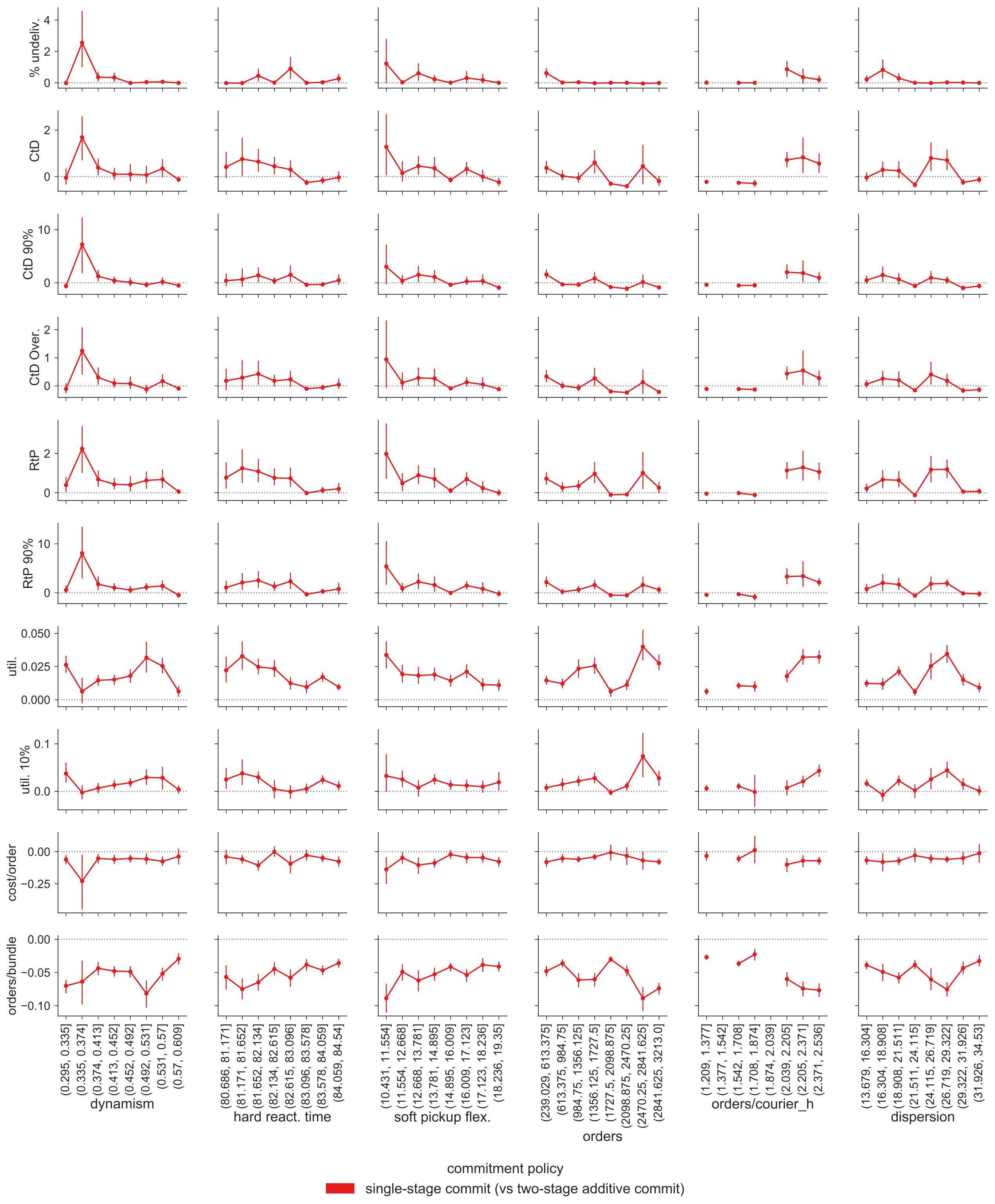

Figure 32: Performance difference of algorithm with single-stage commitment rule (as opposed to default two-stage additive rule) vs instance characteristics

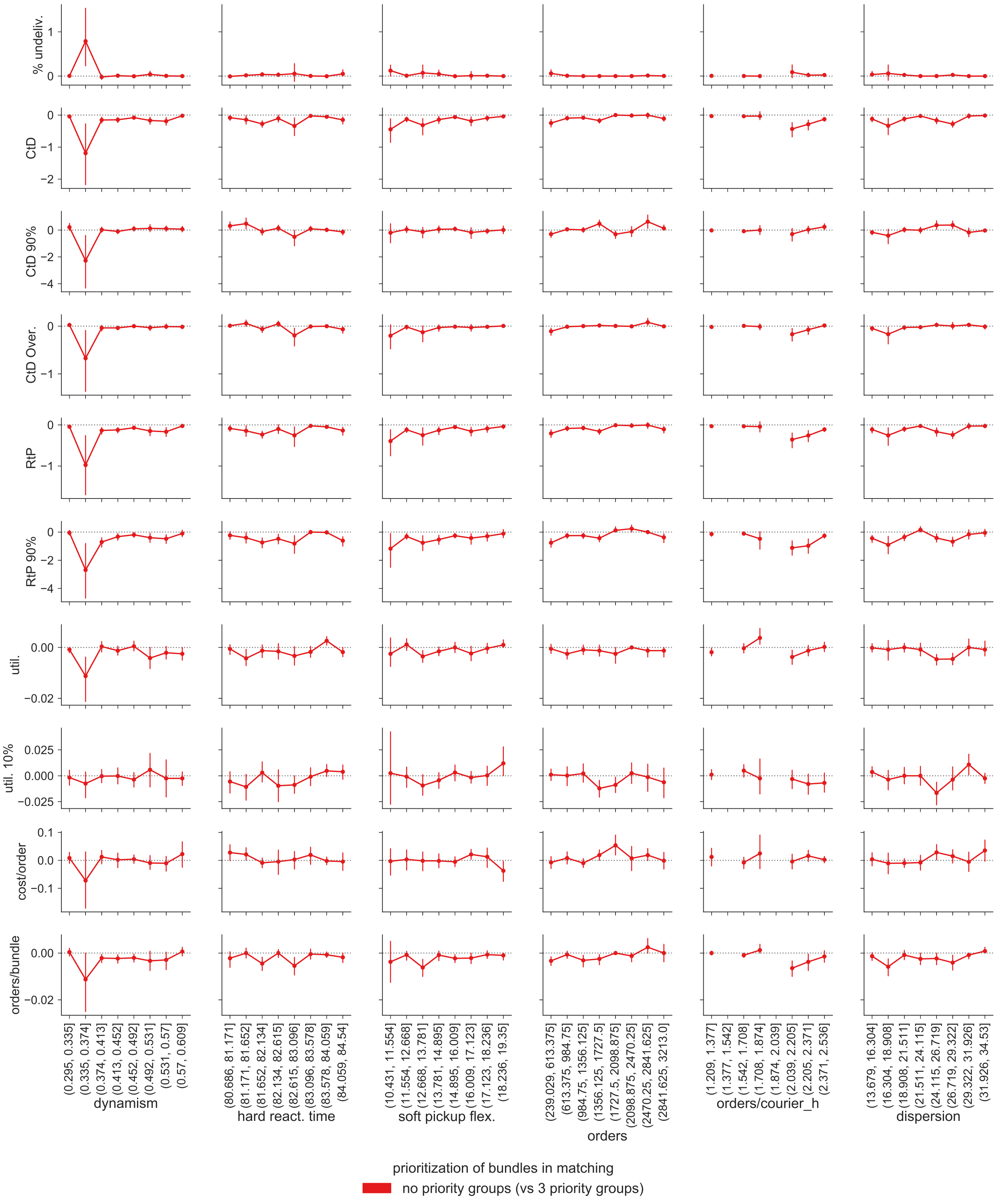

Figure 33: Performance difference of algorithm with a single matching problem per optimization run (as opposed to default sequence of 3 matchings based on priority) vs instance characteristics

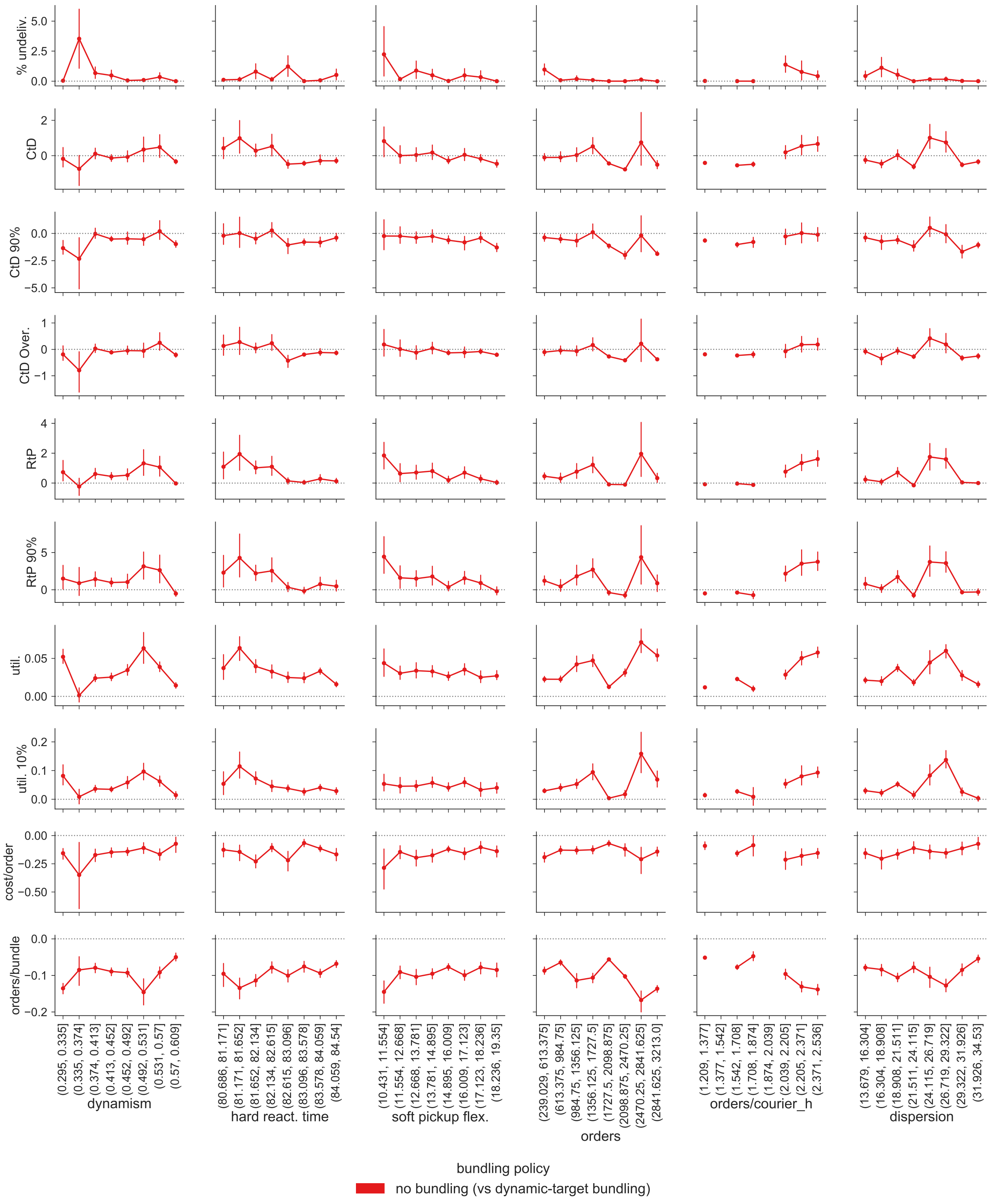

Figure 34: Performance difference of algorithm that completely forbids bundling (as opposed to default with dynamic bundling intensity) vs instance characteristics

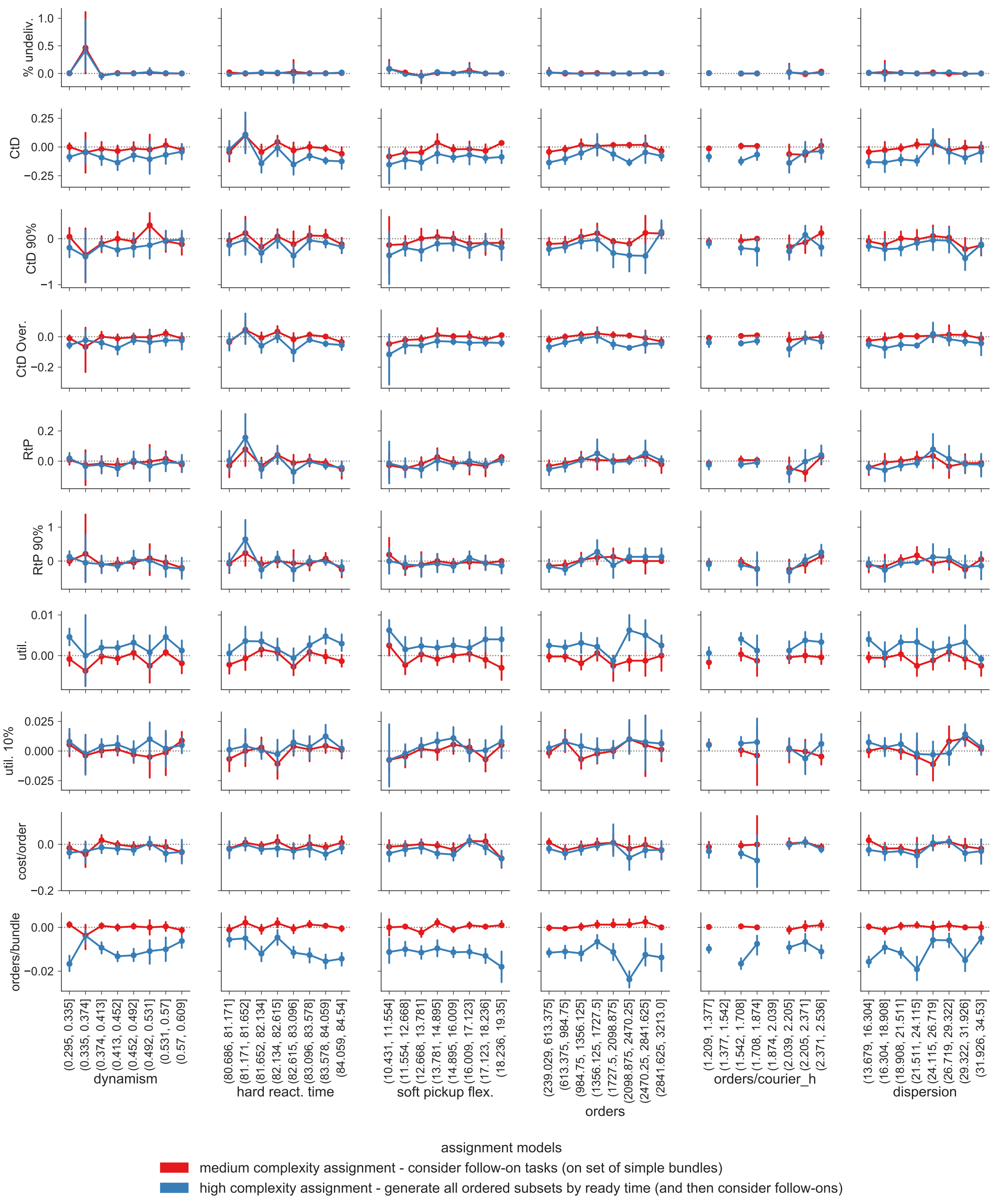

Figure 35: Performance difference of algorithm with increasingly complex assignment models (as opposed a matching linear program) vs instance characteristics