CIRRELT

CIRRELT-2024-19

Online Stochastic Optimization for Real-Time Transfer Synchronization in Public Transportation Networks

Laura Kolcheva
Antoine Legrain
Martin Trépanier

June 2024

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Online Stochastic Optimization for Real-Time Transfer Synchronization in Public Transportation Networks

Laura Kolcheva*, Antoine Legrain, Martin Trépanier

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematical and Industrial Engineering, Polytechnique Montréal

Abstract. Transfer speed and protection are critical factors that influence passengers' willingness to use public transportation. Due to unpredictable traffic patterns, fixed transfer schedules may not always align. This work proposes two online stochastic optimization algorithms for the transfer synchronization problem using the hold, skip-stop, and speedup tactics. First, we design an offline arc-flow model using time-expanded graphs to enumerate all possible tactics. The model minimizes total passenger travel time by reducing transfer times. Then we implement two online stochastic optimization algorithms based on the offline model in a discrete-event dynamic environment. Decisions are made based on predictions of the future state of the bus network using sampling to generate scenarios from historical and real-time data. The performance of the algorithms is compared using a real dataset from the public transit system of Laval, Canada. The results show significant improvements in both the number of successful transfers and total passenger travel time across 29 bus lines. This supports the practicality of using online stochastic optimization algorithms to solve the real-time transfer synchronization problem in real-world transportation systems.

Keywords: transit network, bus hold, transfer synchronization, real-time control, online stochastic optimization.

Acknowledgements. The authors would like to thank the Société de transport de Laval, in particular Vincent DIONNE and Grzegorz WIELINSKI, for providing data and valuable feedback. Funding was provided by the Chair in Transportation Transformation, the Fonds de Recherche du Québec - Nature et Technologie (FRQNT-#323365), the Natural Science and Engineering Research Council of Canada (NSERC), the Observatoire international sur les impacts sociétaux de l'IA et du numérique (OBVIA) and Polytechnique Montréal. Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: laura.kolcheva@polymtl.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2024

1. Introduction

Public transit (PT) systems are becoming increasingly important in the context of urban growth, traffic congestion, and sustainable development. PT networks are designed in multiple phases, including planning, operation, and control. Implementing changes in network design or operating systems is often costly and difficult. However, innovative control strategies can provide a cost-effective solution to enhance the performance of PT networks.

Research indicates that transfer speed and protection are crucial factors that affect passengers' willingness to use PT (Ceder et al. (2013)). Although transfers are usually synchronized during the planning process, buses operate in a stochastic environment and may deviate from their schedules, leading to missed transfers and a decrease in ridership. Therefore, transfer synchronization strategies for PT systems, particularly in real-time, are of increasing interest. The availability of real-time passenger and vehicle data is on the rise. By using demand data from smart cards and bus occupancy sensors, planned transfer data from travel apps, and vehicle locations from GPS, we can significantly improve our understanding of the state of PT networks. Dynamic predictions offer insight into the impact that real-time control can have on users throughout all segments of their trips.

This research proposes an arc-flow formulation for the real-time transfer synchronization problem in a dense urban network. Three control tactics are implemented alone or simultaneously in order to synchronize transfers and minimize passenger travel times. The *hold* tactic makes a bus wait at a stop after all passengers have boarded or alighted the vehicle. To *skip-stop* one or more consecutive stops can help reduce bus travel times or make up for schedule delays. Additionally, the *speedup* tactic can decrease bus travel times, help catch up with schedule delays, and prevent missing synchronized transfers. These tactics are enumerated in the arc-flow formulation where variables represent passenger flows in the network. The offline model is tested on instances with many transfer stops from a real large-scale dataset from the city of Laval. We implement two online stochastic optimization (OSO) algorithms using the offline arc-flow formulation in a stochastic, event-based dynamic environment with a rolling control horizon. Predictions of future states of the PT system are integrated using both real-time data and historical data made available by the "Société de Transport de Laval" (STL). To the best of our knowledge, we make the following contributions:

- Designing an arc-flow formulation that enumerates all control tactics for the real-time PT transfer synchronization problem.
- Implementing online stochastic optimization algorithms for the transfer synchronization problem for the first time and comparing their performances.
- Testing on 29 lines from a dense PT network with real-time smart card and GPS data. In real-time, solving large instances that contain entire bus lines and multiple transfer stops while considering all stops as control points.

2. Literature Review

Transfer synchronization has been discussed in multiple literature reviews on PT (Ibarra-Rojas et al. (2015), Liu et al. (2021), Gkiotsalitis and Cats (2021)). Gkiotsalitis et al. (2022) present a review of transfer synchronization during the real-time control phase, which includes at-stop control such as *hold* or *skip-stop*, inter-stop control such as *speedup*, or other types of control such as adding vehicles (Eberlein et al. (1999)).

2.1. *Hold* Tactic

The bus *hold* tactic is to make the vehicle wait at a specific stop for a predetermined amount of time. This can result in reduced operating speed and increased travel or waiting times for passengers. The literature suggests that the *hold* tactic, either alone or in combination with other tactics, is the most effective and easiest to implement (Ibarra-Rojas et al. (2015)). The impact of *hold* can be further improved by using many control stops while considering future predictions. Cats et al. (2012) demonstrate that if all stops are control points, then the propagation of disruptions is limited by control measures distributed throughout the entire line. Furthermore, Berrebi et al. (2018) compare various *hold* strategies for maintaining headway stability, both from the literature and from the industry, using historical AVL (Automatic Vehicle Location) and APC (Automatic Passenger Counter) data. The study demonstrates that predictive methods can stabilize headways, but at the cost of increased *hold* times. Additionally, the study found that the performance of predictive methods improves as prediction accuracy increases.

The *hold* tactic can be used to ensure successful planned transfers. Ting and Schonfeld (2007) apply the *hold* tactic using real-time bus location data to ensure that transfers synchronized at the planning phase take place. The method is tested on an artificial network with three transfer points. Chung et al. (2020) also apply *hold* times to protect planned transfers and wait for delayed buses. The authors take into account the stochastic nature of travel times and model stochastic passenger arrivals. The model is tested using two transfer points. Finally, Daganzo and Anderson (2016) propose a dynamic model for the *hold* tactic to synchronize transfers in a multimodal transportation network. The approach is tested using simulations on four bus instances with a total of fifteen transferring passengers.

Recent research has focused on the use of real-time passenger data, but testing instances have remained small in size. In a study by Kieu et al. (2016), transfers between a main line and a feeder line are synchronized in real-time at a single stop using dynamic predictions of passenger transfer demand. The study uses historical data on passenger origin/destination pairs and real-time information on smart card usage. Gavriilidou and Cats (2019) test two *hold* tactic strategies on a high-frequency bus line to reconcile service regularity with transfer synchronization. Real-time passenger demand is integrated using bus occupancy or smart card validation data from the tramway network of The Hague. The tests are conducted using BusMezzo, a traffic simulation model, on two lines with two transfer stops and a total of five control stops.

2.2. *Skip-stop* and Combined Tactics

The literature also explores the use of real-time *skip-stop* tactic. While this tactic can reduce travel times, it may inconvenience passengers who wish to board or alight at skipped stops, particularly for low-frequency bus lines. Hadas and Ceder (2008) introduce the concept of transfer segments, which are consecutive stops where two vehicles can exchange passengers. The authors present a dynamic programming model that minimizes total passenger travel time by applying *hold* and *skip-stop* tactics. They test it on an artificial network with three bus lines and fourteen transfer segments. Later, Ceder et al. (2013) develop a mixed-integer program running in exponential time using the *hold* and *skip-stop* tactics for the transfer synchronization problem. The model is tested on an artificial network consisting of three bus lines and fourteen transfer segments. Passenger and transfer demands are considered known and are deterministic.

In recent years, an increasing number of articles uses simulation to address the transfer synchronization problem. Guevara and Donoso (2014) combine the *skip-stop* and *hold* tactics in a rule-based method with polynomial running time. They use microsimulation to evaluate the performance of different rules for applying tactics at a single transfer stop with high ridership. They use historical data on passenger demand. Manasra and Toledo (2019) develop an event-based simulation using a rolling prediction horizon. Each re-optimization considers only one prediction of the state of the PT network and includes approximately forty variables. The optimization model minimizes total passenger travel time by combining the *hold* and *speedup* tactics. The running times for a horizon of three stops and three buses average 1.25 seconds. The authors test the model on a network with three lines and three transfer stops.

2.3. Online Stochastic Optimization Algorithms

The literature on stochastic programming initially concentrated on offline or strategic-phase problems due to the high computational costs involved (Shapiro and Philpott (2007), Birge and Louveaux (1997), Kall and Wallace (1994)). Nowadays, it is deemed essential to consider uncertainty to achieve practical and relevant results (Powell (2019)). Uncertainty is often modelled by multiple scenarios of the variables describing future states of the system. This approach provides more robust solutions and accounts for the gradual reveal of new information. Statistical solutions are obtained by generating scenarios on future states as demonstrated in the Sample Average Approximation method by Kleywegt et al. (2002). By solving the offline model for multiple scenarios and computing averages over scenarios, Shapiro (2013) improves the solutions of their online algorithm with predictions from Kleywegt et al. (2002). Algorithm **EXPECTATION (E)**, introduced by Chang et al. (2000), is not well-suited for online applications due to its time-consuming nature and because it distributes optimizations across requests under time-constraints.

As stochastic optimization algorithms that use sampling require larger sample sizes to improve accuracy (Bent and Hentenryck (2004)), some studies have focused on improving the efficiency of such methods. Bent and Van Hentenryck (2004) introduce Algorithm **CONSENSUS (C)** to address the issue of applying

E in real-time. Algorithm **C** chooses the statistically best decision over several scenarios. To improve this work, Bent and Hentenryck (2004) introduce Algorithm **REGRET (R)**, which aims to select a decision that performs reasonably well across all scenarios. Hentenryck and Bent (2009) provide an extensive overview of such methods and their applications. Other studies reduce the sample size by generating more relevant scenarios and considering historical data (Philpott and de Matos (2012), Lin et al. (2013), De Filippo et al. (2019)).

2.4. Research Gap

Many articles in recent literature focus on a single or a small number of transfer points. Passenger demand is often generated using an average arrival rate instead of using real historical data. Additionally, the use of real-time data is typically limited to AVL and APC, with other data sources only being used in offline approaches. The use of smart card data in real-time provides accurate information on passenger demand and enables improved predictions of future demand. However, this information is rarely used in the literature. Finally, the application of OSO algorithms has not been investigated for the transfer synchronization problem.

The rest of this article is organized as follows. Section 3 presents the offline arc-flow formulation for the transfer synchronization problem. Section 4 describes the implementation of OSO algorithms based on the offline model. Section 5 presents the case study used to test the methodology, a sensitivity analysis of the models, and the results of all experiments. Finally, in section 6 we present our conclusion, including a discussion of the results and potential future works.

3. Offline Model

An arc-flow model is formulated for the offline transfer synchronization problem using control tactics. The model minimizes total passenger travel time by improving transfer times, while also constraining deviations from the schedule. This model is a continuation and improvement of previous work by the authors (Kolcheva et al. 2024), refining the mathematical formulation by reducing the number of variables and computational times.

3.1. Arc-flow Formulation

The arc-flow formulation integrates all tactics into time-expanded graphs. The model includes a main line where tactics can be applied and feeder lines that are fixed. A control horizon is considered, ranging from a few stops to the entire main line. Unlike in the literature, all stops are control stops, allowing tactics to be implemented at any stop or between any two stops. This approach enables more efficient control.

The time-expanded graphs in this study use a time-space representation. Each node is associated with a stop and a specific time. Alighting (resp. boarding) passengers are represented using a negative (resp. positive) exogenous flow. Transfers to and from feeder lines are also represented by nodes with exogenous

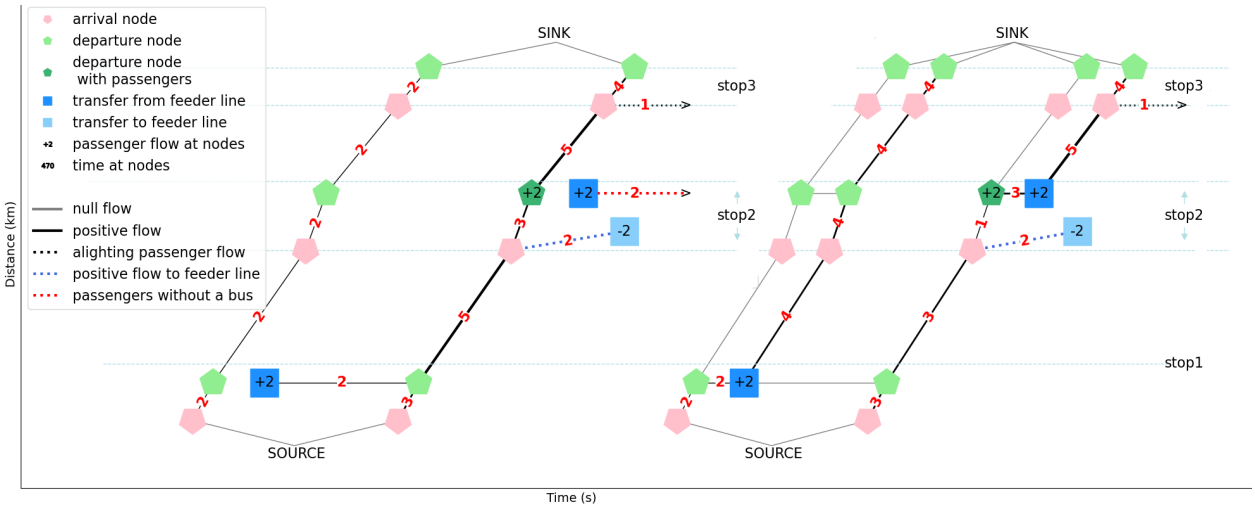

Figure 1 Mock example of a graph construction. Left-no tactics, right-with tactics.

flows and time stamps. To represent waiting times and travel times between stops, arcs are constructed between nodes, with the weight of each arc equal to the time difference between its origin and destination nodes. Passengers who miss their bus will wait at the stop for the next one. The implementation of *hold* tactics is limited to waiting times between existing nodes, represented by arcs in the graphs. This design choice greatly reduces the number of possible values for *hold* times and eliminates the need for additional nodes. If a *speedup* tactic is implemented, an additional 'faster' arc with an earlier node at the next stop will be created, resulting in multiple possible paths for each vehicle. If a *skip-stop* tactic is used, an arc without the possibility for passengers to board and alight the vehicles is added.

Figure 1 illustrates the process of constructing a graph with a mock example. The arcs are labeled with the number of passengers using them, if non-zero. For the sake of clarity, arrival and departure nodes are slightly offset on the y-axis. The optimization model uses the bold black arcs in the solution, while the grey arcs represent all other possible paths. Exogenous flows are shown on the nodes. This mock example includes two vehicles and three stops in the control horizon. In the 'no tactics' case (left), two passengers miss the first bus at 'stop 1' and wait for the following bus. At 'stop 2', a transferring bus arrives too late causing two passengers to miss it. As a result, they are unable to catch another bus within the control horizon (as indicated by the red dotted arrow). In the case with tactics (right), additional arcs are added to create more possible paths for both buses. Some *hold* times are implemented to minimize the total passenger travel time by enabling passengers to catch their buses.

After the time-expanded graphs integrating all possible tactics are constructed, they are used as inputs for a minimum cost flow problem.

3.2. Mathematical Formulation

The following assumptions are made for the offline model. Street conditions allow the implementation of the *speedup* tactic between stops. When faced with multiple choices, passengers choose the fastest route.

Table 3.1 Sets, parameters and variables used in the mathematical formulation.

Sets		
N		set of nodes with s the source node and t the sink node.
N^-		set of nodes (transfer or not) with negative exogenous flows, not including t
A		set of arcs (u,v)
B		set of buses
S		set of stops
A_b^s	$b \in B$, $s \in S$	set of arcs for the bus b between stops s and $s+1$
AN_b^v	$b \in B$, $v \in N^-$	AN_b^v is the set of arcs passengers wishing to alight at node v take to board the bus b. There is one arc per passenger
AM_b^v	$b \in B$, $v \in N^-$	AM_b^u is the set of arcs passengers wishing to alight at node v take to board the bus b, after missing the previous bus. There is one arc per passenger.
Parameters		
c_{uv}	(u,v) $\in A$	passenger flow capacity on arc (u,v)
w_{uv}	(u,v) $\in A$	travel time between nodes u and v.
f_v	$v \in N$	exogenous passenger flow at node v
g_v	$v \in N$	bus departures or bus arrivals at node v ($g_v = 0, \forall v \neq s,t$)
p	$p \geq 0$	additional waiting time perceived by passengers for waiting out of bus compared to waiting in bus.
Variables		
x_{uv}	$(u,v) \in A$	$x_{uv} \in \mathbb{N}$, passenger flow on arc (u,v)
y_{uv}	$(u,v) \in A$	binary variable for the bus flow on arc (u,v)
z_{uv}	$(u,v) \in A$	$z_{uv} \in \mathbb{N}$, indicator variable equal to x_{uv} if y_{uv}=0, and 0 otherwise

Passengers arrive shortly before the scheduled arrival time of buses because they have access to real-time information. Passengers are informed about the use of *skip-stop* tactics prior to the first skipped stop. If passengers are unable to alight due to a *skip-stop* tactic, they can get off at the nearest stop and walk to their intended stop. Passengers who are unable to board a bus due to a skipped stop, as well as those who miss their (transferring) bus wait for the next available bus.

The objective (1) minimizes total passenger travel time as perceived by passengers. The first part represents the total passenger travel time, while the second part describes the additional cost perceived by passengers when waiting for a bus. Constraints (2) define the flow conservation for users. Constraints (3) ensure that if passengers fail to board a bus, then they cannot alight from that bus. These passengers are added to the flows of the bus they manage to board. These constraints also ensure that the solution's flows correspond to the actual origin/destination pairs of passengers. Constraints (4) ensure that each bus takes only one possible path. The source node has a positive exogenous bus flow, while the sink node has a negative exogenous bus flow, both corresponding to the number of buses in the control horizon. Constraints (5) ensure passengers can only travel between stops while aboard a bus. Constraints (6) allow the variables z_{uv} to measure the passenger out-of-bus waiting time. Constraints (7) define the bus flow variables and constraints (8) ensure that passenger flows are non-negative.

Objective

$$\min \quad \sum_{(u,v)\in A} w_{uv}x_{u,v} + \sum_{(u,v)\in A} w_{uv}pz_{u,v} \tag{1}$$

Constraints

$$\sum_{(v,w)\in A} x_{vw} - \sum_{(u,v)\in A} x_{uv} = f_v, \forall v \in N \setminus N^{\text{-}}, v \neq t \tag{2}$$

$$\sum_{(v,w)\in A} x_{vw} - \sum_{(u,v)\in A} x_{uv} = f_v + \sum_{(u,w)\in AN_b^v} (1 - x_{uw}) - \sum_{(u,w)\in AM_b^v} (1 - x_{uw}), \forall b \in B, v \in N^{\text{-}} \tag{3}$$

$$\sum_{(v,w)\in A} y_{vw} - \sum_{(u,v)\in A} y_{uv} = g_v, \forall v \in N \tag{4}$$

$$x_{uv} \leq c_{uv}y_{uv}, \forall b \in B, s \in S, (u,v) \in A_b^s \tag{5}$$

$$x_{uv} - c_{uv}y_{uv} \leq z_{uv}, \forall (u,v) \in A \tag{6}$$

$$y_{uv} \in \{0,1\}, \forall (u,v) \in A \tag{7}$$

$$x_{uv}, z_{uv} \geq 0, \forall (u,v) \in A \tag{8}$$

4. Online stochastic optimization (OSO)

The offline model presented in the previous section does not integrate the dynamic and uncertain nature of a PT network's operations. We therefore formulate a corresponding online problem and implement it in a dynamic environment. We propose two OSO algorithms (**CONSENSUS** and **REGRET**) to solve this problem. We also implement Algorithm **DETERMINISTIC** and a **PERFECT INFORMATION** solution to evaluate the performance of the OSO algorithms.

4.1. Real-time Data and Dynamic Environment

The offline model has information on real and planned travel times and dwell times, on bus schedules, current bus delays, as well as on passenger and transfer demands. In contrast, real-time data only provides bus

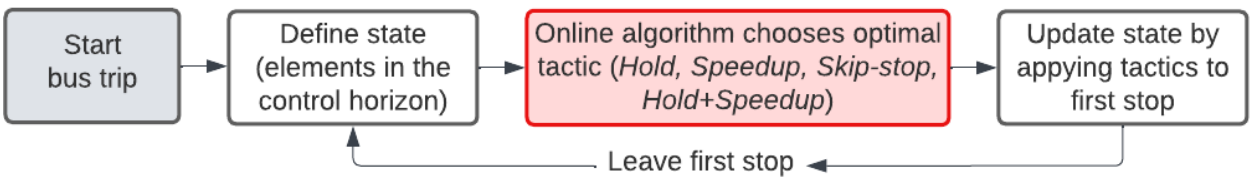

Figure 2 Dynamic environment for the online application of the arc-flow model.

schedules, current bus delays, passenger (and transfer) demand at past stops, as well as current bus occupancy. Information on travel times and passenger flows is not available for stops that have yet to be visited, and this information is gradually revealed each time a stop is reached. Additionally, real-time information on passengers' transfer demand and transfer stop choices is not available. This data is only obtained from AFC (Automatic Fare Collection) systems after passengers have boarded their transfer bus and validated their smart card. Therefore, we create a discrete-event dynamic environment allowing the gradual reveal of real-time data at each stop along a bus line.

The dynamic environment, illustrated in Figure 2, allows the implementation of OSO algorithms. In this environment, the system is re-optimized every time a bus on the main line leaves a stop (called the current stop). During re-optimization, the state S describes the set of all buses (main and feeder lines), stops, and passengers present within a certain control horizon h after the current stop. The first mainline stop in state S after the current stop is denoted s_0. An OSO algorithm is used to make decisions on tactics σ_{s_0} to use at the stop s_0. Tactics include *hold* for a transfer or for the scheduled departure time at s_0, *speedup* between the current stop and s_0, a combination of *hold* and *speedup* tactics, or *skip-stop* s_0. The *speedup* tactic is not allowed when the travel time between two stops is too short (the driver would not have enough time to accelerate and decelerate safely). Note that the computation time for the OSO algorithms needs to be reasonable to apply the *speedup* tactic between the current stop and s_0. Then tactic σ_{s_0} is applied in the real instance. Finally, the position of the bus, the number of passengers in the bus and waiting at stops after the use of the tactic σ_{s_0} are updated. A new re-optimization begins when the vehicle leaves s_0 and s_0 becomes the current stop. All future control tactics are re-evaluated at every re-optimization. The computation times of the OSO algorithms must be low enough to allow real-time implementation.

4.2. Online Stochastic Algorithms: CONSENSUS (C) and REGRET (R)

This section describes two OSO algorithms. Algorithms **C** and **R** both model uncertainty by generating multiple random scenarios using sampling. They are adapted from the literature (Bent and Van Hentenryck (2004)) for the transfer synchronization problem.

Algorithm 1 provides the general framework for the OSO algorithms used in the dynamic environment. The OSO algorithms benefit from the reveal of real-time data every time a new stop is reached. During each re-optimization, the OSO algorithms only have partial knowledge of the state S, limited to the real-time information available at the current stop. Therefore, a number N of scenarios are generated to model travel and dwell times, passenger and transfer demands, and transferring bus arrival times for the stops in S. The scenarios are sampled using available real-time information and historical data. The way scenarios are sampled is specific to every algorithm. For each scenario ω, a graph G^ω incorporating all possible tactics T is built. The offline model defined in section 3.2 solves the problem corresponding to graph G^ω and returns a solution in the form of a sequence of tactics $\boldsymbol{\sigma}^*=(\sigma^*_{s_i})_{i=0}^{h-1}$ to apply at the stops in the control horizon h.

Algorithm 1 Online Stochastic algorithm

function ONLINE_STOCHASTIC_ALGORITHM(S, T, N, ALG)
  score(σ) $\leftarrow 0, \forall \sigma \in T$
  repeat N **times**
    $\omega \leftarrow$ GENERATE_SCENARIO(S, ALG)
    $G^\omega \leftarrow$ BUILD_GRAPH(ω, T)
    $\boldsymbol{\sigma}^* \leftarrow$ SOLVE_OFFLINE_MODEL(G^ω)
    score $\leftarrow$ UPDATE_SCORE(G^ω,$\boldsymbol{\sigma}^*$,ALG, score)
  return $\arg\max_{\sigma \in T}$ score(σ)

Considering this solution, each algorithm uses a specific function to update the scores of tactics in T. The algorithms consider all scenarios and return the tactic with the highest score.

Algorithm **C** evaluates multiple scenarios and then returns the tactic that is optimal most of the time. Whenever a tactic $\sigma^*_{s_0}$ is optimal for stop s_0, its score is increased by one (i.e., the function UPDATE_SCORE returns score($\sigma^*_{s_0}$) $+= 1$). The final score of a tactic is determined by the number of scenarios for which it is optimal. Algorithm **C** then returns the tactic that is used most frequently across all scenarios. However, it does not provide information on other tactics that perform similarly, even if slightly worse. Indeed, it does not measure how well a tactic performs across all scenarios, and it does not recognize that a tactic may never be optimal for any scenario but still be robust overall.

On the other hand, Algorithm **R** aims to select a tactic that performs well in all scenarios. Algorithm 2 describes how it updates the score of all tactics after considering each scenario. In its formulation for the transfer synchronization problem, we evaluate the loss or 'regret' of using sub-optimal tactics at the first stop in the control horizon, while keeping the optimal tactics for the following stops. The 'regret' value in this context represents the additional travel and wait times experienced by passengers due to the change in tactics at the first stop. For each tactic $\sigma \neq \sigma^*_{s_0}$, the 'regret' value of σ for the scenario ω is equal to the increase in the objective function resulting from switching from the optimal sequence of tactics $\boldsymbol{\sigma}^*$ to the sub-optimal sequence $\boldsymbol{\sigma'}$=$(\sigma, \sigma^*_{s_1}, ..., \sigma^*_{s_{h-1}})$. To compute 'regret' values, the current bus in the horizon is restricted to only one possible path corresponding to the tactics in the solution $\boldsymbol{\sigma'}$. This effectively fixes all bus flow variables and constrains passenger flows to a single possible path. The problems solved for each tactic are much smaller compared to the problem containing all possible paths for scenario ω. Furthermore, the same underlying graph G^ω is used to compute all 'regret' values. This allows the evaluation of the performance of all tactics without a significant increase in computation time. Finally, Algorithm **R** returns the tactic with the smallest 'regret' across all scenarios. In this case, we use the negative of the 'regret' values in order to align with algorithm 1, which returns the argument of the maximum score. If a tactic performs very poorly on

some scenario, it will accumulate a significant 'regret' value and become non-competitive. Therefore, tactics returned by Algorithm **R** should not perform badly on any scenario and tactics that perform similarly should have close 'regret' values. Implementing Algorithm **R** is more challenging than implementing Algorithm **C** because it calculates a 'regret' value for all tactics in each scenario, while Algorithm **C** only updates the score of one tactic per scenario.

Algorithm 2 REGRET algorithm

function UPDATE_SCORE(G^ω, $\boldsymbol{\sigma}^*$,**R** , score)
  $obj \leftarrow$GET_OBJECTIVE_VALUE(G^ω,$\boldsymbol{\sigma}^*$)
  for $\sigma \in T \setminus \{\sigma^*_{s_0}\}$ **do**
    $\boldsymbol{\sigma'} \leftarrow (\sigma, \sigma^*_{s_1}, ..., \sigma_{s^*_{h-1}})$
    $obj' \leftarrow$GET_OBJECTIVE_VALUE(G^ω,$\boldsymbol{\sigma'}$)
    score(σ)+ =($obj - obj'$)
  return score

4.3. PERFECT INFORMATION (PI) solution and Algorithm DETERMINISTIC (D)

This section presents two approaches using Algorithm 1 with only one scenario ($N = 1$). Both the **PI** solution and Algorithm **D** are implemented to evaluate the performance of the proposed OSO algorithms. First, the **PI** solution is the implementation of the perfect information offline model in a dynamic environment where decisions on tactics are made with knowledge limited to the stops in the horizon. The single scenario used is the real state S and there is no uncertainty. The optimal tactic $\sigma^*_{s_0}$ found by the offline model for stop s_0 is returned. When only the *hold* tactic is allowed, the offline model can solve large instances (three consecutive buses for a main line with more than 80 control points, and many transferring stops and buses) in real-time. When the *speedup* or *skip-stop* tactics are allowed, many more possible paths are created for each bus. Solving instances with many transfers and three consecutive buses is not possible with the offline formulation for lines with more than thirty control stops when *speedup* or *skip-stop* tactics are allowed. This is why, the **PI** solution is used instead of the offline model to evaluate the performances of the OSO algorithms implemented in this research. When only the *hold* is allowed, the performance of the **PI** solution with a limited horizon of fifteen stops is very similar to the performance of the offline model with perfect information and an unlimited horizon.

On the other hand, Algorithm **D** does not model uncertainty either. The single generated scenario ω uses the mean historical values of all variables describing the state S and represents what the state of the system is on average. The optimal tactic $\sigma^*_{s_0}$ found by the offline model for stop s_0 is also returned. Algorithm **D** is designed to evaluate if sampling, used in Algorithms **C** and **R**, brings significant improvements compared

to a deterministic algorithm. Note that it is easier to implement Algorithm **D** compared to Algorithms **C** and **R** as it requires a single scenario using only historical data. It's computation time is therefore lower. More details on computation times are given in the next section.

5. Experiments

This section presents the case study used for the numerical experiments and presents a sensitivity analysis for the OSO algorithms described above. Finally, results from tests made on 29 different lines in a dense urban PT network are presented.

5.1. Case Study

The experiments are based on data provided by the Société de transport de Laval, a city in Canada with a population of 436,000. The PT network of the STL includes 46 bus lines, many of which connect to the Montreal subway network, and more than a thousand bus stops. Most passengers in this network use phone applications to receive live updates on schedules and estimated time of arrival. The STL provided a comprehensive dataset of bus routes, schedules, user ridership, and bus locations for a full month. This data is collected in real-time from multiple sources including AFC and APC systems installed on all vehicles. By using accurate smart card data and previous research on individual trip destinations (Trépanier et al. (2007)), the STL provides all passenger origin/destination pairs and reconstructed multi-segment passenger trips. This enables us to use real demand and real transfer flows in the experiments.

5.2. Scenario Generation

The data provided by the STL is used for testing as well as to generate scenarios on the future state of the system used by the OSO algorithms. To efficiently generate scenarios, the full month of historical data from the STL is pre-processed. Weekends and holidays are removed from the month of data. The remaining days are divided into three working sets: training (80% of data), testing (10% of data) and validation (10% of data). The training set is used to collect data on all variables that describe the state of the PT system, including travel times between stops, dwell times at stops, the number of boarding and alighting passengers at stops, the number of transferring passengers at stops, the number of transfers between lines, and headway times for all lines. For every stop, each variable's data is clustered based on the time of occurrence, resulting in clusters corresponding to rush-hours or lower-ridership periods. Extreme data points are removed from the clusters. Secondly, the testing set is used to test the scenario generation, and to conduct a sensitivity analysis of the algorithms. To generate a scenario we require four inputs: a main line, a starting time, the first stop to be considered in the scenario and a control horizon of h stops. For each stop in the horizon, travel times, dwell times, and passenger (and transfer) demands are independently and randomly sampled from the corresponding clusters. Feeder line delay information available at the starting time is used to generate the arrival times of transfer vehicles in the horizon. The sampled data is used to form bus trips for the main

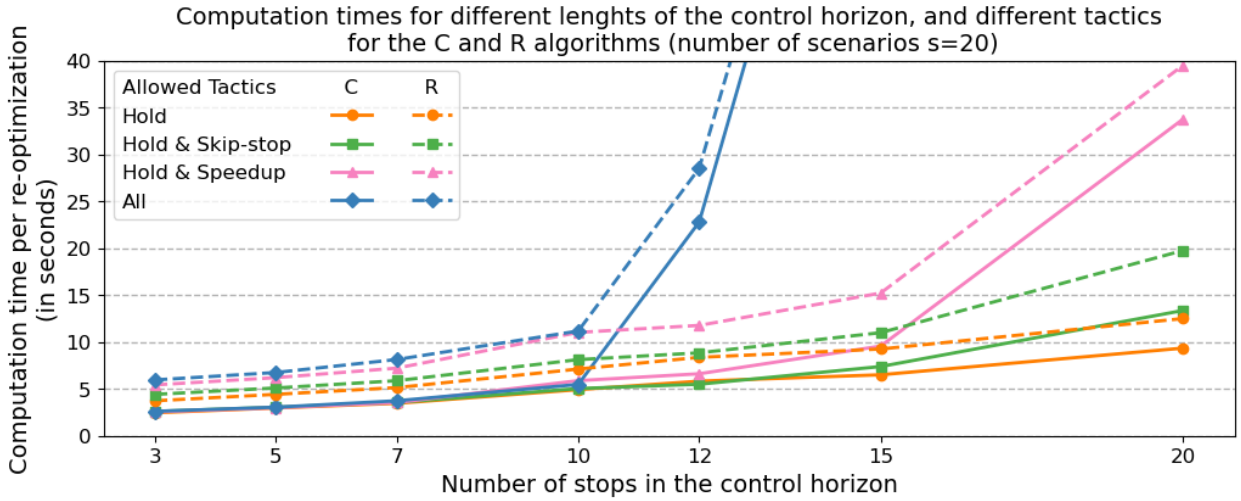

Figure 3 Computation times per re-optimization for different control horizon lengths, algorithms and tactics.

line and to allocate passenger flows along the main and feeder lines, resulting in a scenario ω. Sampling enables the generation of various random scenarios using the same inputs. Finally, the validation set is used to conduct all numerical experiments.

5.3. Sensitivity Analysis

This section discusses the sensitivity of the OSO algorithms to changes in the control horizon length, the price of out-of-bus waiting time for passengers and the number of scenarios at each re-optimization.

Figure 3 shows the average computation time per re-optimization in the dynamic environment for Algorithms **C** and **R** when using various control horizon lengths. The re-optimization computation time includes scenario generation, creating a time expanded graph for each scenario, computation times for the underlying problems and updating the state S. Computations are performed with twenty scenarios at each stop. Algorithm **D** and the **PI** solution are not shown in this picture as they are faster than Algorithms **C** and **R**, solving only one scenario per re-optimization. Computation time is linear with respect to the number of scenarios, and the length of the control horizon when using the *hold* tactic alone or *hold* with *skip-stop* tactics. However, when the *speedup* tactic is allowed, the computation time becomes exponential with respect to the length of the control horizon. In particular, when all tactics are allowed, it becomes too large to allow implementation in real-time for control horizons with more than twelve stops (when all stops are control points). Computation times indicate that all algorithms can be executed in real-time for a control horizon with ten stops which is sufficient for operational needs.

Figure 4 shows the performance of Algorithm **R** for different lengths of the control horizon and different allowed tactics. When only the *hold* tactic is allowed, increasing the control horizon leads to less missed transfers and improvements in total passenger travel time. For the cases with the *speedup* and/or *skip-stop* tactics, the same phenomenon is observed up to a horizon of ten stops. For a horizon of twelve stops there

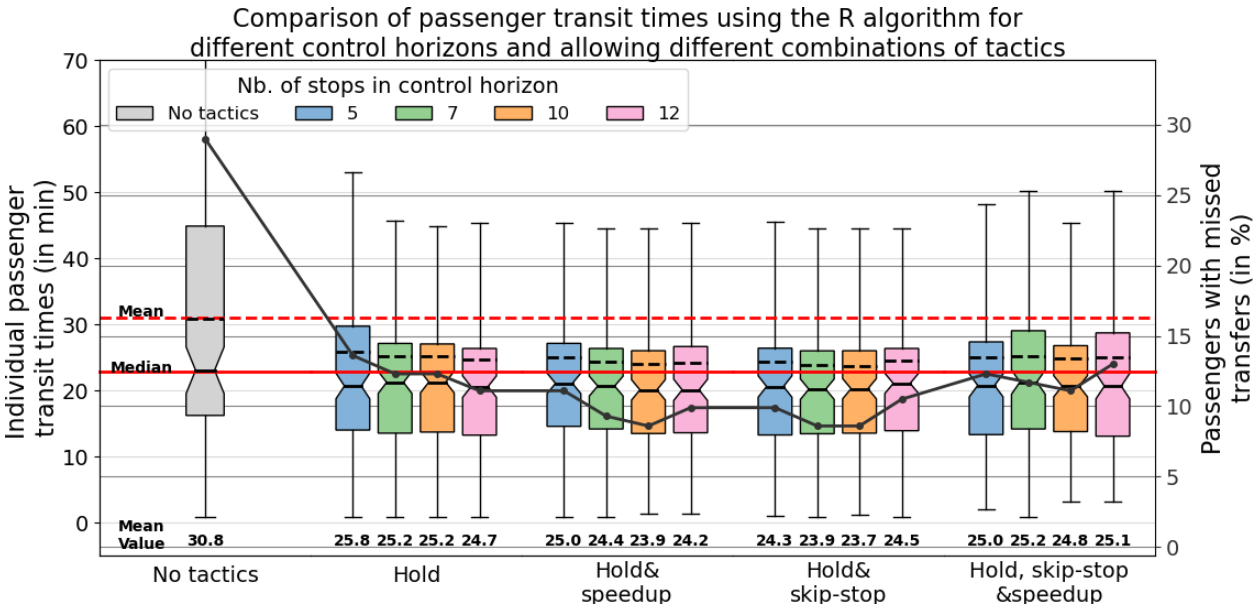

Figure 4 **Total passenger travel times and percentage of passengers with missed transfers for different lengths of the control horizon for Algorithm R .**

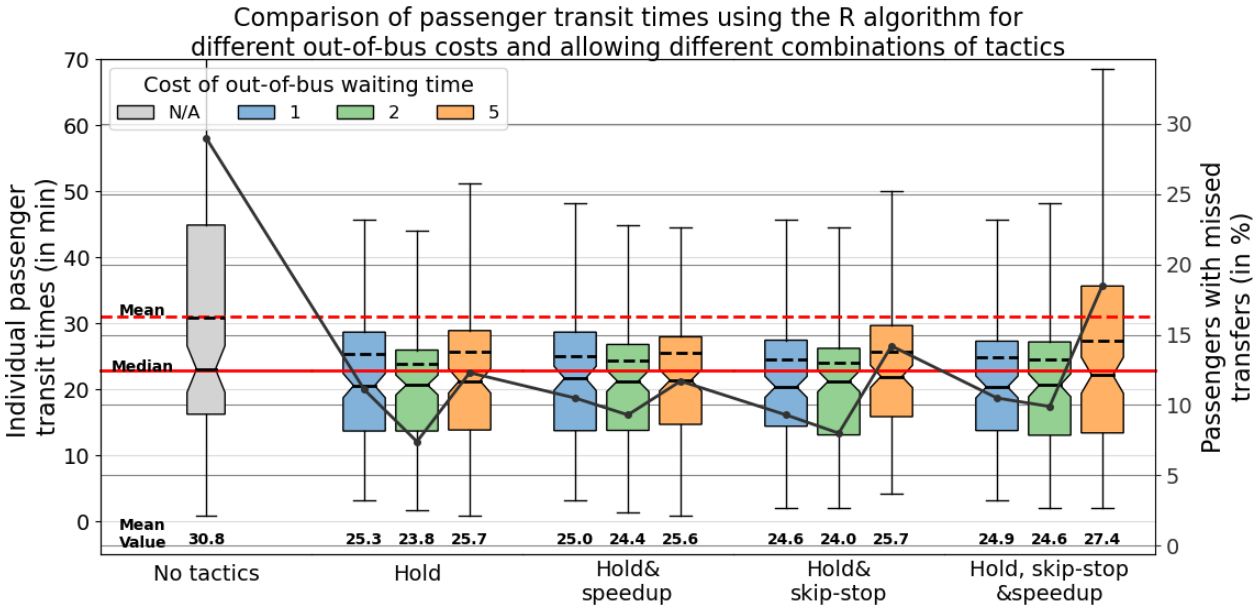

Figure 5 **Total passenger travel times and percentage of passengers with missed transfers for different out-of-bus waiting time costs for Algorithm R .**

is a slight increase in passenger travel times compared to the case with a horizon of ten stops. In this case, Algorithm **R** overuses the *speedup* and *skip-stop* tactics to try to catch up transfers at much later stops in the scenarios. This can result in arriving at some stops before the scheduled time or before a transferring bus. Similar results are observed for Algorithm **D** .

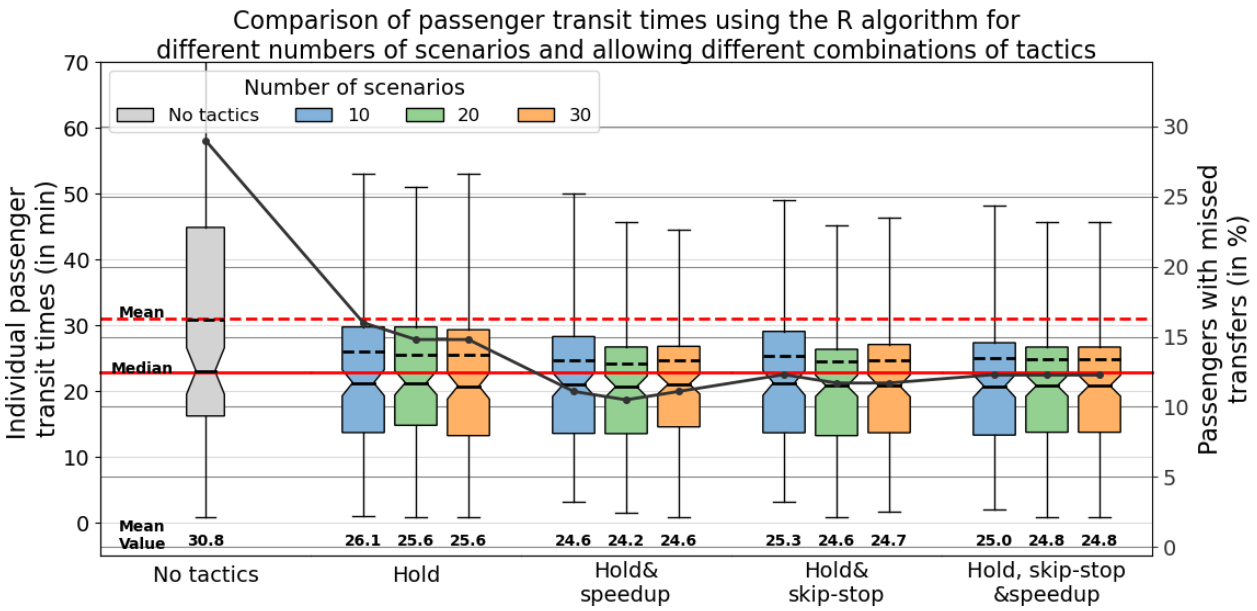

Figure 6 **Total passenger travel times and percentage of passengers with missed transfers for different numbers of scenarios solved at each re-optimization with Algorithm R.**

Figure 5 shows passenger travel times and the percentage of passengers with missed transfers for different out-of-bus waiting time costs for Algorithm **R**. Tests are performed with a control horizon of ten stops. There is an improvement in travel times and successful transfers when moving from a cost equal to one to a cost of two. In this case, out-of-bus waiting times (associated with waiting for transfers) are prioritized over time spent in-bus for passengers. However, for a cost greater than two, the model will make decisions that minimize out-of-bus wait times by using larger *hold* times, many *skip-stop* or *speedup* tactics in the generated scenarios. The 'mistakes' made by Algorithm **R** are more costly in this case because the applied tactics are more extreme. For example, let *hold* for two minutes be the optimal tactic to use at the first stop for some scenario. Using any other tactic at this stop will result in missing a transfer and will induce significant 'regret' values for all other tactics due to the large out-of-bus cost. If there is no transferring passenger in the real instance, the bus will wait for nothing and may miss future transfers. Therefore, a very large out-of-bus cost deteriorates both passenger travel times and the number of successful transfers for Algorithm **R** . This also suggests that some improvements can be made in the scenario generation.

Figure 6 shows passenger travel times for different numbers of scenarios solved for each re-optimization with Algorithm **R**. Increasing the number of scenarios improves total passenger travel time and lowers the percentage of passengers with missed transfers. However, these improvements stagnate between 20 and 30 scenarios.

Finally, Figure 7 shows passenger travel times when using a control horizon of ten stops and different numbers of scenarios solved at each re-optimization with Algorithm **C** . Total passenger travel time and the percentage of passengers with missed transfers decrease as the number of scenarios increases. For the case

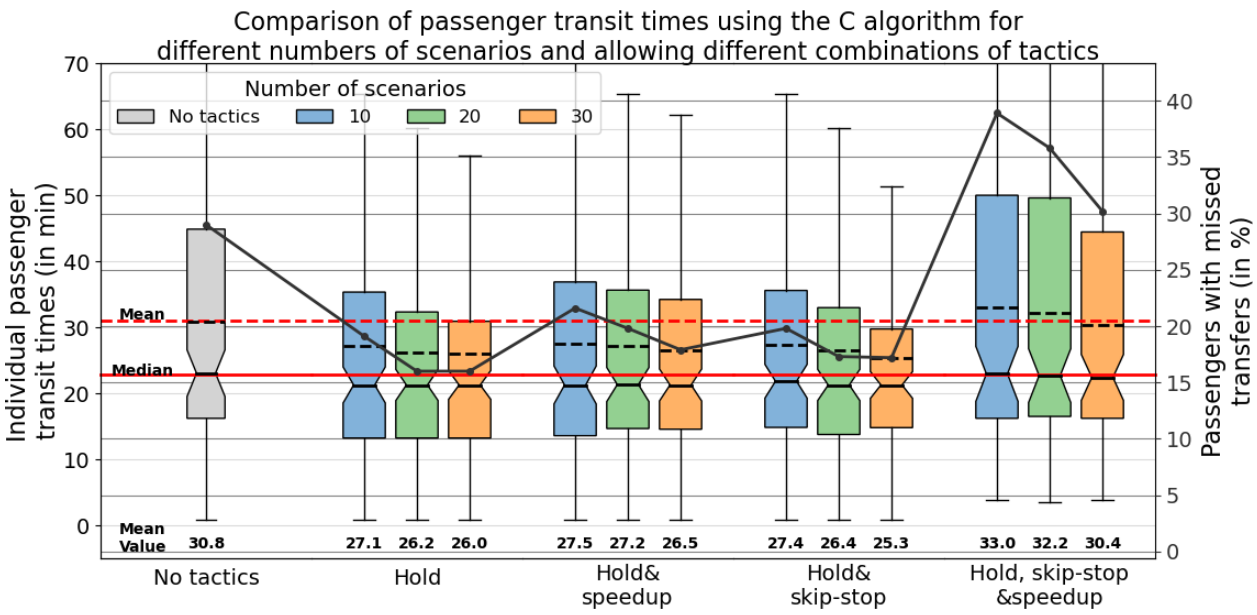

Figure 7 **Total passenger travel times and percentage of passengers with missed transfers for different numbers of scenarios solved at each re-optimization with Algorithm C.**

with *hold*, *skip-stop* and *speedup* tactics, Algorithm **C** performs worse than the case without any tactics. This is because using a tactic may be optimal for some scenarios but can result in significant time losses for others. It is important to note that the scenarios used for each combination of tactics with a given number of scenarios in Algorithms **C** and **R** are the same to allow the comparison of the performance of the two algorithms. However, the scenarios differ between combinations of allowed tactics and between different numbers of scenarios in the sensitivity analysis.

We conclude the sensitivity analysis by selecting a horizon of fifteen stops for tests that only allow the *hold* tactic and a horizon of ten stops for all tests conducted with *speedup* and/or *skip-stop* tactics. Additionally, twenty scenarios are used for all tests using Algorithms **C** and **R** . Finally, an out-of-bus cost of two will be used in all following numerical experiments. As shown in Figure 3, this combination of horizons and number of scenarios enables computation times per re-optimization that are generally under ten seconds, making it sufficiently fast for real-time operations. Moreover, this combination of parameters results in the greatest reduction in passenger travel time and in the percentage of passengers with missed transfers across the sensitivity analysis.

5.4. Results

This section presents results from 29 lines and provides detailed information on experiments conducted on two lines. Line 42 has a high frequency, a high occupancy and a high transfer demand. Line 33 has a medium frequency, an average occupancy and many transfer stops.

Figure 8 shows detailed results from computations on instances from line 42. These results consider more than 6500 passengers with close to 2000 transfers. Algorithm **C** has the worst performance, even

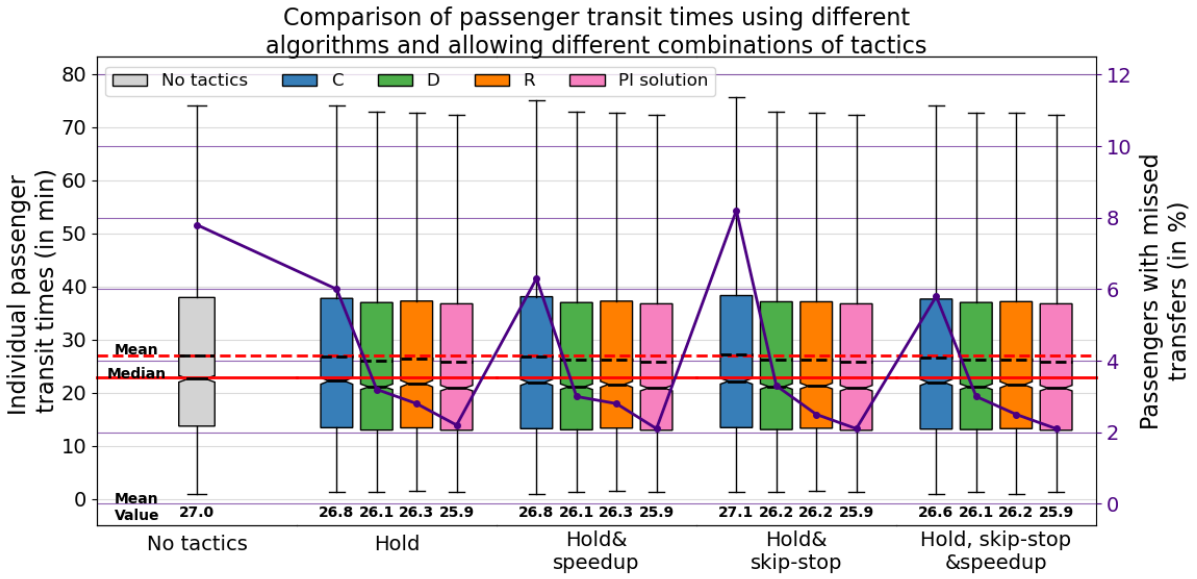

Figure 8 **Total passenger travel times and percentage of passengers with missed transfers for different algorithms and tactics for line 42.**

Note. The figure displays fifteen boxplots, each representing the transit times of individual passengers in minutes (as seen on the left y-axis). Outliers, which account for 0.7 percent of passengers, are not shown for clarity. The y-axis on the right shows the percentage of passengers who missed their transfer, calculated for each boxplot. The first boxplot represents passenger travel times without any tactics. This is what happened in real life. The next four boxplots display passenger travel times when only the *hold* tactic is allowed. The following four boxplots display the case where both the *speedup* and *hold* tactics are allowed. The following boxplots illustrate the case where the *hold* and *skip-stop* are allowed. The final group of boxplots represents the case where all three tactics are allowed. For each group of four boxplots, each box represents an algorithm used to compute optimal tactics. The first box from the left represents the results for Algorithm **C** , followed by Algorithm **D** , Algorithm **R** and the **PI** solution. The mean for each boxplot is written bellow it. A line representing the median and the mean of the 'no tactics' case is drawn across all boxplots to facilitate the comparison between cases.

performing worse than the 'no tactics' case when applying the *hold* with *skip-stop* tactics. It does not take into account that a tactic optimal for some scenarios may perform very poorly for other scenarios, leading to implementing tactics that are sometimes very costly in the real instance.

Algorithm **D** consistently outperforms the 'no tactics' case and Algorithm **C** , resulting in a significant reduction in the percentage of missed transfers without any negative impact on passenger travel times. For the *hold* tactic only case, the small increase in travel times caused by the *hold* tactic is offset by the significant improvement in travel times for passengers who successfully make their transfer. As a result, both the mean and median passenger travel times show improvements. Similar results are observed for the cases with *speedup* and *skip-stop* tactics. The number of passengers with successful transfers improves the most when all three tactics are allowed.

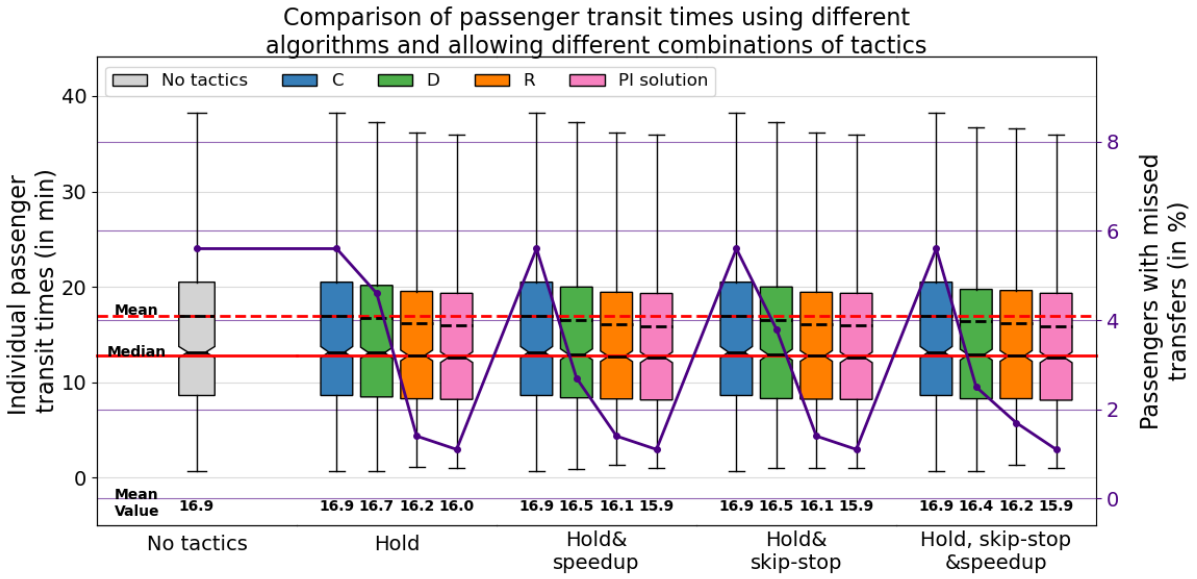

Figure 9 **Total passenger travel times and percentage of passengers with missed transfers for different algorithms and tactics for line 33.**

Algorithm **R** consistently outperforms the 'no tactics' case, as well as Algorithms **C** and **D**. When comparing Algorithms **D** and **R** in the *hold* tactic only case, Algorithm **R** shows greater improvements in the percentage of successful transfers, but the passenger travel times are slightly longer because Algorithm **R** applies the *hold* tactic more frequently. The same observation applies to the *hold* with *speedup* tactics case where the *speedup* tactic does not compensate for the time lost due to the *hold* tactic. The number of successful transfers is further improved when using the *hold* tactic in combination with the *skip-stop* tactic. The *skip-stop* tactic can be more effective than the *speedup* tactic in situations where congestion prevents the use of *speedup* or when the dwell time at stops is longer. In the model, we do not allow for *speedup* tactics when the travel time between two stops is too short, as the driver would not have enough time to accelerate and decelerate safely. Finally, Algorithm **R** shows a slightly lower percentage of passengers with missed transfers when all three tactics are employed. This demonstrates the potential of using available real-time data for real-time synchronization.

Figure 9 presents results from computations on instances from line 33, considering around 1000 passengers who transferred close to 250 times. For a medium frequency line, each successful transfer saves more time compared to a high frequency line. Although the percentage of passengers with missed transfers for the 'no tactics' case of line 33 is lower compared to line 42, the difference between the performances of Algorithms **D** and **R** is more pronounced. Algorithm **D** cannot compensate for any poor decisions when only the *hold* tactic is allowed. When both the *speedup* and *skip-stop* tactics are allowed, Algorithm **D** performs better but still underperforms compared to Algorithm **R**. This demonstrates the advantage of modelling the future using sampling. Additionally, it shows that the models are more effective on medium and lower

frequency lines, or lines with fewer passengers. In this case, a small number of tactics can have a significant impact on transferring passengers while inconveniencing fewer passengers onboard the buses. Algorithm **R** has the best performance when both the *hold* and *skip-stop* tactics are allowed. However, when all three tactics are allowed, there is a slight degradation in the results due to overusing tactics.

Table 5.1 Mean percentage of missed transfers and mean change in total passenger travel time compared to the real case for 27 lines of the STL network.

		Algorithm		**PI**
Tactics	Average	**D**	**R**	Solution
Hold	Passengers with missed transfers	4.21%	3.47%	1.62%
	Reduction in total passenger travel time	1.53%	2.39%	4.21%
Hold& Speedup	Passengers with missed transfers	4.81%	3.41%	1.50%
	Reduction in total passenger travel time	1.27%	2.89%	4.78%
Hold& Skip-stop	Passengers with missed transfers	5.02%	3.35%	1.48%
	Reduction in total passenger travel time	0.50%	2.49%	4.31%

THE AVERAGE PERCENTAGE OF PASSENGERS WITH MISSED TRANSFERS FOR THE **'NO TACTICS'** CASE IS **6.09%**.

Tests were conducted on 27 additional lines from the STL bus network, using the same methodology as in figures 8 and 9. The results are summarized in Table 5.1, and the detailed results are presented in figures 10, 11 and 12 in the Appendix. Algorithm **C** is not included as it did not improve the performance of the 'no tactics' case. Algorithm **R** outperforms Algorithm **D** for all tactics, reducing both the percentage of passengers with missed transfers and total passenger travel time. Moreover, Algorithm **R** mirrors the behaviour of the **PI** solution. They both achieve a lower percentage of missed transfers for the *hold* with *skip-stop* case compared to the *hold* with *speedup* case, at the cost of a smaller reduction in total passenger travel time. Algorithm **R**'s performance is closest to that of the **PI** solution in the case with *hold* and *skip-stop* tactics. There is no clear correlation between ridership and the percentage of passengers with missed transfers.

6. Conclusion

Synchronizing transfers is a crucial aspect of network planning for PT operators. The reliability of transfers is closely linked to the quality of service and the way passengers perceive a PT network. This research proposes a novel approach to solving the real-time transfer synchronization problem using three tactics: *hold*, *speedup* and *skip-stop*. First, we propose an arc-flow formulation that integrates all possible tactics and limits the number of variables for the offline problem. A discrete-event dynamic environment is created to simulate real-time operations. Two OSO algorithms are adapted to the transfer synchronization problem and tested within the dynamic environment using historical data and available real-time information. Algorithm **C** generates multiple scenarios at each iteration and selects the tactic that is optimal across most scenarios. Algorithm **R** selects the tactic with the smallest regret value when summed across all scenarios. Both OSO

algorithms have low enough computation times to allow for real-time implementation. To evaluate the performance of the OSO algorithms we implement Algorithm **D** and a **PI** solution. Algorithm **D** generates a single scenario representing the 'average' future state of the system at each re-optimization. The study tested the two OSO algorithms with all combinations of the three control tactics on 29 lines from a large-scale real dataset provided by the STL. The results were compared to the 'no tactics' case, to Algorithm **D** and to the **PI** solution. Algorithm **D** showed improvement in total travel times, individual travel times, and the number of successful transfers for any combination of tactics when compared to the 'no tactics' case. Algorithm **C** 's performance is hindered by its tactic selection process. Algorithm **R** outperforms both Algorithm **C** and Algorithm **D** and can significantly improve the number of successful transfers while reducing passenger travel times. The performance of Algorithm **R** is closest to that of the **PI** solution. The STL has a modern PT network with abundant real-time information. Passengers receive live updates on estimated bus arrival times and occupancy when using the transit application. This enables the implementation and use of those algorithms in real life.

The quality of scenario generation greatly affects the performance of Algorithms **C** and **R**. Additionally, computation times are limited in dynamic environments. Future research could focus on better scenario generation techniques and on graph compression techniques to reduce computation times. Graph sizes could also be reduced by decreasing the number of control stops while maintaining the same control horizon. Finally, the study of a single main line could be expanded to include multiple lines in the bus network.

References

Bent RW, Hentenryck PV (2004) Regrets only! online stochastic optimization under time constraints. *AAAI Conference on Artificial Intelligence*, URL https://api.semanticscholar.org/CorpusID:1819722.

Bent RW, Van Hentenryck P (2004) Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research* 52(6):977–987, URL http://dx.doi.org/10.1287/opre.1040.0124.

Berrebi SJ, Hans E, Chiabaut N, Laval JA, Leclercq L, Watkins KE (2018) Comparing bus holding methods with and without real-time predictions. *Transportation Research Part C: Emerging Technologies* 87:197–211, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2017.07.012.

Birge JR, Louveaux F (1997) *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering (Springer), ISBN 0387982175, URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0387982175.

Cats O, Larijani AN, Ásdís Ólafsdóttir, Burghout W, Andréasson IJ, Koutsopoulos HN (2012) Bus-holding control strategies: Simulation-based evaluation and guidelines for implementation. *Transportation Research Record* 2274(1):100–108, URL http://dx.doi.org/10.3141/2274-11.

Ceder AA, Hadas Y, McIvor M, Ang A (2013) Transfer synchronization of public transport networks. *Transportation Research Record* 2350(1):9–16, URL http://dx.doi.org/10.3141/2350-02.

Chang H, Givan R, Chong E (2000) On-line scheduling via sampling .

Chung EH, Nesheli MM, Shalaby A (2020) Transit holding control model for real-time connection protection. *Journal of Transportation Engineering, Part A: Systems* 146(4):04020021, URL http://dx.doi.org/10.1061/JTEPBS.0000332.

Daganzo C, Anderson P (2016) Coordinating transit transfers in real time. Institute of Transportation Studies, Research Reports, Working Papers, Proceedings qt25h4r974, Institute of Transportation Studies, UC Berkeley, URL https://ideas.repec.org/p/cdl/itsrrp/qt25h4r974.html.

De Filippo A, Lombardi M, Milano M (2019) How to tame your anticipatory algorithm. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 1071–1077 (International Joint Conferences on Artificial Intelligence Organization), URL http://dx.doi.org/10.24963/ijcai.2019/150.

Eberlein X, Wilson NHM, Bernstein D (1999) Modeling real-time control strategies in public transit operations. *Lecture Notes in Economics and Mathematical Systems* 325–346.

Gavriilidou A, Cats O (2019) Reconciling transfer synchronization and service regularity: real-time control strategies using passenger data. *Transportmetrica A: Transport Science* 15(2):215–243, URL http://dx.doi.org/10.1080/23249935.2018.1458757.

Gkiotsalitis K, Cats O (2021) At-stop control measures in public transport: Literature review and research agenda. *Transportation Research Part E: Logistics and Transportation Review* 145:102176, ISSN 1366-5545, URL http://dx.doi.org/https://doi.org/10.1016/j.tre.2020.102176.

Gkiotsalitis K, Cats O, Liu T (2022) A review of public transport transfer synchronisation at the real-time control phase. *Transport Reviews* 0(0):1–20, URL http://dx.doi.org/10.1080/01441647.2022.2035014.

Guevara CA, Donoso GA (2014) Tactical design of high-demand bus transfers. *Transport Policy* 32:16–24, ISSN 0967-070X, URL http://dx.doi.org/https://doi.org/10.1016/j.tranpol.2013.12.004.

Hadas Y, Ceder A (2008) Public transit simulation model for optimal synchronized transfers. *Transportation Research Record* 2063(1):52–59, URL http://dx.doi.org/10.3141/2063-07.

Hentenryck PV, Bent R (2009) *Online Stochastic Combinatorial Optimization* (The MIT Press), ISBN 0262220806.

Ibarra-Rojas O, Delgado F, Giesen R, Muñoz J (2015) Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological* 77:38–75, ISSN 0191-2615, URL http://dx.doi.org/https://doi.org/10.1016/j.trb.2015.03.002.

Kall P, Wallace S (1994) *Stochastic Programming*, volume 46. URL http://dx.doi.org/10.2307/2584504.

Kieu LM, Bhaskar A, Almeida PEM, Sabar NR, Chung E (2016) Transfer demand prediction for timed transfer coordination in public transport operational control. *Journal of Advanced Transportation* 50(8):1972–1989, URL http://dx.doi.org/https://doi.org/10.1002/atr.1440.

Kleywegt AJ, Shapiro A, Homem-de Mello T (2002) The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* 12(2):479–502, URL http://dx.doi.org/10.1137/S1052623499363220.

Kolcheva L, Legrain A, Trepanier M (2024) Data driven synchronization strategies of a bus line in a transit network. *Cahiers du CIRRELT* .

Lin M, Tang K, Yao X (2013) Dynamic sampling approach to training neural networks for multiclass imbalance classification. *Neural Networks and Learning Systems, IEEE Transactions on* 24:647–660, URL http://dx.doi.org/10.1109/TNNLS.2012.2228231.

Liu T, Cats O, Gkiotsalitis K (2021) A review of public transport transfer coordination at the tactical planning phase. *Transportation Research Part C: Emerging Technologies* 133:103450, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2021.103450.

Manasra H, Toledo T (2019) Optimization-based operations control for public transportation service with transfers. *Transportation Research Part C: Emerging Technologies* 105:456–467, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2019.06.011.

Philpott A, de Matos V (2012) Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research* 218:470–483.

Powell WB (2019) A unified framework for stochastic optimization. *European Journal of Operational Research* 275(3):795–821, URL http://dx.doi.org/10.1016/j.ejor.2018.07.01.

Shapiro A (2013) *Sample Average Approximation*, 1350–1355 (Boston, MA: Springer US), ISBN 978-1-4419-1153-7, URL http://dx.doi.org/10.1007/978-1-4419-1153-7_1154.

Shapiro A, Philpott A (2007) A tutorial on stochastic programming .

Ting CJ, Schonfeld P (2007) Dispatching control at transfer stations in multi-hub transit networks. *Journal of Advanced Transportation* 41(3):217–243, URL http://dx.doi.org/https://doi.org/10.1002/atr.5670410302.

Trépanier M, Tranchant N, Chapleau R (2007) Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems* 11(1):1–14, URL http://dx.doi.org/10.1080/15472450601122256.

Appendix

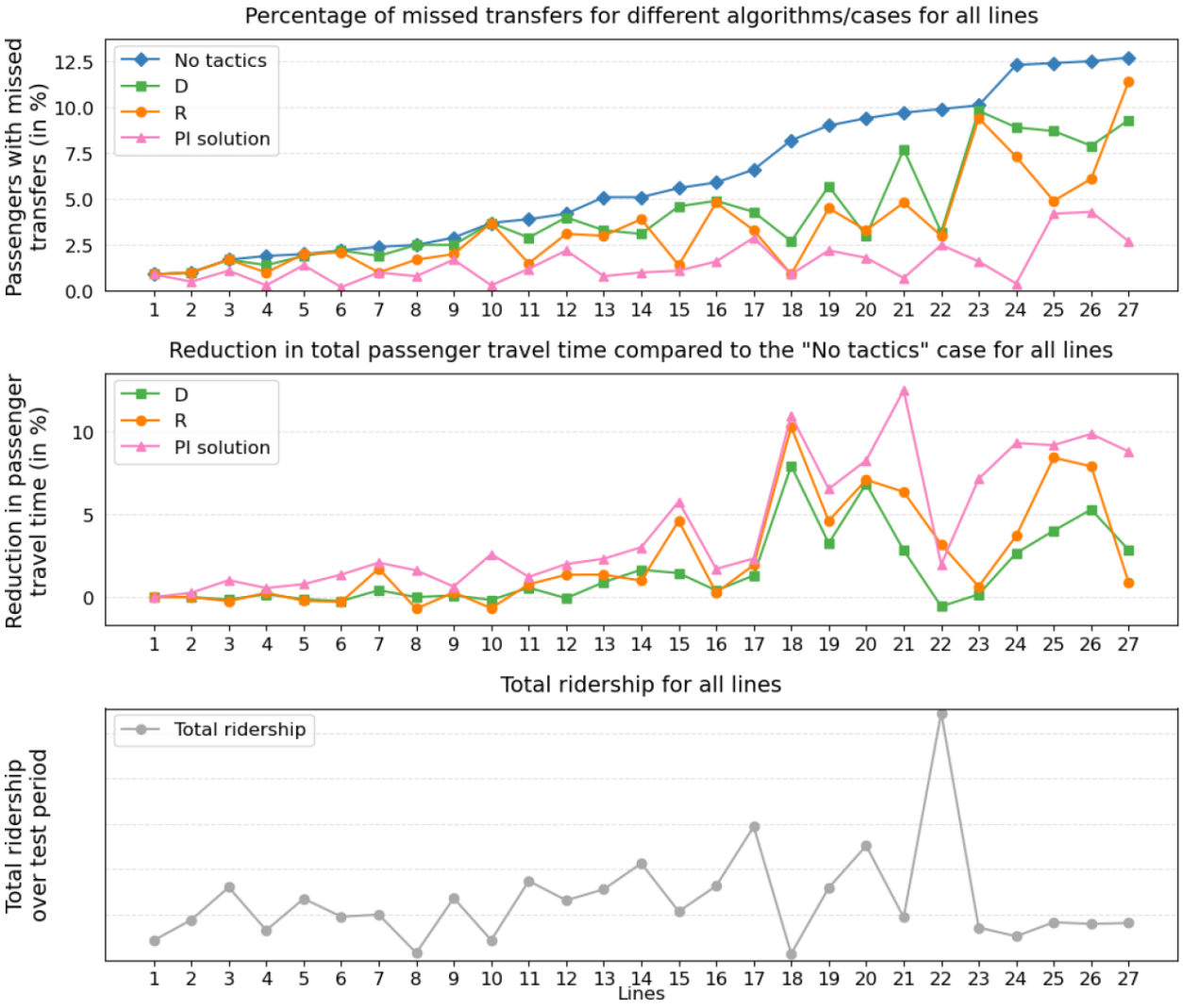

Figure 10 **Total passenger travel times, percentage of passengers with missed transfers and total ridership for different algorithms/cases using the *hold* tactic for 27 lines in the STL bus network.**

Note. The lines are sorted by percentage of passengers with missed transfers in the 'No tactics' Case.

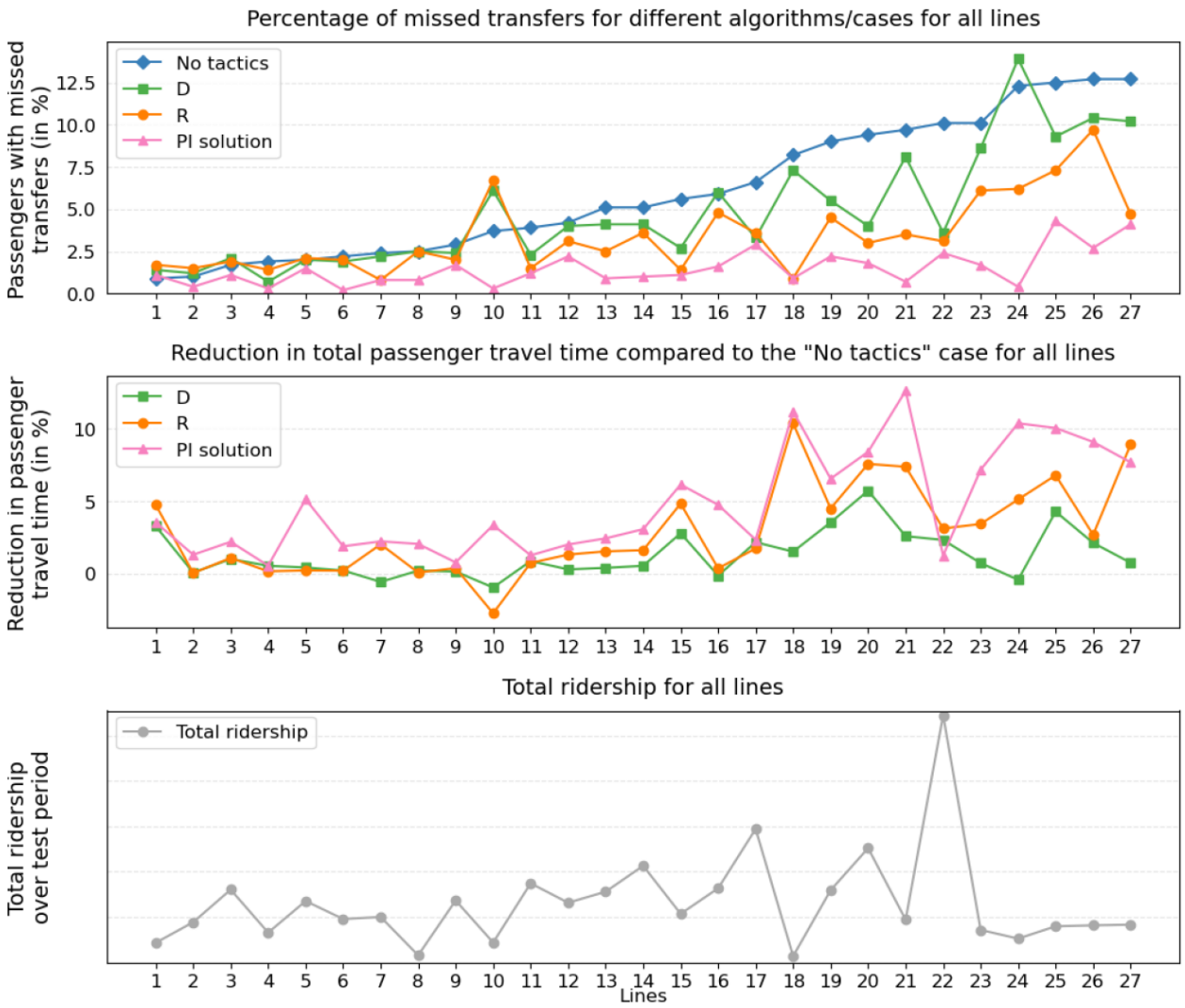

Figure 11 **Total passenger travel times, percentage of passengers with missed transfers and total ridership for different algorithms/cases using the *hold* and *speedup* tactics for 27 lines in the STL bus network.**

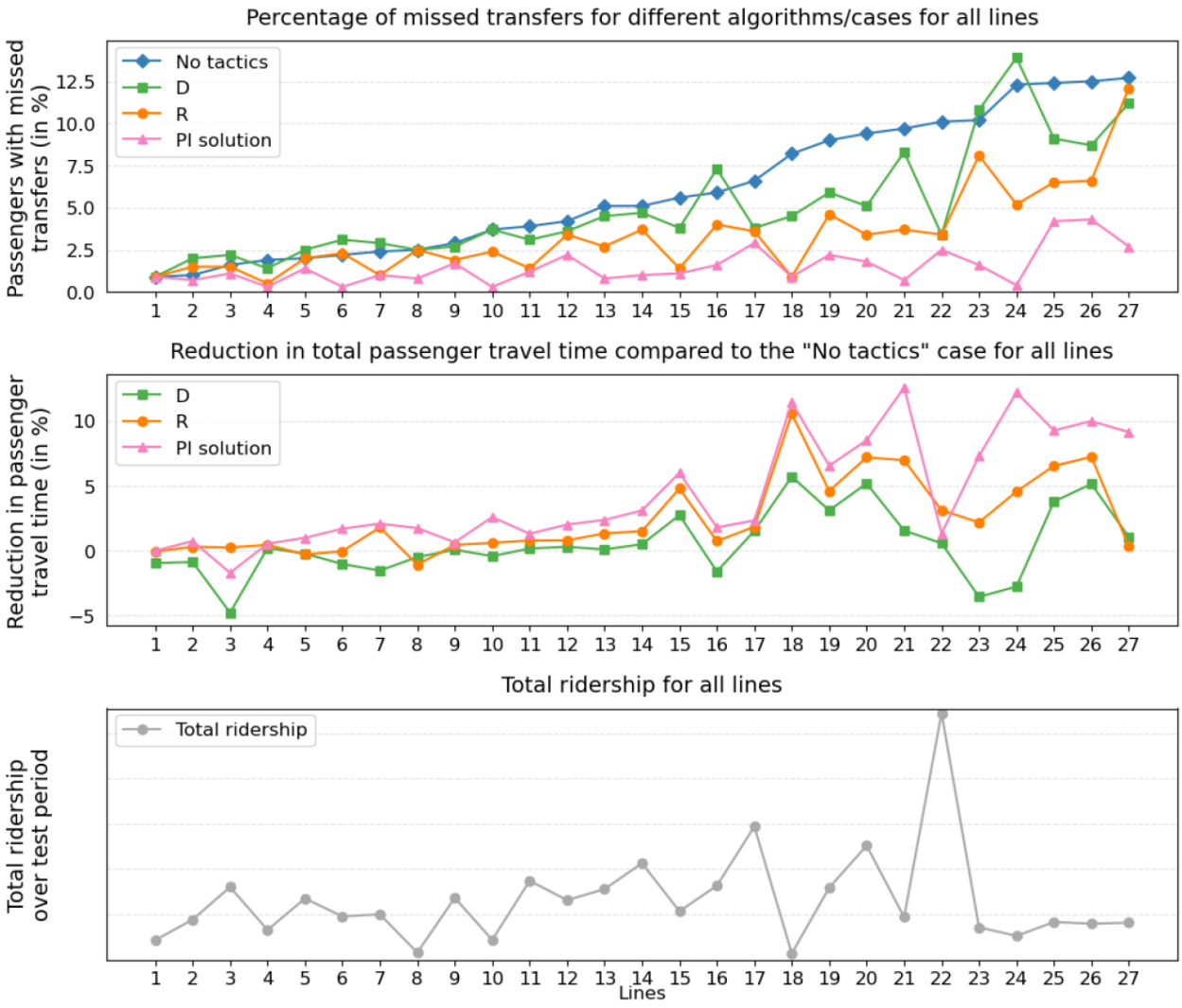

Figure 12 **Total passenger travel times, percentage of passengers with missed transfers and total ridership for different algorithms/cases using the *hold* and *skip-stop* tactics for 27 lines in the STL bus network.**