https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
ISSN 0030-364X, EISSN 1526-5463

Accepted for publication in *Operations Research*
Manuscript OPRE-2026-01-2625.R3; accepted October 7, 2026

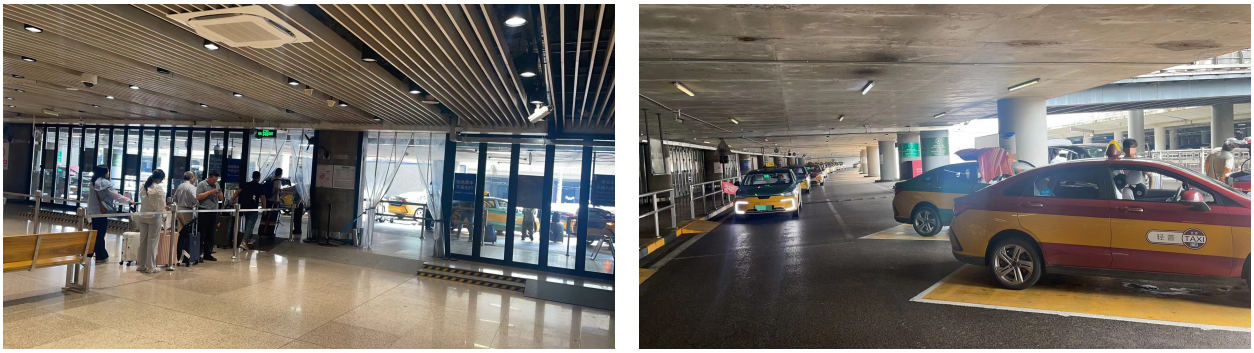
Online Appendix to The Complexity of Scheduling Pickup and Delivery Jobs

Tianyu Wang
School of Economics and Management, Beihang University, Beijing 100191, China, tianyuwang@buaa.edu.cn

Desheng Wu*
Economics and Management School, University of Chinese Academy of Sciences, Beijing 100190, China, dwu@ucas.ac.cn
*Corresponding author

David L. Olson
Department of Supply Chain Management and Analytics, University of Nebraska–Lincoln, Lincoln, Nebraska 68588-4114, USA, david.olson@unl.edu

A. Real-World Queueing and Request-Aggregation Examples



(a) Passenger groups wait in an ordered line at the taxi boarding area. The passenger line is the APDP-Q request queue.

(b) Taxis enter the common pickup facility and successively serve the passenger at the queue head.

Figure 1 Queue at Beijing Capital International Airport. Source: Authors' photographs.

Beijing Capital International Airport provides designated taxi boarding areas at Terminals 2 and 3. Figure 1(a) shows passenger groups forming a stanchioned line at such a boarding area, while Figure 1(b) shows taxis entering the common pickup facility. The request-side operating rule is straightforward. A newly arriving passenger group joins the end of the passenger line, and the next taxi that becomes available at the loading point serves the group at the front of that line. Consequently, a later passenger group cannot be selected while an earlier eligible group remains waiting.

This mechanism maps directly to APDP-Q. Each passenger group is a request. The airport taxi rank is its pickup location. The group's destination is its delivery location. The order in which groups join the passenger line defines the queue order. Whenever a driver serves the airport location, the driver must take the queue-head request. At selected airports, Uber's PIN-based pickup requires riders to first submit an UberX request and obtain a PIN before joining a designated passenger queue. Upon reaching the queue head, the rider presents the PIN to the first available driver. Lyft's Fast Match operates in a similar manner. Riders first submit their requests, receive pickup codes, and join designated passenger lines, where they are served by the next available driver upon reaching the queue head (Uber 2019, Lyft 2019).



(a) Current and forecast demand heat around the airport, together with recent calls, idle-vehicle supply, and an estimated time to the next order.

(b) Inspection of another spatial cell, including predicted and recent calls, idle vehicles, and an option to navigate toward the selected location.

Figure 2 DiDi's driver-side spatial aggregation and repositioning interface. Source: Screenshots captured by the authors.

The screenshots in Figure 2 show a DiDi driver-side demand heat map around Beijing Capital International Airport. The map partitions the service region into hexagonal cells, with darker orange cells indicating a larger number of orders according to the on-screen legend. In Figure 2(a), the panel for the selected airport location reports 28 calls in the preceding five minutes, including 14 orders in the app's discounted category, 181 currently idle vehicles, and an estimated 9–14 minutes until an order is received. The tabs at the top allow the driver to inspect the current heat level or forecasts 15 and 30 minutes ahead. In Figure 2(b), the driver inspects another location 7.4 km away. The panel reports fewer than three predicted calls in the next five minutes, three calls in the preceding five minutes, and two idle vehicles, and provides a button for navigating to that location.

DiDi's official open-platform documentation defines a passenger-side line-up function for peak periods and areas in which passenger demand exceeds driver supply. Under that function, ride requests are queued and vehicles are assigned on a first-come, first-served basis. The interface can expose the passenger's current rank and estimated waiting time (DiDi Open Platform 2026). This mechanism is a direct operational analogue of APDP in queueing mode.

B. A Motivating Example

Queueing mechanisms are widely used in applications such as ride-hailing, ambulance dispatching, and airport ground transportation. Appendix A presents several real-world illustrations, including the physical taxi queue at Beijing Capital International Airport, DiDi's demand-aggregation and queueing mechanisms, and comparable virtual airport queues operated by Uber and Lyft.

Intuitively, APDP in queueing mode (APDP-Q) appears simpler to solve as the service order of requests in a queue follows their submission order. In other words, the queue heads are always served first. One may presume that drivers always serve the queue heads of their current or nearest locations. We term this rule Nearest-Queuehead-First (NQF).

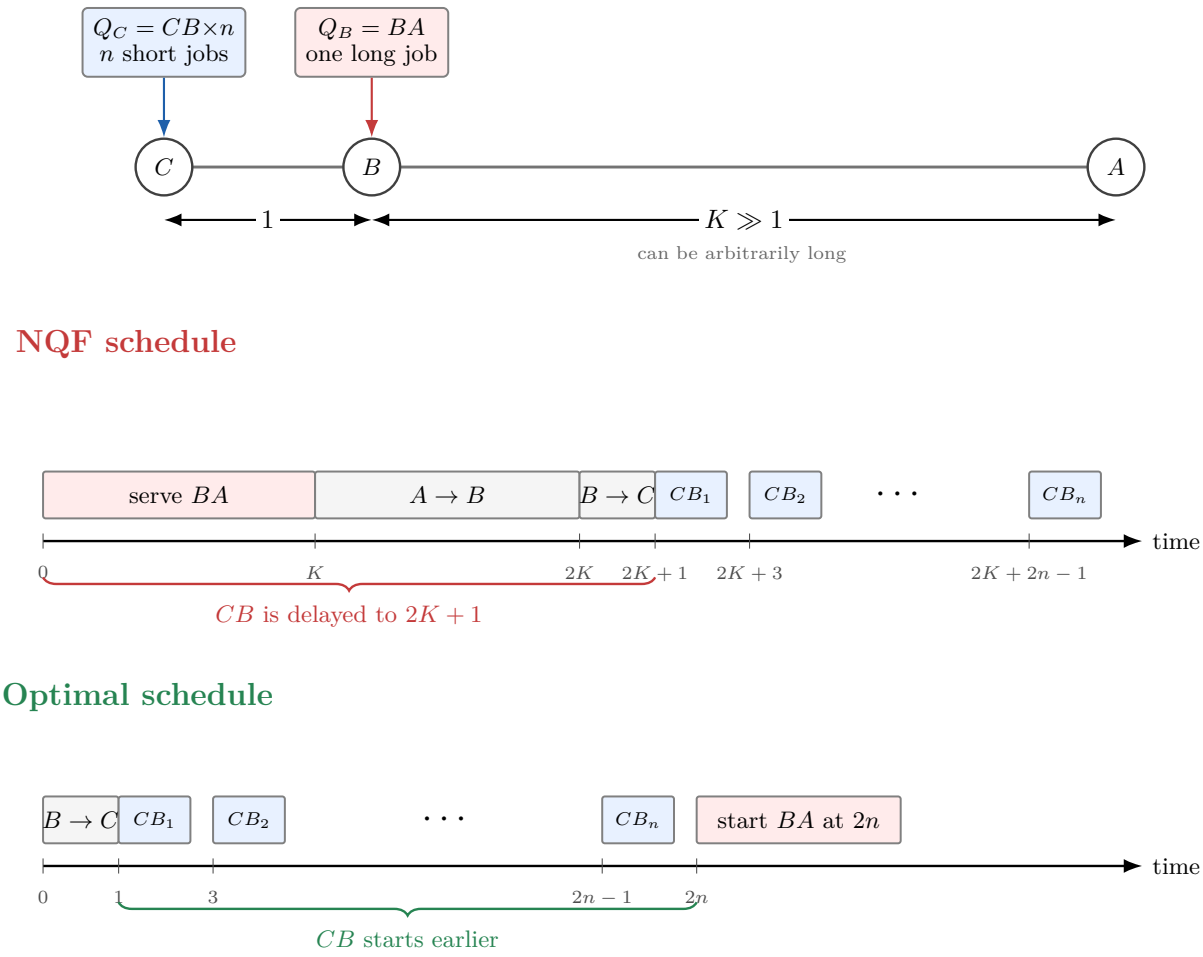


Figure 3 Three collinear locations A, B, C satisfy $l_{A,B} = K$ and $l_{B,C} = 1$, a single driver initially stays at B. Location B has one job going to A, while location C has n jobs going to B.

Unfortunately, NQF does not always yield an optimal solution and can, in fact, perform arbitrarily poorly, as presented in Proposition 3. Setting $K = 5$ and $n = 2$ in the example illustrated in Fig. 3, the optimal total waiting time is $1 + 3 + 4 = 8$, whereas NQF yields a total waiting time of $0 + 11 + 13 = 24$.

We observe that drivers may travel to another location instead of serving the job at the head of the queue of their current location. The key challenge lies in deciding which queue head to serve. In pooling mode, drivers face even more choices, as they can serve any job and are not limited to queue heads. Without fixing the map size, APDP in pooling mode (APDP-P) is a generalized vehicle routing problem, which is strongly NP-hard. This implies that simple rule-based algorithms, such as always serving the shortest trip, are not optimal for APDP-P. Consequently, solving both APDP-P and APDP-Q and determining their computational complexity are non-trivial.

C. Related Works

This paper contributes to the study of computational complexity in both PDP and scheduling problems.

As even TSP is strongly NP-hard, research on the complexity of VRP and PDP is not extensive. Some studies focus on the variants of PDP. For example, Berbeglia et al. (2012b) studied the complexity of PDP with fixed partial routes. de Paepe et al. (2004) adopted the notation from scheduling theory to classify PDP and surveyed the complexity results for different objectives and constraints. Our findings align with the survey's observation that the computational complexity of PDP is highly sensitive to the choice of objective functions. However, the APDP in our study presents a unique case not covered in this survey, as the introduction of job queues and pools makes its computational complexity more intricate than that of problems with conventional constraints. Our model emerges from the nature of request aggregation in platform economies within major cities and applies to real-world scenarios such as ride-hailing (Bertsimas et al. 2019), electric vehicle routing (Desaulniers et al. 2016), and meal delivery (Ulmer et al. 2021).

Haddad et al. (2018) proved the NP-hardness of multi-vehicle one-to-one pickup and delivery with split loads. Agnetis et al. (2023) showed the NP-hard nature of the proposed meal delivery problems. There has also been research that proved the NP-hardness of sub-problems encountered in exact solving frameworks, such as the pricing problem in column generation and branch-and-price frameworks (Dror 1994, Spliet 2023). Most contemporary research on PDP focuses on solving NP-hard delivery problems, utilizing exact methods such as the branch-price-and-cut algorithms of Cherkesly et al. (2016) for problems with time windows and multiple stacks, the reinforcement learning method of Auad et al. (2023) for dynamic courier capacity acquisition, and the heuristics proposed by Berbeglia et al. (2012a) for static and dynamic PDP.

A machine scheduling problem assigns jobs to machines and sequences the jobs on each machine, with the machines operating in parallel. The queueing mode of APDP resembles precedence-constrained scheduling problems, where some jobs cannot be started before the completion of some others. The problem is generally NP-hard even when the in-degree of the precedence graph is limited to 2 (Lenstra and

Rinnooy Kan 1978). Unlike PDP research, studies on scheduling problems continue to explore both polynomial exact algorithms and NP-hardness proofs for specific models, as their complexity varies significantly with constraints. The problem with an in-tree precedence graph and equal-length jobs can be solved by the well-known algorithm of Hu (1961) for minimum makespan, or by Monma (1982) for lateness. However, minimizing the total waiting time, as in APDP, is strongly NP-hard (Wang and Bellenguez-Morineau 2019). When it comes to arbitrary processing times, minimizing the makespan and minimizing waiting times are both strongly NP-hard, as proved by Du et al. (1991). Similarly, when preemption is allowed, the makespan can be minimized in polynomial time by Muntz and Coffman Jr (1970) but minimizing the waiting times is still NP-hard, as proven by Du et al. (1991). Solving the fixed-parameter tractable version (Cygan et al. 2015) or using approximative methods are two common solutions for NP-hard scheduling problems, as in Hall et al. (1997), Timkovsky (2003). The pooling mode of APDP is analogous to scheduling problems without precedence constraints. Horn (1973) proved the polynomial solvability of this problem by reducing to an assignment problem.

D. A Formulation of ILP for APDP

In the compact formulation of ILP, we use the binary decision variable $x^d_{o_1,o_2}$, which is equal to 1 if and only if job o_2 is the next job served by driver d after serving o_1, and the nonnegative continuous variable t_o, which represents the waiting time of job o. For each driver d, we introduce a zeroth dummy job o_d to set the boundary conditions. Let $\mathcal{O}$ and $\mathcal{O}_d = \{o_d : d \in \mathcal{D}\}$ be the sets of non-dummy and dummy jobs, respectively.

For a job $o \in \mathcal{O}$, let $P(o)$ and $D(o)$ denote its pickup and delivery locations, and let L^0_d denote the initial location of driver d. We define $x^d_{o_1,o_2}$ only for

$$\mathcal{A}_d = (\{o_d\} \times \mathcal{O}) \cup \{(o_1, o_2) \in \mathcal{O} \times \mathcal{O} : o_1 \neq o_2\}.$$

For every $(o_1, o_2) \in \mathcal{A}_d$, let

$$l_{o_1,o_2} = \begin{cases} l_{P(o_1),D(o_1)} + l_{D(o_1),P(o_2)}, & o_1 \in \mathcal{O}, \\ l_{L^0_d,P(o_2)}, & o_1 = o_d, \end{cases}$$

which is the elapsed time from the start of o_1 to the earliest possible start of o_2. We also introduce a continuous ordering variable u_o for each $o \in \mathcal{O}$. Let

$$\bar{l} = \max_{d \in \mathcal{D},\, (o_1,o_2) \in \mathcal{A}_d} l_{o_1,o_2}, \qquad U = |\mathcal{O}|\bar{l}, \qquad M = (|\mathcal{O}| + |\mathcal{D}|)\bar{l}.$$

Without loss of optimality, U is a valid bound: in an earliest feasible schedule, any simple path in the timing-constraint graph contains at most $|\mathcal{O}|$ route arcs, while queue-precedence arcs have weight zero. Moreover, $M \geq U + \bar{l}$. The problem is to route the drivers, and their routes can be recovered from the values of $x^d_{o_1,o_2}$.

The formulation of APDP-Q is as follows:

$$\min \sum_{o\in\mathcal{O}} t_o \tag{1}$$

s.t.

$$\sum_{d\in\mathcal{D}} \sum_{o_1:(o_1,o_2)\in\mathcal{A}_d} x^d_{o_1,o_2} = 1 \qquad \forall o_2 \in \mathcal{O}, \tag{2}$$

$$\sum_{d\in\mathcal{D}} \sum_{o_2:(o_1,o_2)\in\mathcal{A}_d} x^d_{o_1,o_2} \le 1 \qquad \forall o_1 \in \mathcal{O}\cup\mathcal{O}_d, \tag{3}$$

$$\sum_{o_2:(o_1,o_2)\in\mathcal{A}_d} x^d_{o_1,o_2} \le \sum_{o:(o,o_1)\in\mathcal{A}_d} x^d_{o,o_1} \qquad \forall d\in\mathcal{D},\ o_1\in\mathcal{O}, \tag{4}$$

$$t_{o_d} = 0 \qquad \forall d\in\mathcal{D}, \tag{5}$$

$$t_{o_1} \le t_{o_2} \qquad \forall o_1,o_2\in\mathcal{O}:\ o_1 \prec o_2, \tag{6}$$

$$M(1-x^d_{o_1,o_2}) + t_{o_2} \ge t_{o_1} + l_{o_1,o_2} \qquad \forall d\in\mathcal{D},\ (o_1,o_2)\in\mathcal{A}_d, \tag{7}$$

$$u_{o_2} \ge u_{o_1} + 1 - |\mathcal{O}|\left(1 - \sum_{d\in\mathcal{D}} x^d_{o_1,o_2}\right) \qquad \forall o_1,o_2\in\mathcal{O}:\ o_1 \ne o_2, \tag{8}$$

$$x^d_{o_1,o_2} \in \{0,1\} \qquad \forall d\in\mathcal{D},\ (o_1,o_2)\in\mathcal{A}_d, \tag{9}$$

$$0 \le t_o \le U, \qquad 1 \le u_o \le |\mathcal{O}| \qquad \forall o\in\mathcal{O}. \tag{10}$$

Constraints (2) and (3) ensure that each job has exactly one predecessor and at most one successor. Constraint (4) preserves driver continuity while allowing a route to terminate. The boundary and queue-precedence constraints are given by (5) and (6), respectively, while (7) links the routing and waiting-time variables. Constraint (8) eliminates directed cycles, including zero-time cycles. The ILP of APDP-P consists of the same objective and constraints except for (6).

E. Applications

As a static and deterministic model, APDP is applied to dynamic decision-making scenarios, such as ride-sharing and meal delivery, through a rolling-horizon approach based on batching and reoptimization, e.g., Lobel and Martin (2024), Bertsimas et al. (2019), Yan et al. (2024), Yildiz and Savelsbergh (2019), Ashlagi et al. (2022). That is, jobs submitted within a period of time are collected and executed, and this process is repeated as time progresses. The proof of NP-hardness for the static APDP also provides insight into the complexity of algorithms based on the rolling-horizon approach, thereby avoiding the exploration of exact solutions for the subproblems within each time slot.

The applicability of APDP extends beyond the real-time scenarios in the platform economy, such as meal delivery and ride-sharing, to other logistics problems characterized by aggregated requests. In medical logistics, for instance, biological samples or inter-hospital patient transfer requests, collected within time

windows, are aggregated into an order pool at a fixed number of hospital departments and optimized offline after a collection deadline. Similarly, in the logistics for perishable goods, a fixed number of regional collection centers aggregate large quantities of independently trackable fresh produce from various sources. In these applications, the objective is driven by the urgency of patient transfers, the time-sensitive degradation of sample viability, and the direct correlation between product value and freshness. Crucially, an increase in the volume of requests does not expand the map, as the number of hospital departments and collection centers remains fixed. Therefore, APDP is applicable to a wide range of logistics problems featuring request aggregation at a fixed number of locations and driven by a time-sensitive objective.

F. Proof Idea and Technical Notes on the Reduction

This section only interprets and illustrates the formal construction and equivalence proof in Subsections 4.1–4.2.

Main-zone choice. Figure 4 shows the eight-edge main cycle, the outer job flows, and the two routes from H and G to B or E. Initially, $\mathcal{H}_{i,1}$ reaches H at $t_{i,1} - 2l$, while each new $\mathcal{G}_{i,j}$ reaches G at $\tau_{i,j} - 2l$. In every period, the outer U-blocks force exactly b H-side and $(q-1)b$ G-side drivers toward B, with the complementary drivers going to E. Thus the combinatorial decision is which complete surviving early-clock H-groups constitute the required mass b.

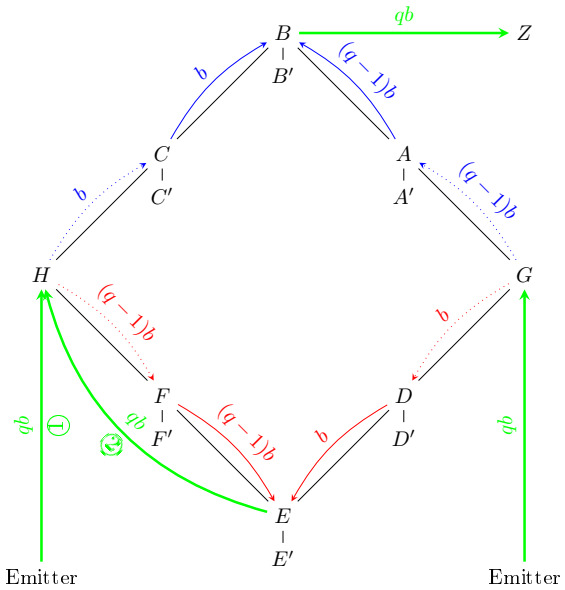


Figure 4 The main zone. Solid arrows are job flows. Dashed arrows show how H- and G-side driver groups split toward B and E.

Why L-service records the choice. At B, $O_{B,i,j} \mid P_{B,i,j} = BZ * (a_i - 1) \mid BB' * M_9 a_i \mid BZ$. If all a_i drivers arrive at $t_{i,j}$, they L-serve this block and $P^\circ_{i,j}$ starts at $t_{i,j} + 1$. If even one arrives only at $\tau_{i,j}$, the long suffix starts at $\tau_{i,j} + 1$. The difference is exactly $M_9 M_{10} a_i$. The outer U-blocks permit only mass b to realize this saving in each period, so the target requires complete surviving early-clock H-groups of total weight b. Figure 5 illustrates the service pattern without adding notation to the formal proof.

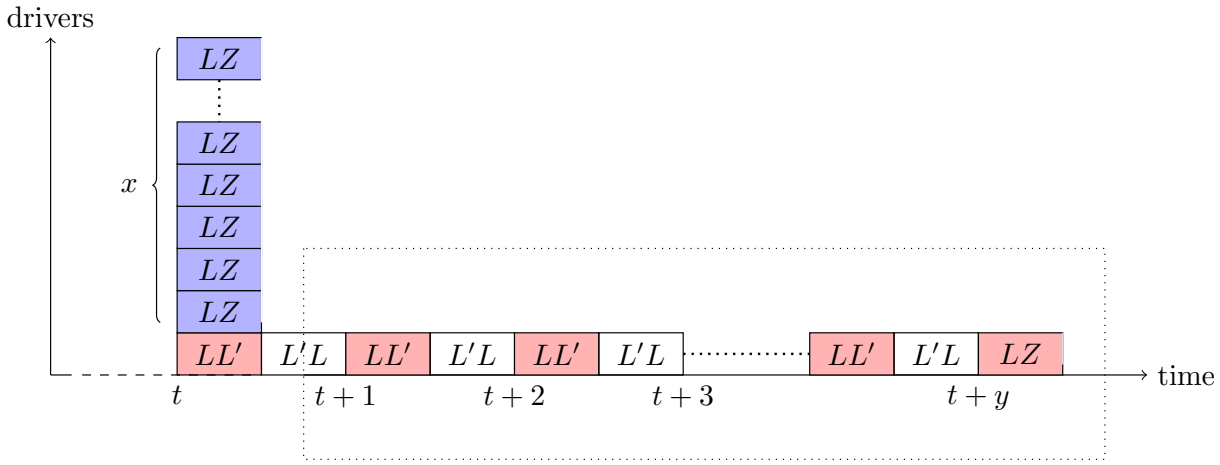


Figure 5 L-service by s drivers. The figure's labels correspond to $x = s - 1$, $y = h$, and $t = u$ in the formal definition.

Feedback across periods. For $i \in \mathcal{I}_j$, all a_i surviving early-clock drivers in $\mathcal{H}_{i,j}$ follow HCB and are committed to Z, while $\mathcal{G}_{i,j}$ returns through $GDEH$ on the late clock. Otherwise the E-side return may mix H/G drivers and hence have early- and late-clock subbatches. Each returning subbatch keeps its clock, and together they form $\mathcal{H}_{i,j+1}$. An index selected once therefore loses its only early-clock group and cannot be selected early again. Repeating this argument over q periods extracts disjoint sets of weight b, hence a 3-partition.

Gadgets and scales. M_0 and the Z-jobs make a committed driver unavailable. M_1 separates emitters from the main zone. The terminal M_2 block forces emitter NQF. $V \mid W$ at scales (M_5, M_4) is the period blocker. $Q \mid R$ at (M_8, M_7) is the cell blocker. (M_9, M_{10}) turns an early $P^\circ_{i,j}$ into the exact saving $M_9 M_{10} a_i$. Together these devices make a complete surviving early-clock H-group necessary for an early P° saving and rule out cross-cell or cross-period reassignment, mistimed emissions, and repeated early savings. The first panel of Fig. 6 depicts the emitter epochs, and the last two show the period and cell time scales.

Forcing hierarchy. The reverse implication removes freedom in decreasing scale order. The terminal emitter block fixes emissions. The W-windows fix period routing and the $\mathcal{K}$-batch L-services. The R-windows fix each cell and the $\mathcal{F}$-batch L-services. The P° identity extracts the partition. Both blocker layers use the same two inequalities. A premature kth tail lacks $s - k + 1 - \eta$ drivers, and a delay δ loses at least $h\delta - C$, which dominates every residual saving at the next smaller scale. The construction consequently

relies on both a growing driver count and widely separated length scales, motivating the tractable cases studied in Section 5.

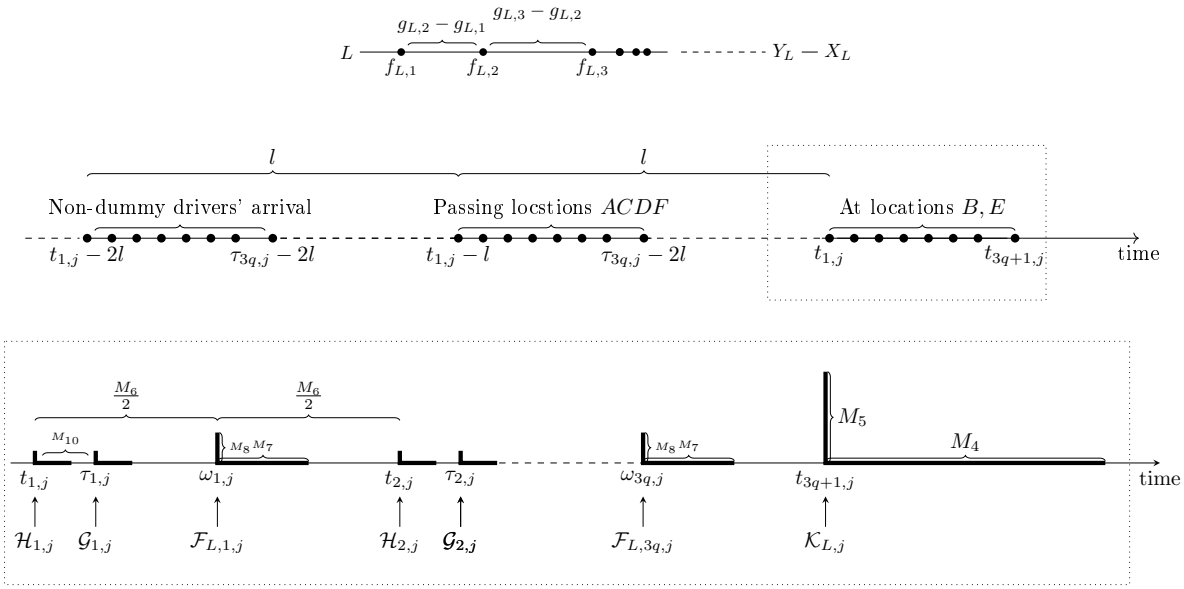


Figure 6 Schematics of emitter epochs, a canonical period, and one cell. Thick segments denote L-services.

G. Proof Idea and Technical Notes for Theorem 4

This section only interprets the formal proof for Theorem 4.

Location splitting. The two clones separate the roles of pickup and delivery without changing geometry. Jobs leave an "out" node and enter an "in" node, while the zero-length move between co-located clones restores the original location before the next pickup. Contracting the clones therefore recovers precisely the original routes and start times.

Equivalent-edge state compression. A route on the bipartite graph alternates from an origin to a customer and then to the next origin. Collapsing these two legs confines every driver's event-state destination to the fixed set $\mathcal{R}$. Under scaled lengths, both destinations and residual times come from a fixed number of categories. Driver identities and histories do not affect future feasible continuations, so only the number of drivers in each category is retained. The accumulated objective is a label, not a state coordinate. Among equal $(t, \vec{\boldsymbol{q}}, \vec{\boldsymbol{d}})$, every larger WT is dominated.

Total distance. When waiting is free, dispatches may be delayed until earlier moves finish. Serializing the moves changes no traveled edge and respects every queue precedence, but removes all residual-time coordinates. The DP consequently records only the unserved suffix of each origin queue, the number of drivers at each origin, and the best distance already incurred.

The PARTITION gate. The two A-job copies encode a small coefficient a_i and a separated coefficient ca_i on top of the common baseline X. The four Z-jobs absorb four drivers permanently. Before T_A only two drivers exist, so they can expose an M-block only by making themselves unavailable. At T_A, the three arriving dummy drivers raise the count to five, exactly enough to start all four Z-jobs while one driver clears both co-located zero-time blocks. Their multiplicity makes a one-unit delay dominate every A- and Z-job saving.

Decoding the partition. The baseline X first forces equal A-job counts. After that term cancels, the coefficient c forces the R_2-weights to split equally. The remaining R_1 equality follows automatically. Thus the two numerical scales perform the two lexicographic comparisons that the expanded algebra obscures.

Encoding convention. A repeated block $J * M$ is run-length encoded. Hence c, X, M, T_A, Γ, and the distance to Z all have polynomial binary length.

References

Agnetis A, Cosmi M, Nicosia G, Pacifici A (2023) Two is better than one? order aggregation in a meal delivery scheduling problem. *Computers & Industrial Engineering* 183:109514, URL http://dx.doi.org/10.1016/j.cie.2023.109514.

Ashlagi I, Burq M, Dutta C, Jaillet P, Saberi A, Sholley C (2022) Edge-weighted online windowed matching. *Mathematics of Operations Research* 48(2):999–1016, URL http://dx.doi.org/10.1287/moor.2022.1289.

Auad R, Erera A, Savelsbergh M (2023) Dynamic courier capacity acquisition in rapid delivery systems: A deep Q-learning approach. *Transportation Science* 58(1):67–93.

Berbeglia G, Cordeau JF, Laporte G (2012a) A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing* 24(3):343–355.

Berbeglia G, Pesant G, Rousseau LM (2012b) Feasibility of the pickup and delivery problem with fixed partial routes: a complexity analysis. *Transportation Science* 46(3):359–373.

Bertsimas D, Jaillet P, Martin S (2019) Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research* 67(1):143–162.

Cherkesly M, Desaulniers G, Irnich S, Laporte G (2016) Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. *European Journal of Operational Research* 250(3):782–793.

Cygan M, Fomin FV, Kowalik Ł, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) *Parameterized algorithms* (Springer).

de Paepe WE, Lenstra JK, Sgall J, Sitters RA, Stougie L (2004) Computer-aided complexity classification of dial-a-ride problems. *INFORMS Journal on Computing* 16(2):120–132.

Desaulniers G, Errico F, Irnich S, Schneider M (2016) Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research* 64(6):1388–1405.

DiDi Open Platform (2026) Ride-request queueing. https://open.es.xiaojukeji.com/doc/openapi/description/lineup.html, in Chinese; accessed July 12, 2026.

Dror M (1994) Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research* 42(5):977–978.

Du J, Leung JY, Young GH (1991) Scheduling chain-structured tasks to minimize makespan and mean flow time. *Information and Computation* 92(2):219–236.

Haddad MN, Martinelli R, Vidal T, Martins S, Ochi LS, Souza MJF, Hartl R (2018) Large neighborhood-based metaheuristic and branch-and-price for the pickup and delivery problem with split loads. *European Journal of Operational Research* 270(3):1014–1027.

Hall LA, Schulz AS, Shmoys DB, Wein J (1997) Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research* 22(3):513–544.

Horn W (1973) Minimizing average flow time with parallel machines. *Operations Research* 21(3):846–847.

Hu TC (1961) Parallel sequencing and assembly line problems. *Operations Research* 9(6):841–848.

Lenstra JK, Rinnooy Kan A (1978) Complexity of scheduling under precedence constraints. *Operations Research* 26(1):22–35.

Lobel I, Martin S (2024) Detours in shared rides. *Management Science* 71(2):1716–1736.

Lyft (2019) Introducing fast match: Making pickups easier at MSY. Lyft News, https://www.lyft.com/blog/posts/making-airport-pickups-easier-for-you-msy, accessed July 12, 2026.

Monma CL (1982) Linear-time algorithms for scheduling on parallel processors. *Operations Research* 30(1):116–124.

Muntz RR, Coffman Jr EG (1970) Preemptive scheduling of real-time tasks on multiprocessor systems. *Journal of the ACM (JACM)* 17(2):324–338.

Spliet R (2023) The complexity of the pricing problem of the set partitioning formulation of vehicle routing problems. *Operations Research* 71(5):1454–1457, URL http://dx.doi.org/10.1287/opre.2023.2481.

Timkovsky VG (2003) Identical parallel machines vs. unit-time shops and preemptions vs. chains in scheduling complexity. *European Journal of Operational Research* 149(2):355–376.

Uber (2019) Enabling seamless airport pickups. Uber Blog, https://www.uber.com/us/en/blog/seamless-airport-pickups/, accessed July 12, 2026.

Ulmer MW, Thomas BW, Campbell AM, Woyak N (2021) The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science* 55(1):75–100.

Wang T, Bellenguez-Morineau O (2019) A complexity analysis of parallel scheduling unit-time jobs with in-tree precedence constraints while minimizing the mean flow time. *Journal of Scheduling* 22(6):709–714.

Yan J, Martin S, Taylor SJ (2024) Trading flexibility for adoption: From dynamic to static walking in ride-sharing. *Management Science* 71(7):5875–5892.

Yildiz B, Savelsbergh M (2019) Provably high-quality solutions for the meal delivery routing problem. *Transportation Science* 53(5):1372–1388.