Real-Time Design of Public Transport Lines: Reconciling Adaptivity and Efficiency

Duo Wang[1], Andrea Araldo[2], and Mounim El-Yacoubi[3]

[1]PhD, Institut Polytechnique de Paris
[2]Associate Professor, Institut Polytechnique de Paris
[3]Professor, Institut Polytechnique de Paris

Short Summary

Demand-responsive transport (DRT) is typically routed by solving Dynamic Vehicle Routing Problems (DVRPs), where individual vehicle trajectories are adjusted on incoming requests. This limits demand consolidation and thus efficiency. On the other hand, Conventional Public Transport (CPT) bus systems are based on a network of lines and users find their routes on it, which provides high demand consolidation. However, such a network is built offline and cannot adapt to the demand. We propose a public transport management strategy that reconciles efficiency and adaptivity by dynamically designing a structured network of lines via a receding-horizon optimization approach. Using real-world trip requests, we show that we nearly double the fraction of served requests compared to DVRP-based routing, and we serve more requests than CPT with lower user trip times.

Keywords: demand-responsive transport; public transport; network design; ride-sharing; dynamic systems

1 Introduction

Conventional public transport (CPT) systems rely on fixed lines and schedules, enabling strong demand consolidation and high efficiency. However, they are designed *offline*, prior to operation, based on nominal conditions and demand. CPT thus lacks flexibility and offers poor service outside dense areas. In contrast, demand-responsive transport (DRT), typically managed via solving Dynamic Vehicle Routing Problems (DVRPs), lacks *structure*: vehicle trajectories are adjusted individually, without an overall master plan. This often prevents the sharing of miles traveled among many users, leading to low occupancy and necessitating excessive subsidies, thereby hampering their sustainability (Currie & Fournier, 2020). The importance of structure to achieve efficiency is illustrated in Fig. 1. The most notable attempts to reconcile efficiency and adaptivity are now briefly discussed. Hybrid systems combine CPT as a backbone and DVRP-based DRT in the last mile (Calabrò et al., 2023; Fielbaum & Alonso-Mora, 2024), but the respective limitations persist in both systems. *Offline* systems improve efficiency by allowing transfers Masson et al. (2014) or by constructing bus lines based on user requests Tong et al. (2017). However, they require knowing the entire demand before computing routes, with prebookings the day

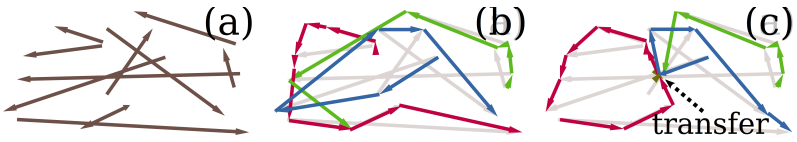


Figure 1: (a) Trip requests (b) DVRP-based routes (c) Structured network of lines

before or longer. Abdelwahed et al. (2023) design lines based on VRP, thereby preventing complex user journeys, which limits efficiency.
This paper proposes a novel way to operate DRT: instead of adjusting individual vehicle trajectories to incoming requests, we design a network of lines, which, as CPT, allows high demand consolidation and complex user journeys including transfers. Different from CPT, in which the network is designed *offline* during strategic and tactical planning, we instead construct it *online*, during its operation, dynamically adjusting it to incoming requests, which do not need to be known in advance.

2 Methodology

Model

We model the network of bus lines as an overlay graph $\mathcal{G}(t) = (\mathcal{E}(t), \mathcal{V}), t \in [0, T]$. $\mathcal{G}(t)$ is embedded onto a substrate graph $\mathcal{G}_{\text{substr}} = (\mathcal{E}_{\text{substr}}, \mathcal{V})$, where $\mathcal{V}$ is the set of candidate stops and $(u, v) \in \mathcal{E}_{\text{substr}} \subseteq \mathcal{V} \times \mathcal{V}$ indicates that a bus is allowed to go directly from stop u to stop v and we denote by $w_{u,v} \geq 0$ the required time.[1] Overlay graph $\mathcal{G}(t)$ is a multi-layer graph, where each layer represents a line $l \in \mathcal{L}$:

$$\mathcal{G}_l(t) = (\mathcal{E}_l(t), \mathcal{V}) \qquad \mathcal{E}(t) = \bigcup_{l \in \mathcal{L}} \mathcal{E}_l(t). \tag{1}$$

We model $\mathcal{G}(t)$ as a *time-expanded graph*, which commonly represents public transport schedules (Fortin et al., 2016, Fig. 2). $\mathcal{G}(t)$ is composed of *time-expanded arcs* $e = (\tau, v, v') \in \mathcal{E}_l(t)$, each indicating that a movement of a bus is scheduled to depart at instant $\tau > t$ to go from node v to v' (where it will arrive at $t + w_{v,v'}$.[2] Unlike static public transport, these lines are not predefined but are progressively constructed during operation, whence their dependence on t. The decision policy to construct them will be detailed in the next subsection.
Requests arrive over time. In particular, request $r = (\tau, v, v')$ arrives at instant τ to go from node v to v'. If a feasible path for r exists in $\mathcal{G}(t), t = \tau$, the request is considered served. A feasible path for r may traverse multiple lines. Within each line, such a path is composed of a sequence of consecutive[3] time-expanded edges. Such a path may also include transfers from a line and another at certain nodes with some non-negative waiting time. A feasible path must be no longer than L time units ($L = 30$ minutes in the numerical results). Note that multiple feasible paths might be available, and we let the user choose among them based on their preferences.[4]
This model is illustrated in Fig. 2.
If no feasible path exists in $\mathcal{G}(t), t = \tau$, request r is placed in a buffer $\mathcal{B}(t), t = \tau$. If subsequent evolution of $\mathcal{G}(t), t > \tau$ yields a feasible path, r is removed from $\mathcal{B}(t)$ and marked as served.
The state of the system at time t is thus $s(t) := (\mathcal{G}(t), \mathcal{B}(t))$ and evolves due to requests arriving and being put in the buffer and to modifications of $\mathcal{G}(t)$, as detailed in the subsection below.

[1] This movement might require traversing several links on the road network.

[2] In the following, we will not specify "substrate" or "time-expanded" for graphs and edges where the context is unambiguous.

[3] "Consecutive" means that the arrival time and node of one edge correspond to the departure time and node of the next. The mathematical definition of feasible path is in our extended report (Wang et al., 2026, (3)-(8)).

[4] For simplicity, when measuring travel times, we consider shortest paths in the numerical results. Note that the service of a user depends on the existence of at least one path and is independent of the user's route choice.

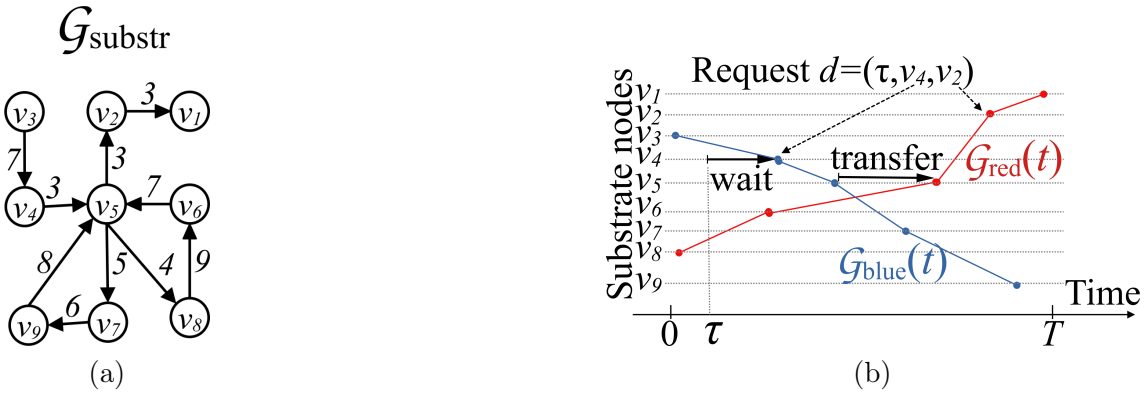


Figure 2: On top of substrate graph $\mathcal{G}_{\text{substr}}$ (a), we design time-expanded graph $\mathcal{G}(t)$ composed of two layers (bus lines). Request τ arrives at instant τ and transfers from the blue to the red layer to get the service.

Network design decisions

Designing the network of bus lines online consists of a sequence of decisions, each determining

- Which edge to add $e = (\tau, v, v')$, i.e., which departure time τ, from which node v to which node v';
- To which layer $\mathcal{G}_l(t), l \in \mathcal{L}$ such an edge must be added, i.e., which bus must perform such movement;
- At which instant $t < \tau$ such an edge must be added. This is the time at which the corresponding bus movement is inserted in the plan (but the actual movement will occur at $\tau > t$).

In our extended report (Wang et al., 2026, Sec. III.D), these decisions are formalized as an Impulsive Control Strategy of a Piecewise Deterministic Markov Process and we analytically prove that, if we aim to maximize the number of served requests, each edge $e = (\tau, v, v')$ must be planned L time units in advance, i.e., it must be added in overlay graph $\mathcal{G}(t)$ exactly at time $t = \tau - L$. We provide here an intuitive explanation. First, observe that for any movement departing at instant τ from node v to v', it is preferable to plan it as soon as possible. Indeed, consider the corresponding time-expanded edge $e = (\tau, v, v')$ and suppose it is added at time t. The only requests that may exploit such an edge in their paths are those arriving in $]t, \tau[$. Therefore, the sooner is the addition time t, the higher is its contribution to serving requests, as this contribution spans a larger time interval. However, there is no reason to anticipate, earlier than $\tau - L$, the insertion in the plan of a bus movement that will occur at instant τ. This is because we only consider as feasible the paths that are no longer than L. Se the formal proof in (Wang et al., 2026, Sec. III.D). We will thus only consider decision strategies where an edge $e = (\tau, v, v')$ is inserted at instant $t = \tau - L$. Establishing this exact timing, allows describing the decision strategy in terms of a Markov Decision Process over a discrete set of time steps $\{t_n\}_{n=1,2,\ldots}$, called intervention times, and with state $s(t) = (\mathcal{G}(t_n), \mathcal{B}(t_n))$. At every intervention time t_n, the action space and the next intervention time is found as follows:

1. For each line $l \in \mathcal{L}$, we calculate its current termination time: suppose that the latest edge of a line l is $e = (\tau, v, v') \in \mathcal{E}_l(t)$, its arrival time $\tau + w_{v,v'}$ corresponds to the termination time of that line.
2. We then find the "shortest line" (and denote with with l^*), i.e., the one with the earliest termination time (which we denote with tt_l^*). This is the line that needs to be extended before the others.

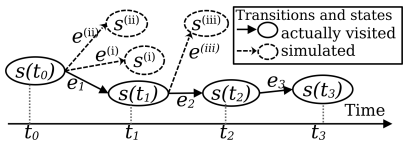


Figure 3: An example of search tree $Tree(s(t_0))$

3. The action space is the set of potential time-expanded edges of layer $\mathcal{G}_l(t)$ that depart at tt_l^*. The edges of such action space are thus of type $e_l^* = (tt_l^*, v_l^*, v_l'^*)$, with $(v_l^*, v_l'^*) \in \mathcal{E}_{\text{substr}}$.

4. The next intervention time is set to $t_{n+1} = tt_l^* - L$.

The instantaneous reward is the number of requests served between t_n and t_{n+1}, including both the requests that immediately find one or more feasible paths upon their arrival and the requests that were waiting in buffer $\mathcal{B}(t)$ and that found a path later, thanks to the insertion of some later time-expanded edge. The objective is to maximize the cumulative reward, i.e., the total number of served requests.
Observe that, by construction, the four steps above ensure that at any instant t, (i) the current network $\mathcal{G}(t)$ already contains the plan of all the movements up to instant $t + L$, and (ii) only modifications occurring after time $t + L$ are allowed. This allows passengers arriving at t (i) to plan their trips and (ii) to have the guarantee that future network design decisions will not disrupt such trips.

Solution method

The online network design decision problem described above has combinatorial complexity, due to the exploding number of possible sequences of edges to add. We use Monte Carlo Tree Search (MCTS) to approximate the optimal policy. Each node in the search tree represents a partial network construction trajectory. From a given state, the algorithm explores possible sequences of actions by simulating future demand realizations (see depiction in Fig. 3). Note that during this exploration, the actual bus line network is not modified, and the sequences of actions are just simulated. During such a simulation, new future requests are simulated as well. [5] In this way, we can account for the future rewards yielded by each action.
The selection of actions (i.e., edges to be added) during tree expansion is guided by an exploration–exploitation trade-off. Let $Q(s, e)$ denote the estimated value of action e in state s, and $N(s, e)$ the number of times it has been explored. Actions are selected according to:

$$e = \arg\max_e \left(Q(e, a) + c \cdot \pi^{\text{aux}}(e|s) \cdot \sqrt{\frac{\log N(s)}{N(s, e)}} \right), \tag{2}$$

where c is a parameter that prioritizes exploration of state-action pairs that have not been explored a lot.
The role of auxiliary policy $\pi^{\text{aux}}(e|s)$ in the formula above is to further improve efficiency. Indeed, without it, we noticed that exploration was too dispersed, i.e., we were wasting simulation time trying unpromising network trajectories. To reduce this issue, we first ran our approach with a simplified version of (2) (setting $\pi^{\text{aux}}(e|s) = 1, \forall e, s$). We dedicated many iterations (and thus much computation time) to explore possible network evolution

[5] To keep our evaluation conservative, we avoided using advanced demand prediction models. In the numerical results, we instead adopted a simple predictor, consisting of a simple statistical model, calibrated on real taxi data, to capture the observed request arrival rate and geographical distribution of origins and destinations. This is detailed in our extended report (Wang et al., 2026, App. III.A). Results are already good with this simple predictor, and even better results could be expected with more advanced forecast models.

Parameter	Value
Num. of candidate stops	67
Fleet size FS	5, 10, 20, 30, 40, 50; Default (if not otherwise stated): 40
Speed of a bus v_{bus}	17.3 km/h Trottenberg (2018)
Private car speed v_{car}	11.4 km/h Trottenberg (2018)
Walking speed v_{walk}	4.3 km/h Google (2024)
Rollout termination number N_{end}	5
Look-ahead buffer B	30 mins
Training set to train π^{aux}	All request data from NYC (2024a), February'24
Tested scenario	20% of request data from NYC (2024a), 9:00-13:00 March 1, 2024

Table 1: Scenario parameters

trajectories. We then let the simplified version of our approach extract the promising trajectories. We collected hundreds of such exemplary trajectories in a training set. We then calibrated $\pi^{\text{aux}}(e|s)$ as the likelihood of observing action e when in state s in the exemplary trajectories. After this preliminary calibration, we achieve that $\pi^{\text{aux}}(e|s)$ gives higher weight to the best transitions and can be used in formula (2) to bias exploration toward the promising state-action pairs. Therefore, the introduction of $\pi^{\text{aux}}(e|s)$ constitutes an overhead that must be borne just once, for its calibration. After that, its use within (2) is instantaneous. Full details of our solution method are provided in our extended version (Wang et al., 2026, Sec. IV).

3 Results and discussion

Considered scenario

We evaluate our method against real user requests in Manhattan (Tab. 1). We replay a randomly selected 20% subset of requests from public dataset NYC (2024a). We use the centroids of the 67 taxi zones as candidate pickup/dropoff stops. In substrate graph $\mathcal{G}_{\text{substr}}$, set $\mathcal{V}$ contains such centroids, each of which is connected to all the others. Weight $w_{u,v}$ of edge $(u, v) \in \mathcal{E}_{\text{substr}}$ is the time for a bus to travel between nodes, i.e., $w_{u,v} = d(u, v)/v_{\text{bus}}$, where $d_{u,v}$ is the distance between $u, v \in \mathcal{V}$ and v_{bus} is the average speed of the bus. The position of set $\mathcal{L}$ of FS buses is randomly initialized.

To train auxiliary policy π_t^{aux}, we first construct a dataset as follows. We randomly sample 100 instances (100 days) of February 2024 from the dataset. For each sampled instance, we run a vanilla MCTS (with classical UCT scoring) between 9am: 1pm and record the resulting solution trajectory, each being a sequence of TEGs. We thus have 100 exemplary trajectories. At each decision step, we record the corresponding transition, i.e., which next stop of a certain bus is added to the trajectory. We also add into the input (i) the ID of the stop where that bus is located before taking the action, (ii) the travel times associated with all feasible actions (all the other stops), and (iii) the cosine of the angle between each candidate action vector and the previous action of the same bus (we do this to let the policy learn to preserve a certain directionality of vehicle movements and avoid zig-zags). In this way, we extract movement patterns of good quality decisions from the exemplary trajectories, such as preferring next stops that are neither too close nor too far, and avoiding unnecessary detours and backtracking.

We evaluate whether the proposed method yields a high **service rate** (more served requests) while using few resources (small **fleet size**), and providing a good user experience, by minimizing **trip time** (average **waiting time**+in-vehicle time). Since our service lays in the middle between dynamic taxis and and a conventional bus network, we compare it against

Table 2: Comparison with DVRP and CPT. Hyperparameters are in *italic*.

Dynamic Vehicle Routing Problem (DVRP)

	Max wait time (min)	Serv. Rate	Trip time (min)	Wait (min)
SOTA	*20*	34%	23	4
	30	50%	25	5
Our method	*20*	63%	28	9
	30	90%	31	14

Conventional Public Transp. (CPT)

	Lines	Bus/ line	Serv. rate	Trip time (min)	Wait (min)
SOTA	*4*	10	17%	10	1
	10	4	50%	17	5
	20	2	73%	32	16
	40	1	80%	56	41
Our method			91%	31	14

Fleet size	Avg. Occupancy (requests/bus)	Avg. Stretch w.r.t direct car trip	Avg. Stretch w.r.t walking trip	Averrage number of transfers
5	24	1.80	0.68	1.24
10	14	1.56	0.61	1.79
20	9	1.90	0.72	1.85
30	7	2.35	0.89	1.59
40	6	2.62	0.99	1.68
50	5	2.83	1.07	1.50

Table 3: Performance with different fleet sizes

- A State-Of-The-Art (SOTA) mobility on demand service management, based on the Dynamic Vehicle Routing Problem (DVRP), representing an optimized dynamic ride-haling service.
- A SOTA conventional bus network, designed offline.
- The performance of the real taxi service in Manhattan.

Comparison with State-Of-The-Art (SOTA) Dynamic Vehicle Routing Problem (DVRP) solutions.

In Table 2-left, we compare our solution with a well-known SOTA DVRP solution Alonso-Mora et al. (2017), of which we run an open source implementation MetaZuo (2017), with two different values of hyperparameter "maximum waiting time", fleet size of 40 vehicles for both solutions and infinite seat capacity.[6]
Observe that the goal of SOTA Alonso-Mora et al. (2017) is to optimize an on-demand "high-capacity" ride-hailing. However, their DVRP-based approach limits the actual capacity they can sustain, whereas, via our network-centric approach, we nearly double the service rate, thanks to the efficiency brought by the network structure we design online. The conceptual reasons of such gains, explained in §1, are confirmed by these results.[7] On the other hand, with our proposed system, users experience an increase in waiting and trip time and also need to deal with transfers. This is however likely acceptable, since it is similar to what they currently experience in Conventional Public Transport (CPT).[8]

[6]We then verify that the actual vehicle occupancy achieved by our solution is compatible with buses and minibuses (Table 3).

[7]Observe that both the SOTA DVRP solution and our method are applied on the same demand and substrate network $\mathcal{G}_{\text{substr}}$. What changes between the two is the way vehicle movement is scheduled on top of $\mathcal{G}_{\text{substr}}$: the former modifies vehicle routes at every request, while we design a "masterplan" time-expanded network $\mathcal{G}(t)$ (overlay on top of $\mathcal{G}_{\text{substr}}$), which vehicles will follow.

[8] We do not aim to replace taxis, which would still be preferred by users willing to pay a high fee for maximum comfort; our system is rather a complement to CPT for everyday mobility. In other words, our method enables a brand-new transport service, rather than a "better" taxi service. Such a new service sits in the middle between the efficiency and high capacity of CPT and the adaptivity and comfort of taxis.

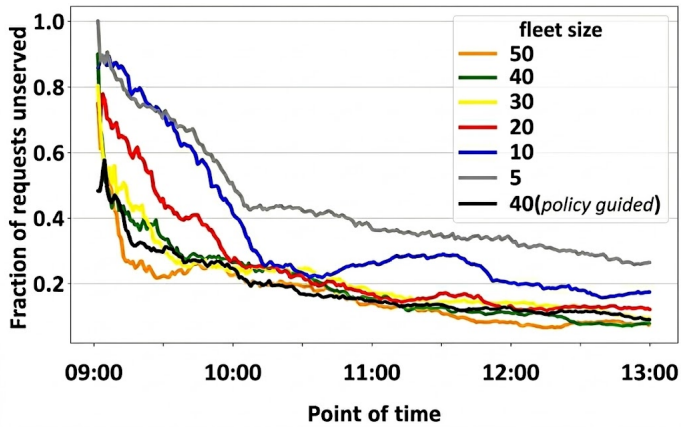


Figure 4: Fraction of requests unserved from 9:00 to 13:00.

Comparison with SOTA conventional bus network design

A key limitation of conventional public transport design is the need to fix the network structure a priori. In practice, this translates into selecting a set of lines, which determines the level of service coverage and operational cost.
The method of Fielbaum & Alonso-Mora (2024) represents conventional public transport (CPT), under the same setting (Table 1), and the same fleet size= 40. Table 2-right shows that by increasing the number of lines, the static bus system covers more areas and improves the service rate. However, some areas remain consistently unreachable, resulting in 20% unserved demand. Moreover, as the number of lines increases, trip and waiting times increase because fewer vehicles operate on each line. In contrast, our real-time design offers much higher service rates without excessively degrading trip and waiting times, due to its capability to adapt to the actual demand, rather than just operating lines previously planned offline.

Comparison with real taxis

From NYC (2024b) we measure that taxi real trips take in real-world $\sim$ 13 minutes on average between 9am and 1pm, which are shorter than the ones our method provides (Table 2). However, we use only 40 buses, versus 2600 taxis and serve 90% of the demand.[9] The loss of user-centric performance is thus justified by the much greater efficiency, which is compatible with the nature of the proposed system (see footnote 8).

Service characteristics

Fig. 4 shows that the neural policy guidance is beneficial at the beginning, as it suggests directly good decisions, thus reducing from 40% to 30% unserved requests 30 minutes after startup, with fleet size $FS = 40$. Fig. 5 shows that increasing fleet size allows serving longer trips, which mechanically results in a higher average trip time (Table 4).
The in-vehicle time distribution (Fig. 6) provides further insight into system performance. For short and medium distances, travel times remain close to those observed in taxi operations. For longer distances, travel times increase moderately due to transfers but remain significantly lower than those observed in static PT.
This result shows that the efficiency gains obtained through consolidation do not come at the expense of excessive detours. Instead, the system achieves a favorable balance between efficiency and user experience.
Fig. 5 also shows that short trips are predominantly served without transfers. To adapt

[9] Around 13k taxis operate in Manhattan NYC (2024b) and 2.6k correspond to the 20% of 13k (remember that we considered 20% of the real demand Tab. 1).

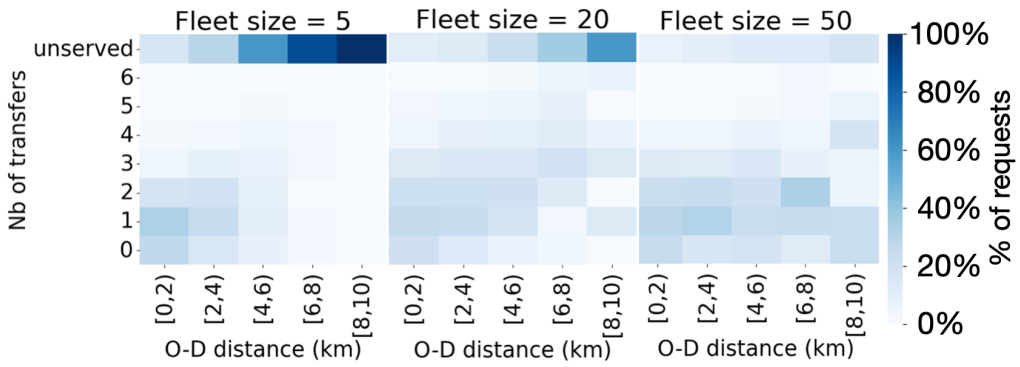


Figure 5: Fraction of requests experiencing certain number of transfers over total requests, per each O-D distance interval.

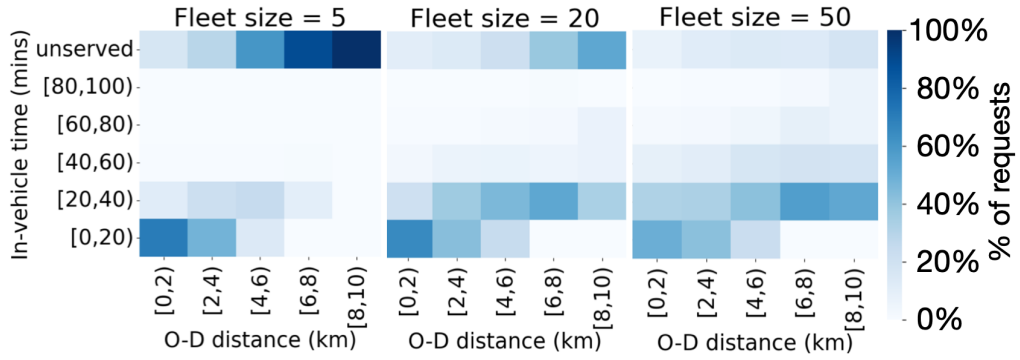


Figure 6: Distribution of in-vehicle time by origin–destination distance

our approach to the case in which users accept no more than 2 transfers, we can label as infeasible any more complex path. In our current system, this would still preserve 88% of completed requests.

Real-time computation

We pinpoint that our method meets the real-time requirements, since the choice of each action is made within seconds, i.e., much faster than the time between two consecutive intervention instants

Discussion on the performance

Table 5 summarizes the main performance indicators of our method againts the SOTA. In summary, the proposed method achieves the best compromise between efficiency and service quality. It offers a high service rate with a small fleet size without excessively degrading trip and waiting times.
The results confirm that the performance gains stem from structural properties of the system. First, DVRP is fundamentally limited by its local optimization structure, constrained by the trajectory-based formulation. Second, static PT achieves consolidation but lacks adaptivity, which yields poor results under high demand variability.
The proposed approach combines both advantages by dynamically constructing a structured network. This enables demand consolidation while adapting to demand in real time. Transfers play a central role in this mechanism. As shown in previous work, allowing transfers can significantly reduce detours and improve system efficiency.
Overall, the proposed paradigm resolves the classical trade-off between flexibility and efficiency in mobility systems.

Fleet size	Service rate	Avg. Trip time (mins)	Avg. Waiting time (mins)
5	73.58%	27.81	16.01
10	82.55%	27.75	15.76
20	87.82%	29.68	15.03
30	90.25%	29.93	14.20
40	91.83%	31.79	14.63
50	92.31%	32.20	13.89

Table 4: Comparison of results with different fleet sizes

Method	Fleet size	Service rate	Trip time	Wait time
DVRP (SOTA)	40	50%	25	5
Static PT (SOTA)	40	80%	56	41
Taxi (real Manhattan traces)	2600	100%	18	2
Proposed	40	91%	31	14

Table 5: Comparison of global performance indicators

4 Conclusions

This paper introduced a new paradigm for flexible mobility: the real-time design of public transport lines. By shifting from vehicle-level routing to network-level design, the proposed approach enables demand consolidation while retaining adaptivity.
Empirical results on real-world data show that the method significantly outperforms both DVRP-based approaches and static public transport design. These findings suggest that future mobility systems should move beyond purely demand-responsive routing and instead leverage adaptive network structures.

References

Abdelwahed, A., van den Berg, P. L., Brandt, T., & Ketter, W. (2023). Balancing convenience and sustainability in public transport through dynamic transit bus networks. *Transportation Research Part C: Emerging Technologies*, *151*, 104100.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. Nat. Acad. Sci.*.

Calabrò, G., Araldo, A., Oh, S., Seshadri, R., Inturri, G., & Ben-Akiva, M. (2023). Adaptive transit design: Optimizing fixed and demand responsive multi-modal transportation via continuous approximation. *Transportation Research Part A: Policy and Practice*, *171*.

Currie, G., & Fournier, N. (2020). Why most drt/micro-transits fail–what the survivors tell us about progress. *Res. Transp. Econ.*.

Fielbaum, A., & Alonso-Mora, J. (2024). Design of mixed fixed-flexible bus public transport networks by tracking the paths of on-demand vehicles. *Transp. Res. Part C*.

Fortin, P., Morency, C., & Trépanier, M. (2016). Innovative GTFS Data Application for Transit Network Analysis Using a Graph-Oriented Method. *J. Pub.Transp.*.

Google. (2024). *Google maps.* [link].

Masson, R., Lehuédé, F., & Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, *41*, 12–23.

MetaZuo. (2017). *Ride sharing.* [link].

NYC. (2024a). *Tlc trip record data.* [link].

NYC. (2024b). *Yellow cab.* [link].

Tong, L. C., Zhou, L., Liu, J., & Zhou, X. (2017). Customized bus service design for jointly optimizing passenger-to-vehicle assignment and vehicle routing. *Transportation Research Part C: Emerging Technologies*, *85*, 451–475.

Trottenberg, P. (2018). *New york city mobility report.* [link].

Wang, D., Araldo, A., & El Yacoubi, M. (2026). Online design of dynamic networks - extended version. Retrieved from `https://arxiv.org/abs/2410.08875`