ABSTRACT
Given the unstable nature of the bus operations, the regularity of the bus services cannot be maintained throughout the day. Especially in densely populated areas, the inherent variability of the trip travel times affects the regularity of the daily services since the arrival times of several trips at stops might oscillate significantly from their expected values. Because of the high level of uncertainty, this work proposes a periodic rescheduling of the dispatching times of the daily bus trips to adjust continuously to the operational changes. A key differentiator from previous works is that a periodic rescheduling does not focus only on the running buses, but also reschedules the dispatching times of all remaining daily trips while considering operational constraints related to layover times and capacity limits. Typical scheduling problems (such as a bus scheduling problem) are NP-hard because of the discrete nature of the decision variables. Hence, the computational burden prohibits their solution with analytical methods. Catering for capacity limits increases further the complexity of our problem because it results in a non-smooth objective function which is not differentiable at every point of its domain. For this reason, this work proposes a sequential hill climbing method to search the solution space more efficiently and reschedule the dispatching times of trips in near real time. This approach is tested using real-time data from bus line 15 L in Denver, USA. Critical issues, such as to what extent can the periodic dispatching time rescheduling improve the regularity of daily operations, are investigated.

1. Introduction
The introduction of sensing mechanisms to bus operations ranging from telematics to smartcard payment systems has revolutionized the monitoring of passenger demand and vehicle trajectories (Calabrese, Di Lorenzo, Liu, & Ratti, 2011; Gkiotsalitis & Stathopoulos, 2015; Pelletier, Trépanier, & Morency, 2011). Automated vehicle location (AVL) and automated fare collection (AFC) data can unveil the main travel time and passenger demand patterns of bus services, thus offering valuable information for tactical and operational planning purposes (Gkiotsalitis & Alesiani 2019; Gkiotsalitis & Cats 2018; Ibarra-Rojas, Delgado, Giesen, & Muñoz, 2015).

In metropolitan areas where the bus services operate in high frequencies, the main key performance indicator (KPI) of the daily operations is the variability of the actual headways from the planned ones (Trompet, Liu, & Graham, 2011). These services are known as regularity-based services since their main objective is to maintain a stable headway which is close to the scheduled one (Gkiotsalitis, Eikenbroek, & Cats, 2019). In practice, because of the unstable nature of trip travel times, the actual headways differ significantly from their expected values resulting in excessive waiting times (EWT) for the bus passengers (Daganzo, 2009; Eberlein, Wilson, & Bernstein, 2001; Gkiotsalitis & Kumar, 2018; Koehler, Kraus, & Camponogara, 2011).

The monitoring capabilities of the bus operators, the introduction of concrete KPIs for measuring the regularity of services, the use of bus arrival time prediction algorithms, and the implementation of corrective control measures to the bus operations have played a significant role on improving the performance of high-frequency services in several urban areas. For instance, the EWT of passengers in London has been reduced from 4 min in 1979 to 1.2 min in 2012 (for more information about the EWT at each specific bus line in London the interested reader can refer to TfL (2017)). Bus operators in urban...
areas are motivated to improve further the service regular-
ity because public transport authorities (PTAs) provide
monetary incentives to bus operators (i.e. the Land
Transport Authority in Singapore have offered as much as
6000 Singaporean dollars for each 0.1 min EWT improve-
ment per line (Leong, Goh, Hess, & Murphy, 2016) or
exclude them from future tenders if they underperform
consistently over a prolonged period.

Reacting on those needs, bus operators have developed
sophisticated control centers that can assist bus drivers to
avoid bus bunching (i.e. ask them to change their dis-
patching times to reflect on operational changes). Notwithstanding, near real-time decisions on dispatching
time changes are based on human experience and/or sim-
plistic local optimization models (Fu & Yang, 2002) with-
out exploiting the full potential of real-time control on
improving the service regularity. This issue is investigated
in this study with the introduction of a near real-time dis-
patching time rescheduling approach that can cater for
disruptions in service regularity while considering layover
and capacity constraints.

1.1. Related studies
The problem of bus operations’ optimization has evolved
over the years from tactical planning that aims at improv-


...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-
to the number of training samples resulting in a large amount of computation time when applied in large-size problems (Leontitsis, Bountis, & Pagge, 2004).

Yu, Lam, and Tam (2011) tested several supervised learning methods such as SVM, ANN, k nearest neighbors algorithm (k-NN) and linear regression. They also concluded that SVM have the highest accuracy in terms of mean absolute percentage error (MAPE) and root mean square error (RMSE) with ANN a close second. They also showed that using bus running times of multiple routes to predict the bus arrival time of each route in isolation is a better option than using models based on the bus running times of a single route. The good performance of ANN was also supported in the works of Chien, Ding, and Wei (2002) and Jeong and Rilett (2004). The former tested two link-based and stop-based models for bus arrival time prediction and the latter compared the performances of ANN against the historical data based model and regression models showing that the ANN model exhibits a better performance in terms of prediction accuracy.

Non-parametric models, such as the k-NN, are also of interest owing to their relatively simple prediction method that does not require to estimate parameters. These results in low computational times, thus such prediction models are particularly interesting in real-time applications. Chang et al. (2010) tested such a model with real-world bus data from Seoul, Korea and reported an average execution time of only 0.59 s per forecast.

The use of bus arrival time predictions in near real-time control strategies has been recently studied by Berrebi et al. (2018) who focused on bus holding control with arrival time predictions. They showed that using predictions in real-time control is beneficial on improving the service regularity. In contrast, such studies are not available on the topic of real-time rescheduling and this work tries to fill this gap by investigating the effect of arrival time uncertainty to the performance of fixed operations and periodically rescheduled ones.

1.2. Focus of this work

Compared to the existing studies, this work shows how the daily operations can be rescheduled in rolling horizons where at each rescheduling instance the implications of the rescheduling actions to the daily trips are fully considered. By following this approach, the bus schedule is continuously adjusted to the needs of the present conditions. A key differentiator from the previous works on dynamic bus control is that a complete rescheduling does not focus only on the running buses, but also adjusts the dispatching times of all daily trips while considering layover/capacity constraints and the degree they are affected by the dispatching time modifications.

This is a key factor because several studies such as Daganzo (2009) and Chen, Adida, and Lin (2013) showed that when considering only the running buses for performing dynamic control, the bus operations in the long run might deteriorate. Intuitively, this might occur because the implications to the future bus trips are not considered. In more detail, the main contributions of this article are:

- the modeling of the bus rescheduling problem as a rolling-horizon optimization problem that considers operational and regulatory constraints;
- the investigation of its complexity concluding that it is an NP-Hard problem with a non-smooth objective function;
- the introduction of a metaheuristic sequential hill-climbing method for obtaining an improved rescheduling solution in near real-time by avoiding the exhaustive exploration of the entire solution space;
- the demonstration of the theoretical improvement of the regularity of bus services when rescheduling the operations in rolling horizons by using General Transit Feed Specification (GTFS)-Realtime data from bus line 15L in Denver, US.

In the following section, the periodic rescheduling problem is formulated as a rolling-horizon optimization problem to update the dispatching times of trips periodically. In Section 3, an evolutionary optimization method based on sequential hill climbing (SHC) is introduced. This method reschedules the dispatching times of the remaining trips in near real-time to cater for the needs of the current operational conditions. Finally, the rolling horizon rescheduling is applied to a bus line with 42 bus stops where at each rolling horizon instance new dispatching times for the remaining trips of the day are computed. The results from this rolling horizon optimization are discussed in Sections 4 and 5.

2. Rescheduling the bus operations with rolling horizon optimization

Rolling-horizon optimization is used in several optimization problems which are affected from external traffic (such as fastest-path problems or traffic signal control problems (Aboudolas, Papageorgiou, Kouvelas, & Kosmatopoulos, 2010) because of the need to update occasionally the computed solutions for adapting to traffic changes. The schedule of a bus line that
is devised at the tactical planning stage cannot guarantee optimal bus operations since external factors such as traffic can entirely change the trajectories of the bus trips from time to time (Gkiotsalitis & Maslekar, 2018a, b). Due to that, the bus operations should be occasionally reevaluated utilizing the information obtained from telematics. To achieve that, the integration of the monitoring part of a traffic management system (TMS) with the control automation part (i.e. continuous rescheduling) is required as described in Figure 1.

Figure 1 shows that at the beginning of the day we have a set of expected arrival times (EAT) for each bus trip at each bus stop that define the daily schedule. Note that the daily schedule is constructed by setting the dispatching times of each daily trip and estimating the arrival times of trips at stops by using travel time and dwell time information. The EATs of bus trips at stops are generally estimated with the use of historical automated vehicle location (AVL) data (Hounsell, Shrestha, & Wong, 2012; Shen, Xu, & Zeng, 2016). After some time period (Time Horizon 1), the bus positioning data generated by the in-vehicle telematics is processed to extract the actual arrival times (AAT) of the running trips at the stops. The obtained AATs provide new information to the bus rescheduling problem and are used together with the EATs of the remaining trips of the day to modify the dispatching times of the remaining trips.

This procedure is iterative and terminates at the end of the daily operations. For instance, there will be time horizons 2, 3, … where at each time horizon the generated AVL data is processed to extract the arrival times of bus trips at stops. Then, it is used as input, along with the EATs of the remaining daily trips, for rescheduling the operations of the remaining trips.

To model this rolling-horizon optimization problem, three 2-dimensional matrices $A$, $E$ and $D$ are introduced. The dimensions of these matrices are $|N| \times |S|$ where $|N|$ is the number of planned daily trips for one service line and $|S|$ the number of stops of that line. Each element $a_{i,j}$ of matrix $A$ represents the AAT of bus trip $i$ at bus stop $j$ if this information is already available from AVL data. In case it is not, then a dummy value is used ($a_{i,j}$ is set equal to 0 to indicate that this trip has not arrived yet at stop $j$). In addition, each element $e_{i,j}$ of matrix $E$ represents the EAT of bus trip $i$ at bus stop $j$. Finally, matrix $D$ is a combination of matrices $A$ and $E$ and is continuously updated at each rolling horizon instance. Matrix $D$ comprises of the AATs of buses at bus stops and the EATs of future bus trips at the corresponding stops:

$$d_{i,j} = \begin{cases} a_{i,j}, & \text{if } a_{i,j} \neq 0 \\ e_{i,j}, & \text{otherwise} \end{cases} (1)$$

Every time matrix $A$ receives an update, the new information replaces the expected times of arrival at matrix $D$ and a new optimization is performed considering matrix $D$ as optimization input (re-scheduling in rolling horizons). The main notation used in the modeling phase is presented below.

![Figure 1](image_url)

**Figure 1.** At each rolling horizon the GTFS-RT data is processed for extracting the actual arrival times of buses at stops and rescheduling the remaining daily trips.
The EAT of a bus trip \( i \) at stop \( j \) depends on the expected trip travel times between stops, the expected dwell times at stops, the scheduled dispatching time of the trip, \( \delta_i \), and the dispatching time modification during the rescheduling phase, \( x_i \): 

\[
e_{i,j}(x) = (\delta_i + x_i) + \sum_{\phi=1}^{j-1} t_{i,\phi} + \sum_{\phi=1}^{j-1} k_{i,\phi}(x) \tag{2}
\]

where \( \sum_{\phi=1}^{j-1} t_{i,\phi} \) is the sum of the expected link travel times from the dispatching stop until stop \( j \) and \( \sum_{\phi=1}^{j-1} k_{i,\phi}(x) \) the sum of the dwell times.

### 2.1. Objective function

In high-frequency bus services, the main objective is the improvement of the service regularity by adhering to the scheduled headways. Transport authorities, such as the Transport for London (TfL), the Land Transport Authority (LTA) in Singapore and others measure the service regularity as the excessive time that passengers have to wait at stops. This EWT is the extra waiting time of passengers over and above the waiting time that might be expected if all the buses on the route run on time. It reflects the regularity of the service and is the difference between the actual average waiting time of passengers at bus stops and the scheduled average waiting time.

The average scheduled waiting time of passengers at a bus stop \( j \in S \) is a fixed value which is obtained from the daily schedule of bus headways. Assuming random passenger arrivals at each stop \( j \in S \), the waiting time of the average passenger is equal to half the headway between two adjacent trips. Hence, the average scheduled waiting time is (Hounsell et al., 2012):

\[
\gamma_j = \frac{\sum_{i=1}^{N-1} (r_{i+j} - r_{i})^2}{2 \sum_{i=1}^{N-1} (r_{i+j} - r_{i})}, \text{ for each stop } j \in S \tag{3}
\]

Similarly, the actual average waiting time of passengers at a bus stop \( j \in S \) is measured as

\[
\bar{\gamma}_j = \frac{\sum_{i=1}^{N-1} (d_{i,j} - d_{i,j})^2}{2 \sum_{i=1}^{N-1} (d_{i,j} - d_{i,j})}, \text{ for each stop } j \in S \tag{4}
\]

where \( d_{i,j} = a_{i,j} \) if bus trip \( i \) has already arrived at bus stop \( j \) and \( d_{i,j} = e_{i,j}(x) \) if not. The expected average excess waiting time (EWT) per passenger at a rolling horizon rescheduling instance is calculated then as follows (Transportation Research Group, 1997):

\[
f(x) = \frac{1}{\sum_{j \in S} \bar{w}_j} \sum_{j \in S} \bar{w}_j (\bar{\gamma}_j(x) - \gamma_j) \tag{5}
\]

where \( \bar{w}_j \geq 0 \) is a weight parameter for every stop \( j \in S \) and is used to assign more weight to stops with higher passenger demand because schedule deviations at such stops will impact more the average waiting times of passengers. Note that in the extreme case where \( \bar{w}_j \) is set to zero for a stop \( j \in S \), that stop is not included in the consideration of schedule deviations (see Cats et al., 2011; Van Oort et al., 2010). Note also that the value of each weight parameter \( \bar{w}_j \)
can be derived after calibration with the use of stop-level demand data from the actual operations.

In Equation (5), \( f(x) \) is the objective function of the bus rescheduling problem at each rolling horizon optimization instance which should be minimized by computing new dispatching times that improve the service regularity.

### 2.2. Layover and capacity constraints

The layover time \( l \) consists of (i) the driver’s break time that is required before starting a next trip plus (ii) the required headdead time for traveling from the last stop of the previous trip to the first stop of the next trip. The deadhead time might be equal to the required deadhead time for traveling from the arrival of the previous trip \( Z_i \) should be greater than the layover time limit:

\[
d_{i,1}(x) - \left( d_{Z_i,S}(x) + k_{i,S}(x) \right) \geq l, \quad \forall i \in N \tag{6}
\]

Each trip \( i \in N \) is also operated by a bus with a capacity \( \psi_i \) in number of passengers. Let \( \lambda_{p,j} \) be the passenger arrival rate at stop \( j \) during the \( p \)-th hour of the day expressed in passengers per minute. Note that the use of a different passenger arrival rate per hour \( p \) allows us to represent better the temporal demand variation across the day. Note also that the arrival rate \( \lambda_{p,j} \) does not depend on the arrival time of buses at the respective stops. This implies that passengers do not coordinate their arrival times at stops with the arrival times of buses. This assumption holds true for high-frequency services that are the focus of this study because if the headways are negligibly small, passengers do not consider the scheduled arrival times of buses when deciding about the timing of their arrival. In contrary, in low-frequency services passengers tend to consider the scheduled arrival times of buses and adjust the timing of their arrivals accordingly. Welding (1957) and Bartholdi and Eisenstein (2012) have established that passengers start to coordinate the timing of their arrivals at low-frequency services with less than five buses per hour.

Let also \( \nu_j \) be the expected percentage of onboard passengers that will alight at stop \( j \), where \( \nu_j \) is always equal to 100% because all onboard passengers will alight at the last stop, \( S \). The busload \( \beta_{i,j}(x) \) of bus trip \( i \in \{2, 3, ..., |N|\} \) when it departs from stop \( j \in \{2, 3, ..., |S|\} \) is expressed as:

\[
\beta_{i,j}(x) = \beta_{i,j-1}(x) - \nu_j \beta_{i,j-1}(x) + \lambda_{p,j}
\]

\[
\left( d_{i,j}(x) + k_{i,j}(x) \right) - \left( d_{i-1,j}(x) + k_{i-1,j}(x) \right), \quad \forall i \in \{2, ..., |N|\}, j \in \{2, ..., |S|\}
\]

Note that in Equation (7) the busload of a trip \( i \in N \setminus \{1\} \) after it departs from stop \( j \in S \setminus \{1\} \) is equal to its busload before arriving at stop \( j \), expressed as \( \beta_{i,j-1}(x) \), minus the number of passengers that alighted at stop \( j \), expressed as \( \nu_j \beta_{i,j-1}(x) \), plus the number of passengers boarding at stop \( j \). The latter depends on the passenger arrival rate at stop \( j \) and the headway between trip \( i \) and its preceding one \( i-1 \) as it is expressed in the relation \( \lambda_{p,j}[(d_{i,j}(x) + k_{i,j}(x)) - (d_{i-1,j}(x) + k_{i-1,j}(x))] \). Finally, we note that in Equation (7), the expected percentage of onboard passengers that will alight at stop \( j \) is not time-dependent. This might not always be true in practice because the distribution of passenger demand might vary across time. In such case, and without loss of generality, \( \nu_j \) can be replaced by \( \nu_{j, \omega} \), where \( \omega \) is the time period within which the value of \( \nu_j \) remains stable.

Equation (7) cannot be applied for all trips \( i \) and stops \( j \). Namely, we require boundary conditions for trip \( i = 1 \) and stop \( j = 1 \). In more detail, there is no trip preceding the first trip of the day, \( i = 1 \). Hence, for \( i = 1 \) the busload \( \beta_{1,j}(x) \) is expressed as:

\[
\beta_{1,j}(x) = \beta_{1,j-1}(x) - \nu_j \beta_{1,j-1}(x) + \bar{b}_j, \quad \forall j \in \{2, ..., |S|\}
\]

where \( \bar{b}_j \) are the expected passenger boardings at each stop \( j \) for the first trip of the day.

The second boundary condition is related to the busload at the first stop. When a bus arrives at the first stop is initially empty. Additionally, there are no alightings at the first stop. That is,

\[
\beta_{1,1}(x) = \lambda_{p,1} \left( d_{1,1}(x) + k_{1,1}(x) \right), \quad \forall i \in \{2, ..., |N|\}
\]

and

\[
\beta_{1,1}(x) = \bar{b}_1
\]

Obviously, in order to ensure that the busload is always below the capacity of the bus that operates trip \( i \), the following inequality constraint should be satisfied at all bus stops:

\[
\psi_i \geq \beta_{i,j}(x), \quad i \in N, \forall j \in S \tag{11}
\]

where \( \psi_i \) is the capacity of each bus trip \( i \in N \) in number of passengers. The layover and capacity
constraints result in a total of \(|N| + |N||S|\) linear inequality constraints for the bus rescheduling problem. Additionally, assuming that

i. passengers use the same door channel for boardings and alightings

ii. there are no additional passenger arrivals while the bus is dwelling since the dwell time is negligibly small

the dwell time of trip \(i \in N \setminus \{1\}\) at stop \(j \in S \setminus \{1\}\) is:

\[
k_{i,j}(x) = \eta_1 \lambda_{p,j}[d_{i,j}(x) - d_{i-1,j}(x)] + \eta_2 \nu_j \beta_{i,j-1}(x)
\]

(12)

where \(\eta_1 \lambda_{p,j}[d_{i,j}(x) - d_{i-1,j}(x)]\) is the required time for passenger boardings and \(\eta_2 \nu_j \beta_{i,j-1}(x)\) the required time for passenger alightings of trip \(i\) at stop \(j\).

When buses depart from the first stop, we do not have passenger alightings. Hence, the following boundary condition applies:

\[
k_{i,1}(x) = \eta_1 \lambda_{p,1}[d_{i,1}(x) - d_{i-1,1}(x)]
\]

(13)

### 2.3. Mathematical program of the rescheduling problem

The dispatching time modifications, \(x\), of bus trips can theoretically take any integer value. It is prudent though to ensure that the rescheduled dispatching times do not differ significantly from the originally planned ones because this can affect (a) the crew schedules; and (b) the practical implementation of the rescheduling. For this reason, we enforce any dispatching time modification \(x_i\) from its planned dispatching time \(\delta_i\) to be within the range of ±30 minutes. Therefore, each decision variable can take any value from the discrete set \(Q = \{-30, -29, 0, \ldots, +29, +30\}\) min.

At a rolling-horizon optimization instance, dispatching time modifications can only be applied to future trips that have not been dispatched yet. For past trips that have been already dispatched, the dispatching time modifications are set equal to zero. Note that if a bus trip \(i\) is already dispatched, this is indicated by \(a_{i,j} \neq 0\) at stop \(j = 1\). Therefore:

\[
x_i = \begin{cases} 
\text{any value from the set } Q & \text{if } a_{i,1} = 0 \\
0 & \text{otherwise}
\end{cases}
\]

(14)

To calculate the dispatching times of the remaining trips of the day that minimize the EWTs of passengers at a rolling-horizon optimization instance, the following Integer Nonlinear Programming (INL) problem needs to be solved in near real-time:

\[
\begin{align*}
\min_f(x) &= \frac{1}{\sum w_j} \sum_{j=1}^{\lceil S \rceil} \left( \sum_{i=1}^{\lceil N \rceil} d_{i,j}(x) - d_{i,j}(x) \right)^2 - \frac{\sum_{i=1}^{\lceil N \rceil} (r_{i+1,j} - r_{i,j})^2}{2 \sum_{i=1}^{\lceil N \rceil} (r_{i+1,j} - r_{i,j})^2} \\
\text{s.t. } & d_{i,1}(x) - \left( d_{\text{Z.},\lceil S \rceil}(x) + k_{i,\lceil S \rceil}(x) \right) - l \geq 0, \ \forall i \in N \\
& \psi_i \geq \beta_{i,j}(x), \ \forall j \in S, \forall i \in N \\
& x_i = \begin{cases} 
0 & \text{if } a_{i,1} = 0 \\
Q & \text{otherwise}
\end{cases} \\
& \beta_{i,j}(x) = \beta_{i,j-1}(x) - \nu_j \beta_{i,j-1}(x) + \lambda_{p,j} \left( d_{i,j}(x) + k_{i,j}(x) \right) - \left( d_{i-1,j}(x) + k_{i-1,j}(x) \right), \\
& \forall i \in N \setminus \{1\}, j \in S \setminus \{1\} \text{ where } d_{i,j}(x) \in \rho - \text{th hour} \\
& \beta_{1,j}(x) = \beta_{1,j-1}(x) - \nu_j \beta_{1,j-1}(x) + \bar{b}_j, \ \forall j \in \{2, \ldots, |S|\} \\
& \beta_{i,1}(x) = \lambda_{p,1} \left( d_{i,1}(x) + k_{i,1}(x) \right) - \left( d_{i-1,1}(x) + k_{i-1,1}(x) \right), \ \forall i \in N \setminus \{1\} \land d_{i,1}(x) \in \rho - \text{th hour} \\
& \beta_{i,1}(x) = \bar{b}_1 \\
& d_{i,j}(x) = \begin{cases} 
\bar{a}_{i,j} & \text{if } a_{i,j} \neq 0 \\
\bar{e}_{i,j}(x) & \text{otherwise}
\end{cases} \\
& e_{i,j}(x) = (\delta_i + x_i) + \sum_{\phi=1}^{i-1} \ell_{i,\phi} + \sum_{\phi=1}^{i-1} k_{i,\phi}(x) \\
& k_{i,j}(x) = \eta_1 \lambda_{p,j} \left[ d_{i,j}(x) - d_{i-1,j}(x) \right] + \eta_2 \nu_j \beta_{i,j-1}(x), \ \forall i \in N \setminus \{1\}, \forall j \in S \setminus \{1\} \\
& k_{i,1}(x) = \eta_1 \lambda_{p,1} \left[ d_{i,1}(x) - d_{i-1,1}(x) \right], \ \forall i \in N \setminus \{1\} \\
& Q = \{-30, -29, 0, \ldots, +29, +30\} \ \text{min}.
\end{align*}
\]
This combinatorial optimization problem has \(|N| + |N| |S|\) linear constraints and a fractional, nonlinear objective function. Given its integer programing nature, the problem is NP-Hard and because of its fractional objective function its continuous relaxation cannot be solved to optimality with analytical methods. For this reason, a globally optimal solution can be computed only with exhaustive search of the solution space (brute force) which can only be applied to small-scale problems.

**Lemma 2.1.** The size of the solution space increases exponentially with the number of bus trips that need be rescheduled.

**Proof.** Let \(n \leq N\) be the subset of daily trips that have not been dispatched at the time instance we perform a rescheduling. Given the combinatorial nature of the problem, the dispatching time modification of each trip \(i \in n\) can take one of the \(|Q| = 61\) values from the set \(Q = \{-30, ..., +30\}\) min. If the dispatching time modifications of all trips in the set \(n\) are fixed and one needs to find the optimal dispatching time modification of only one bus trip \(i \in n\), then \(|Q| = 61\) different options need to be explored. If one needs to determine the optimal dispatching times of all trips from the set \(n\), then the number of combinations that need to be evaluated becomes \(|Q|^n\) and the solution space exploration increases exponentially with the number of trips that need to be rescheduled.

Since the size of the solution space increases exponentially with the number of trips that need to be rescheduled, only small-scale scenarios can be exhaustively explored because of the incurred computational cost (i.e., the solution space when rescheduling only eight trips contains \(61^8 = 1.9E + 14\) alternative combinations).

![Image of the mathematical formulation](https://example.com/math-formulation.png)

**3. Rolling-horizon rescheduling with SHC**

Instead of exploring exhaustively the solution space of the rescheduling problem at each rolling horizon optimization instance, we use a SHC meta-heuristic to obtain a (potentially good) rescheduling solution in near real-time by exploring only a targeted portion of the entire solution space.

In a first step, a direction for the solution search is provided by assigning a higher priority to the satisfaction of constraints. This is achieved by introducing a penalty function which approximates the constrained rescheduling optimization problem with an unconstrained one.

Penalty function methods started to gain attention in the late 1960s (Fiacco & McCormick, 1968). These methods add a term to the objective function that introduces a high cost when constraints are violated (Bertsekas, 1990). In this work, we use exterior point penalization; nevertheless, there are also several interior point penalization methods in literature with most notable the work of Karmarkar (1984) which introduces a barrier against leaving the feasible region.

In this study, we use exterior point penalization because interior penalty functions require to start the solution search from an initial solution guess that satisfies all constraints (feasible solution guess) and finding a feasible initial solution is not a trivial task for multi-variate optimization problems. When using an exterior point penalty function, each constraint penalizes the objective function by a positive value which increases progressively its penalization as described in the following unconstrained optimization problem of Equation (16) that approximates the constrained one of Equation (15).
where $c_i(x)$, $\forall i \in \{1, \ldots, |N| + |N||S|\}$ are the inequality constraints related to the layover times and the vehicle capacities. In the penalty function $p(x)$, the terms $W_i \gg 0$ are positive numbers and have sufficiently high values for ensuring that the satisfaction of the layover and the vehicle capacity constraints has a higher priority than the reduction of the excessive passenger waiting times.

The penalty function, $p(x)$, is formulated in such way that:

- is equal to the objective function value, $f(x)$, if all constraints are satisfied (the penalty function value of a feasible solution is equal to its objective function);
- does not penalize constraints that are satisfied. I.e., if for a solution $x$ one constraint $c_i(x)$ is not violated, then $c_i(x) \geq 0$ and, as a result, $\min(0, c_i(x)) = 0$;
- penalizes progressively the violation of constraints since the violation of a constraint $c_i(x) < 0$ is squared: $W_i(\min(0, c_i(x)))^2 = W_i c_i^2(x)$.

In the SHC search, the dispatching time modifications of the bus trips that are not dispatched yet are initialized as $x_i = 0$, $\forall i \in n \subseteq N$. After that, the initial solution $x$ is evaluated by calculating the penalty function value $p(x)$. Hill climbing attempts to minimize the target function $p(x)$ by adjusting a single element of $x$ at each step and determining whether the change improves the penalty function score. This differs from gradient descent methods, which adjust all the values in $x$ at each step according to the gradient of the hill. Note that in our case we cannot apply gradient descent methods because our problem is not continuous and the function $p(x)$ is not differentiable at every point of its domain due to its $\min(0, c_i(x)))^2$ terms.

The proposed SHC search comprises of the following stages:

- an initial trip, $i$, is randomly selected from the set $n$;
- for this trip, we change the value of $x_i$ by selecting a value from the set $Q$. If the selected value reduces the penalty function score this value is adopted; otherwise is discarded. This procedure continues until exploring all the values from set $Q$;
- the previous procedure is performed for all other trips $i \in n \setminus \{i\}$ in a sequential order;
- another trip $i \in n$ is randomly selected and the above-described procedure is repeated until reaching the maximum number of allowed iterations, $\mu_{\text{max}}$.

This procedure can be translated into the following algorithm:

**Algorithm 1**

1: function Sequential Hill Climbing
2: Initialize $x_i = 0$, $\forall i \in n$;
3: while $\mu_{\text{max}}$ is not reached do
4: Select randomly an initial trip $i \in n$;
5: for each $q \in Q$ do
6: Set $u \leftarrow x_i$;
7: Set $x_i \leftarrow q$;
8: if $p(x)$ is not improved by this change then
9: Set $x_i$ back to its previous value: $x_i \leftarrow u$;
10: end if
11: end for
12: for each $i \in n \setminus \{i\}$ do
13: for each $q \in Q$ do
14: Set $u \leftarrow x_i$;
15: Set $x_i \leftarrow q$;
16: if $p(x)$ is not improved by this change then
17: Set $x_i$ back to its previous value: $x_i \leftarrow u$;
18: end if
19: end for
20: end for
21: return optimal solution $x$ of the dispatching time modifications;
22: end function

The hill-climbing search is sequential because the algorithm does not optimize the dispatching times of all remaining trips in one step. Instead, the algorithm starts from one randomly selected trip and searches the dispatching time modification that reduces the penalty function score the most. Then, this procedure is performed for all other trips in a sequential order, where at each sequence only the dispatching time modifications of the examined trip are explored. In the proposed algorithm, at each iteration another trip is randomly selected for performing a SHC search in a multi-start manner (this random-restart hill climbing search is also known as Shotgun hill climbing and is effective because it spends resources on exploring the solution space, rather than carefully optimizing the initial solution Deb, 2001).

**Lemma 3.1.** The number of computations of the sequential hill climbing search increases polynomially with the number of allowed iterations, $\mu_{\text{max}}$, and the number of remaining trips, $n$, that need to be rescheduled.

**Proof.** In the algorithmic description of the sequential hill climbing algorithm, each random-restart hill-climbing iteration contains two nested for loops that
perform a fixed number of $|n| \cdot |Q|$ calculations. With these calculations, the dispatching time modifications that improved the performance of each trip are computed in a sequential order. Then, the same operation is performed $\mu^{\text{max}}$ times by restarting the sequential search from another randomly selected trip; thus, requiring a total amount of $|n|\cdot |Q|\cdot \mu^{\text{max}}$ computations that increase polynomially with the size of the remaining trips, $n$, and the number of allowed iterations, $\mu^{\text{max}}$.

The opportunistic exploration of the combinatorial solution space with the random-restart sequential hill climbing (SHC) search is scalable due to its polynomial complexity and can find an improved solution to the bus rescheduling problem in near real-time even for high-frequency bus lines with multiple stops. In Figure 2, we present the computational requirements of the SHC search for rolling horizon rescheduling instances with different numbers of remaining trips and different optimization termination, $\mu^{\text{max}}$, thresholds. Note that a maximum number of 400 remaining trips is adopted because, in general, a high-frequency bus line with a frequency of 20 trips/h is not expected to operate more than 400 trips during the daily operational hours.

In the next section, the effect of rolling horizon rescheduling to the regularity of the daily bus operations is tested using GTFS-realtime data from bus line 15 L in Denver, USA.

4. Numerical experiments

4.1. Case study description

In the case study, we use GTFS-RT data from bus line 15 L in Denver, USA to (i) measure how much the actual EWTs of passengers differ from the planned ones; and (ii) compute the potential gain in service regularity if the dispatching times of buses are periodically rescheduled.

Bus line 15 L (Figure 3) is a bus line with 42 bus stops that connects the east part of the Denver metropolitan area with a main train station at the south west (Aurora Metro Center). This is a high-frequency line and serves the downtown area of the city. The first trip of the day is planned at 6:02 am and the last trip of the day is expected to arrive at the terminal around 1:03 am (for the planned arrival times at selected stops, the interested reader can refer to: http://www3.rtd-denver.com/schedules/getSchedule.action?routeId=15L). As a high-frequency service, we assume that passengers do not coordinate the timing of their arrivals with the arrival times of buses at stops (Bartholdi & Eisenstein, 2012; Welding, 1957). Despite this assumption, we note that the transport authority in Denver offers real-time information about the positions of buses and updates their arrival time predictions at least every 30 s. Consequently, passengers that have access to real-time information on their mobile phones might want to coordinate their arrival times at stops to the EATs of buses even in this high frequency line.

Starting from the beginning of the day (6 am), the GTFS-RT data from the running vehicles is processed in time intervals of 15 min. The newly generated GTFS-RT data during each time interval is used to update the AATs of buses at stops. It also updates the matrix $D$ that combined the actual and EATs of bus trips at stops. The updated matrix $D$ is used for re-computing new dispatching times for the remaining trips of the day at each rolling horizon (i.e. end of the 15-min interval). Thus, the total number of rolling horizon rescheduling instances from the beginning
until the end of the day (6:02 am–1:03 am) is 78 and the initial daily schedule of bus line 15 L is updated 78 times.

After analyzing the GTFS-RT data, one of the findings was that during the weekdays many planned bus trips are canceled. For this reason, we selected Saturday for the numerical experiments because we observed that on Saturdays almost all planned trips are operated without cancelations. This allows us to have a less biased comparison between the scheduled waiting times of passengers and the actual ones. On Saturdays, the scheduled number of operational trips from 6:02 am to 1:03 am is 61. To calculate the EATs of future bus trips to stops, we processed 4 months of historical AVL data from Saturday operations and estimated the travel times of trips between consecutive bus stops (Figure 4).

From the 42 bus stops of the examined bus line, only a subset of the 16 most significant bus stops is considered for the calculation of the passengers’ EWTs from the bus operator (namely, the stops 1, 4, 5, 8, 9, 12, 14, 15, 16, 19, 20, 24, 26, 29, 36, and 42). Consequently, we set \( \tilde{w}_j = 1 \) for the 16 bus stops of the above set and \( \tilde{w}_j = 0 \) for all other stops. Those 16 bus stops are the “control points” of the operations and are selected because they are either intersection stops or have high passenger boarding numbers, thus it is important to maintain a high level of regularity.

Initially, we use the GTFS-RT data from Saturday, July 22, 2017 to compare the AATs of bus trips at the 16 control point stops against the scheduled ones and derive the average EWT per passenger. On that day operated 60 trips out of the 61. The results of calculating the average EWT per passenger are summarized in Table 1. Note that the reported average daily passenger EWT (1.947 min) is calculated according to Equation (5) and is the KPI of the service regularity. Note also that plugging in the values of \( \tilde{w}_j \) in Equation (5) we get \( \frac{1}{\sum \tilde{w}_j} = 1/16 \).

This significant difference between the actual and the scheduled passenger waiting times occurred mainly because the (i) traffic conditions change during the day leading to significant differences between the expected and the actual trip travel times as concluded also from many works in the literature (Chang et al., 2010; Gurmu & Fan, 2014; Yan, Liu, Meng, & Jiang, 2013) and the (ii) dwell times of bus trips which are behind schedule increase creating a domino effect that leads to bus bunching.

For this reason, it is important to reschedule the dispatching times of the remaining daily trips periodically, as it is demonstrated in Subsection 4.4.

4.2. Computational tests and solution quality

In this subsection, we study the performance of the proposed SHC algorithm in terms of convergence (optimality gap) and computational costs when rescheduling the trips in our previously described case study.

With respect to the optimality gap tests, the solution of the SHC should be compared against a globally optimal solution. Our rolling-horizon rescheduling problem is NP-Hard, therefore a globally optimal solution can be computed only at small-scale problem instances. For this reason, the optimality gap of the SHC method is studied at small-scale scenarios that permit the computation of a globally optimal solution with brute force.

Regarding the computational cost tests, we compare the computational time of our algorithm against another common solution method for combinatorial optimization: the genetic algorithm. Even though both SHC and the genetic algorithm cannot guarantee the convergence to a globally optimal solution, their solution performance can be compared in larger-scale scenarios providing an indication of the effectiveness of our approach.

The results of the computational tests are summarized in Table 2. For a detailed explanation of the genetic algorithm used in these tests, the interested reader can refer to the Appendix. Note that all tests are conducted in a 2.40 GHz, quad-core processor with a 16GB RAM. The SHC algorithm is programmed in Python 2.7 and the genetic algorithm is implemented with the use of the Distributed Evolutionary Algorithms in Python (Deap) package Fortin et al. (2012) in Python 2.7. Note that the hyperparameters
Table 1. Performance of the daily operations of bus line 15L (Saturday, July 22nd, 2017).

<table>
<thead>
<tr>
<th>Bus stop</th>
<th>Average passenger waiting time (min)</th>
<th>Scheduled passenger waiting time (min)</th>
<th>Bus stop</th>
<th>Average passenger waiting time (min)</th>
<th>Scheduled passenger waiting time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>≈ 12.66</td>
<td>10.61</td>
<td>16</td>
<td>≈ 12.87</td>
<td>10.67</td>
</tr>
<tr>
<td>4</td>
<td>≈ 12.13</td>
<td>10.61</td>
<td>19</td>
<td>≈ 12.80</td>
<td>10.67</td>
</tr>
<tr>
<td>5</td>
<td>≈ 11.86</td>
<td>10.61</td>
<td>20</td>
<td>≈ 12.79</td>
<td>10.67</td>
</tr>
<tr>
<td>8</td>
<td>≈ 11.91</td>
<td>10.62</td>
<td>24</td>
<td>≈ 12.83</td>
<td>10.66</td>
</tr>
<tr>
<td>9</td>
<td>≈ 12.01</td>
<td>10.62</td>
<td>26</td>
<td>≈ 12.86</td>
<td>10.66</td>
</tr>
<tr>
<td>12</td>
<td>≈ 12.72</td>
<td>10.63</td>
<td>29</td>
<td>≈ 12.90</td>
<td>10.66</td>
</tr>
<tr>
<td>14</td>
<td>≈ 12.74</td>
<td>10.65</td>
<td>36</td>
<td>≈ 12.85</td>
<td>10.68</td>
</tr>
<tr>
<td>15</td>
<td>≈ 12.74</td>
<td>10.66</td>
<td>42</td>
<td>≈ 12.78</td>
<td>10.67</td>
</tr>
</tbody>
</table>

Daily passenger Excess Waiting Time: 1.947 min

Table 2. Performance of different solution methods in idealized scenarios.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>No. of bus stops</th>
<th>Brute force</th>
<th>Sequential hill climbing</th>
<th>Genetic algorithm</th>
<th>Optimized p(x) score (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. of trips</td>
<td>Brute force</td>
<td>Sequential hill climbing</td>
<td>Genetic algorithm</td>
<td>Brute force</td>
<td>Sequential hill climbing</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>3.83 sec</td>
<td>0.22 s</td>
<td>1.38 s</td>
<td>62.62</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>8 min</td>
<td>0.35 s</td>
<td>2.01 s</td>
<td>58.98</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>8 h 12 min</td>
<td>0.72 s</td>
<td>3.33 s</td>
<td>58.41</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>—</td>
<td>3.12 s</td>
<td>5.48 s</td>
<td>—</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>—</td>
<td>6.78 s</td>
<td>21.24 s</td>
<td>—</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>—</td>
<td>16.00 s</td>
<td>49.73 s</td>
<td>—</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>—</td>
<td>27.64 s</td>
<td>101.25 s</td>
<td>—</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>—</td>
<td>44.42 s</td>
<td>246.53 s</td>
<td>—</td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>—</td>
<td>62.98 s</td>
<td>284.32 s</td>
<td>—</td>
</tr>
</tbody>
</table>

of the genetic algorithm (namely, the population size, the mutation rate and the maximum population generations) have been calibrated for each scenario to guarantee that it will attain its best performance when compared against the SHC approach. Finally, the brute force approach is also programmed in Python 2.7 and evaluates the performance of the objective function for each possible combination of rescheduled departure times.

To perform the computational and solution quality tests, we reschedule the scenarios in Table 2 (ranging from 2 to 60 trips). Note that if we reschedule 2 trips, those are the last 2 of the day, if we reschedule 3 those are the last 3 and so forth. The upper rescheduling limit is 60 trips that contains the entirety of daily trips of our case study. For each scenario, we report the computational time of the brute force method, the SHC and the genetic algorithm. With the use of brute force, we can solve up to scenarios with four bus trips with a computational time of 8h and 12 min. That is, the globally optimal solution can be established when rescheduling up to four trips. In addition, in the last three columns of Table 2 we report the optimized value of the objective function, $p(x)$ presented in Equation (16).

From Table 2, one can note that the optimized objective function value with the SHC approach is equal to that one with the brute force approach for scenarios with up to four bus trips. That is, SHC converges to a globally optimal solution and there is no optimality gap. For more than four trips, that point cannot be proved with absolute certainty due to the absence of a reliable benchmark. In scenarios with a higher number of trips, we note that SHC performs better than the genetic algorithm when rescheduling 20 trips or more. Particularly, in the rescheduling case of 60 trips, SHC results in an improved solution by 65% compared to the genetic algorithm. As it is presented in Figure 5, the main reason is that the population of the genetic algorithm is not improving any further after the 30th population generation. In contrast, SHC exhibits a rapid convergence and terminates with a $p(x)$ score of 6.37 s after only six iterations.

In terms of computational costs, SHC can reschedule 60 trips (which can be the entirety of the daily trips) in $\approx$ 1 minute using a conventional computing machine. This proves that we can reschedule a real-sized bus line in near real-time. Additionally, the SHC is consistently at least two times faster than the genetic algorithm. All in all, both approaches have a similar performance when rescheduling up to 20 trips. After that limit, SHC performs better both in terms of convergence and computational costs.

4.3. Sensitivity analysis of the EWT with respect to the range of dispatching time modifications, $Q$

In our case study, we allow the dispatching time modifications, $x$, to receive values from the set $Q = \{-30, -29, ..., 0, ..., +29, +30\}$ min at each rescheduling optimization instance. Nevertheless, bus
operators might not be willing to modify the dispatching times that much from their scheduled values, even if this is beneficial for the service regularity.

To investigate the influence of a restricted range of dispatching time modification options, we perform again the experiments of the previous section. We reschedule all 60 trips in Table 2 with the use of SHC and we alter every time the range of the dispatching time modification options, $Q$. The results are summarized in Table 3. The ranges tested in our evaluation are reported at the first column of Table 3. For each range of dispatching time modification options, we compute the optimized value of our objective function which is reported at the second column of the table.

![Figure 5. Convergence of the sequential hill climbing and the genetic algorithm when rescheduling 60 trips of bus line 15 L.](image)

### Table 3. Sensitivity of solution performance to restrictions on dispatching time modifications, $Q$.  

<table>
<thead>
<tr>
<th>Range of $Q$</th>
<th>Optimized $p(x)$ score (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[-30 \text{ min}, +30 \text{ min}]$</td>
<td>6.37 s</td>
</tr>
<tr>
<td>$[-25 \text{ min}, +25 \text{ min}]$</td>
<td>6.58 s</td>
</tr>
<tr>
<td>$[-20 \text{ min}, +20 \text{ min}]$</td>
<td>7.30 s</td>
</tr>
<tr>
<td>$[-15 \text{ min}, +15 \text{ min}]$</td>
<td>8.03 s</td>
</tr>
<tr>
<td>$[-10 \text{ min}, +10 \text{ min}]$</td>
<td>8.21 s</td>
</tr>
<tr>
<td>$[-7 \text{ min}, +7 \text{ min}]$</td>
<td>17.03 s</td>
</tr>
<tr>
<td>$[-5 \text{ min}, +5 \text{ min}]$</td>
<td>34.54 s</td>
</tr>
<tr>
<td>$[-4 \text{ min}, +4 \text{ min}]$</td>
<td>38.86 s</td>
</tr>
<tr>
<td>$[-3 \text{ min}, +3 \text{ min}]$</td>
<td>51.71 s</td>
</tr>
<tr>
<td>$[-2 \text{ min}, +2 \text{ min}]$</td>
<td>68.03 s</td>
</tr>
</tbody>
</table>

Contrary to the above, if we are only allowed to modify the dispatching times of trips by less than 10 min, then the potential benefit follows a sharp reduction. For modifications in the range of $\pm 5$ min, the potential gain shrinks resulting in an EWT of 38.86 s. This sharp deterioration of the solution quality continues and escalates for modification ranges of $\pm 2$ min. Consequently, a potential recommendation is to permit, when possible, dispatching time modifications in the range of 10 min.

### 4.4. Application of the periodic rescheduling

Splitting the operating time of the day into 15-min periods, we have 78 rolling horizons. At each rolling horizon instance, the dispatching times of the remaining daily trips are re-computed based on the new information regarding the AATs of buses at stops (GTFS-RT data). At each rolling-horizon instance, the observed arrival times of buses at stops are used together with the EATs of the remaining trips of the day for forming the matrix $D$ and updating the dispatching times.

As the rolling horizons increase ($\rightarrow 78$) the rescheduling affects the dispatching times of a smaller fraction of trips because:

- several trips have already been dispatched; hence their dispatching time cannot be rescheduled.
- after several periodic optimizations, the degrees of freedom (i.e. rescheduling options) are limited and the operations cannot be improved further.

The rescheduling options at each rolling horizon are computed with the aim of minimizing the daily average EWT of passengers. Note that at a rolling...
The average daily EWT of passengers comprises the actual EWT (for trips that have already served the bus stops) and the expected EWT (for future trips for which we estimate their arrival times at stops). Hence, the daily average EWT evolves at every rolling horizon instance and stabilizes at the end of the day (rolling horizon 78) where all trips have completed their services and the value of the daily average EWT of passengers is definitive. The evolution of the daily average EWT at each periodic optimization instance is presented in Figure 7.

At the beginning of the day, the expected daily average EWT is close to zero. This means that the rescheduling will lead to high regularity levels if the AATs of future trips at stops are identical to the expected ones (something that is not the case in practice). However, this expectation changes after ~15 rolling horizons (at 9:47am) because the observed arrival times of buses differ considerably from the expected ones and the bus operations should adjust accordingly. Another important observation is that the expected daily average EWT at 00:02 exhibits a higher value than the observed one at the end of the day. This might occur in some occasions if the actual travel times of the remaining trips are more favorable than the expected ones leading to more even time headway distributions for the remaining trips.
5. Discussion and sensitivity analysis of performance under travel time noise

The observed EWT of passengers on Saturday, July 22nd was 1.947 min (cf. Table 1). With the use of periodic rescheduling on time intervals of 15 min, the daily EWT was reduced to 0.92 min (cf. Figure 7). This regularity improvement of \( \approx 50\% \) refers to that particular day. An interesting topic for further investigation though is the importance of the periodic rescheduling for different levels of travel time noise.

This is further examined by replicating the arrival times of buses at stops at the Simulator of Urban MOBility (SUMO). To this end, we import the road network of Denver in SUMO using OpenStreetMaps. Then, we add the stop locations of bus line 15 L (see Figure 8). To replicate the arrival times of buses at stops, we calibrate the traffic flow-related parameters with the use of the Constrained Optimization BY Linear Approximation (COBYLA) algorithm from the SciPy library in Python.

COBYLA is a standard optimization algorithm which is used in SUMO to find the optimal parameter values of traffic flows and traffic signal cycles by minimizing the root mean square error (RMSE) between the observed and simulated arrival times of buses at stops (a well-known benchmark calibration with COBYLA is in Smilowitz, Daganzo, Cassidy, & Bertini, 1999). COBYLA terminates when an acceptable RMSE error is achieved indicating that the calibrated traffic scenario results in simulated arrival times at stops which are too close to the observed ones from the GTFS-RT data.

Apart from the traffic-related parameters, the vehicle parameters of the running buses (i.e., acceleration, max. speed, deceleration) are communicated to SUMO using a standard.xml file that follows the SUMO guidelines. Based on historical data from bus line 15 L, their values are set as follows: acceleration—2.61 m/s\(^2\), deceleration—4.68 m/s\(^2\), max. speed—62 miles/h. The RMSE error of our best performing traffic-related parameter values was 4.2%.

By using a traffic simulator, we investigate the performance of the periodic rescheduling in different scenarios considering the effect of travel time noise. The examined scenarios are summarized in Table 4. Each travel time noise scenario assumes that the actual link travel times follow a normal distribution where the mean value of each normal distribution is equal to the respective value of the expected link travel time.

The first scenario considers a standard deviation \( \sigma_{i,j} \) of the normal distribution equal to 10\% of the expected link travel time \( t_{i,j} \) for each trip \( i \in N \) and stop \( j \in S \setminus \{\|S\|\} \). The second and the third scenarios consider a standard deviation equal to 20 and 30\% of the mean value of the normal distribution. Finally, the fourth scenario samples the link travel time values from a normal distribution with standard deviation equal to 40\% of the mean value of the distribution. To reduce the sampling bias, at each one of those four scenarios we run 1000 simulations where at each simulation the link travel time noise values are sampled from the corresponding normal distribution.

The performance of the periodic rescheduling in the presence of different travel time noise levels is compared against the performance of the do-nothing case where the planned schedule is not updated during the day.
Results are summarized in Table 5 where the Tukey boxplot convention is utilized (plotting the lowest datum still within 1.5 the interquartile range (IQR) of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile). Table 5 demonstrates the improvement when using periodic rescheduling in the presence of different link travel time disturbances.

Testing the performance of the periodic rescheduling when the standard deviation is 10% of the expected value of the corresponding link travel time, one can observe that the mean benefit is reduced to 19.5%. Simulations show that the significance of periodic rescheduling increases when the variation of the actual link travel times from their expected values increases (for a standard deviation of 20% the EWT improvement becomes 27.9% and for 30% becomes 40.4%). Specific attention is given in the case of strong travel time disturbances (where the standard deviation of the normal distribution from which the travel times are sampled is 40% of the expected link travel times). This case is plotted in Figure 9 and shows a potential EWT improvement of 48.2%.

Finally, we report in our discussion that during the daily operations one of the scheduled trips might be canceled. Given the time frame of our optimization though, any trip cancelation can be accounted for in the next rolling horizon. This can be achieved in our model by re-indexing the trips in the mathematical program of Equation (16) prior to applying our solution method.

### Table 5. Daily excess waiting time (min).

<table>
<thead>
<tr>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>Q1</td>
<td>Median</td>
<td>Q3</td>
</tr>
<tr>
<td>Follow planned schedule</td>
<td>0.53</td>
<td>0.44</td>
<td>0.41</td>
</tr>
<tr>
<td>10% noise</td>
<td>Periodic rescheduling</td>
<td>0.44</td>
<td>0.34</td>
</tr>
<tr>
<td>Follow planned schedule</td>
<td>0.91</td>
<td>0.73</td>
<td>0.61</td>
</tr>
<tr>
<td>Periodic rescheduling</td>
<td>0.77</td>
<td>0.54</td>
<td>0.44</td>
</tr>
<tr>
<td>Follow planned schedule</td>
<td>1.94</td>
<td>1.57</td>
<td>1.36</td>
</tr>
<tr>
<td>Periodic rescheduling</td>
<td>1.16</td>
<td>0.97</td>
<td>0.81</td>
</tr>
<tr>
<td>Follow planned schedule</td>
<td>3.21</td>
<td>2.23</td>
<td>1.89</td>
</tr>
<tr>
<td>Periodic rescheduling</td>
<td>1.50</td>
<td>1.28</td>
<td>0.98</td>
</tr>
</tbody>
</table>

**Figure 9.** Tukey boxplot of the performance of the do-nothing scenario that follows the originally planned timetable and the periodic rescheduling in 15-min intervals.

6. **Concluding remarks**

This article models the bus rescheduling problem in rolling horizons considering the vehicle capacity and the layover constraints. After the formulation of the periodic rescheduling problem as an integer nonlinear programing problem, it is proved that the problem complexity is exponential. This impedes the solution of the problem with analytical methods. Due to that, the constrained optimization problem is approximated with the use of exterior point penalties and a SHC solution method is developed to minimize the value of the resulting penalty function by exploring more efficiently the vast solution space.

This heuristic search method reduces the exponential computational complexity to polynomial and enables the rescheduling of the dispatching times of
the remaining daily trips of a bus line at each rolling horizon instance. Applying the periodic rescheduling to bus line 15L in Denver demonstrated a \( \approx 50\% \) potential improvement of the daily average EWTs of passengers. This service regularity improvement is scenario-specific. For this reason, we study the sensitivity of the regularity improvement with periodic rescheduling for different levels of travel time disturbances. The results of the simulation-based sensitivity analysis show that if the daily operations are close to normal, periodic optimization offers a \( \approx 20\% \) margin of improvement. In contrary, if the AATs of buses at stops differ significantly from the expected ones, periodic optimization can offer higher gains in the range of 40–50%.

One important topic for future research is the consideration of multiple KPIs during the periodic rescheduling, such as the bus occupancy levels and the unsatisfied demand. Especially nowadays, the availability of smart-card data can be used to support this research topic and expand the EWT KPI by offering a more holistic view that considers both the service regularity and the passenger demand satisfaction as suggested by Randall, Condry, Trompet, and Campus (2007). Additionally, future research can investigate further the coordination between passenger arrival times and bus arrival times in high-frequency services because passengers that have access to real-time information on their mobile phones might try to synchronize the timing of their arrivals even if their expected waiting times are low.

Notes

The processed GTFS-RT data of bus line 15L in Denver is publicly released at http://doi.org/10.4121/uuid:c4ab5a8a-a834-4b47-b700-16b38cab226. The release contains the actual arrival times of all daily trips on the 22nd of July, 2017 which were used at the numerical experiments section.

ORCID

Konstantinos Gkiotsalitis http://orcid.org/0000-0002-3009-1527

References


Appendix A. Genetic algorithm description

A typical genetic algorithm (GA) contains a number of strings which form the population at each of the iterations. Each string is a population member (individual) and represents a potential solution to the optimization problem. In our case, each population member \( \mathbf{m} \) is a \([N]\)-valued vector of potential values of the decision variable \( \{x_1, x_2, ..., x_{[N]}\} \) that determine the dispatching time modifications of all daily trips.

At the initialization stage of the GA, we should decide about the population size, \( |P| \), which is a hyperparameter of the GA. This parameter can be determined based on the tradeoff between solution space exploration and computational cost since a GA with a larger population size is expected to conduct a more comprehensive exploration of the solution space but requires also more time to evaluate all possible solutions.

Each population member, \( \mathbf{m} \in P \), is a vector \( \mathbf{m} = (m_1, ..., m_i, ..., m_{[N]}) \) with \([N]\) elements (known as genes) where each element \( m_i \in \mathbf{m} \) represents the dispatching deviation of the \( i \)-th bus trip from its scheduled dispatching time. Each gene \( m_i \in \mathbf{m} \) of an individual \( \mathbf{m} \) is allowed to take an integer value from the set \( Q \) or is set equal to zero if bus trip \( i \in N \) is already dispatched when the current re-scheduling occurs.

Therefore, a random initial population \( P \) can be generated as follows:

For \( \mathbf{m} = 1 \) to \( |P| \)
For \( i = 1 \) to \([N]\)  
If \( a_i \neq 0 : m_i \leftarrow \text{random.choice}(Q) \)  
If \( a_i = 0 : m_i \leftarrow 0 \)  
Next \( i \)  
Next \( \mathbf{m} \)

GAs are typically designed to maximize fitness. Notwithstanding this, given the fact that our problem of Equation (16) is casted as a minimization problem, in our study a population member \( \mathbf{m} \) is considered more fit when its fitness function value, \( p(\mathbf{m}) \), is lower.

In the parent selection stage, the fittest population members (individuals) are selected for reproduction and they pass their genes to the next generation. This is achieved by using the well-known roulette-wheel selection method (Goldberg & Deb, 1991). In the roulette-wheel selection method, each individual \( \mathbf{m} \) has a probability of being selected which is proportional to its fitness value divided by the fitness values of all other population members. The parent selection process is repeated until the number of parents which are selected for reproduction is the same as the population size \( |P| \).

At the crossover stage, two parents exchange their genes at a randomly selected crossover point selected from the set \( \{1, 2, ..., [N]\} \) for generating two offsprings. For instance, if the crossover point of two parents \( \mathbf{m} = (m_1, ..., m_i, m_{i+1}, ..., m_{[N]}) \) and \( \mathbf{m}' = (m'_1, ..., m'_i, m'_{i+1}, ..., m'_{[N]}) \) is \( i \in N \); then, the two generated offsprings will have the set of genes \( (m_1, ..., m_i, m'_{i+1}, ..., m'_{[N]}) \) and \( (m'_1, ..., m'_i, m_{i+1}, ..., m_{[N]}) \).

After the crossover stage follows the mutation stage. The mutation can be potentially applied to any generated offspring after the crossover stage. In our case, we specify a small probability, \( p_c \), for replacing each gene of the generated offspring with a random value from the set \( Q \).

The procedure described above continues iteratively until a pre-determined number of population generations, \( \mu_{\text{max}} \), is reached. The population member with the best performance is then selected as the final solution.

Note that the number of population members, \( |P| \), the mutation rate, \( p_c \), and the maximum number of population generations, \( \mu_{\text{max}} \), are hyperparameters of the GA which should be externally defined and can affect the performance of the computed solution. For this reason, finding the most suitable hyperparameter values requires to optimize the same scenario multiple times with different hyperparameter values at each optimization.