Unsupervised Mechanisms for Optimizing On-Time Performance of Fixed Schedule Transit Vehicles

Fangzhou Sun, Chinmaya Samal, Jules White, Abhishek Dubey
Institute of Software Integrated Systems, Vanderbilt University, Nashville, TN, USA
{fangzhou.sun, chinmaya.samal.1, jules.white, abhishek.dubey}@vanderbilt.edu

Abstract—**The on-time arrival performance of vehicles at stops is a critical metric for both riders and city planners to evaluate the reliability of a transit system. However, it is a non-trivial task for transit agencies to adjust the existing bus schedule to optimize the on-time performance for the future. For example, severe weather conditions and special events in the city could slow down traffic and cause bus delay. Furthermore, the delay of previous trips may affect the initial departure time of consecutive trips and generate accumulated delay. In this paper, we formulate the problem as a single-objective optimization task with constraints and propose a greedy algorithm and a genetic algorithm to generate bus schedules at timepoints that improve the bus on-time performance at timepoints which is indicated by whether the arrival delay is within the desired range. We use the Nashville bus system as a case study and simulate the optimization performance using historical data. The comparative analysis of the results identifies that delay patterns change over time and reveals the efficiency of the greedy and genetic algorithms.**

Keywords*—*public transportation; optimal scheduling; statistical distributions; genetic algorithm;

I. Introduction

Emerging trends and challenges. In the last decade, public transit ridership in the United States increased by 37% [1]. Compared with other modes of transportation like subway and light rail, bus service has advantages of low cost and large capacity, and thus is the backbone of public transit services in many cities. However, bus operations are also more easily affected by uncertain factors, such as traffic congestion, weather condition, road construction, passenger/bicycle loading, big events, etc. If the same vehicle or operator is scheduled to be used by two consecutive bus trips, the accumulated delay occurred on previous trips may cause a delay in consecutive trips by affecting the initial departure time of the next trip. This unreliability in bus services can decrease rider satisfaction and loyalty, resulting in lower fleet utilization [2]. Regarding this issue, studies [3], [4], [5] have been conducted to reduce the uncertainty in transit systems and provide predictive information to users.

Providing convenient, efficient and sufficient bus services to meet the expanding demand and reliability requirement for public transit remains a great challenge for transit agencies. Therefore, transit agencies have developed various indicators to evaluate public transit systems and monitored service reliability through several key performance measurements from different perspectives [6]. Common indicators of public transit system evaluation include schedule adherence, on-time performance, total trip travel time, etc. To quantity bus on-time arrival performance, many regional transit agencies use the range of [-1,+5] min compared to the scheduled bus stop time as the on-time standard to evaluate bus performance using historical data [7]. They analyze the historical data and adjust the scheduled time by hand, which is time consuming and heavily relies on the past experience of transit engineers.

A variety of studies have been conducted on improving bus on-time performance and many use heuristics solutions. Specific and ad-hoc heuristic search (e.g. greedy algorithms), neighborhood search (e.g. simulated annealing (SA) and tabu search (TS)), evolutionary search (e.g. genetic algorithm [8], [9], [10], [11]) and hybrid search [12], [13] are popular methods to search for the optimization solutions. The optimization objectives of existing works also vary, such as minimizing passenger transfer time (required time to switch from one route to another route to get to destinations) [11], minimizing transfer user cost [14], bus frequency setting (the frequency of departure buses of one route) [12]. However, there are few stochastic optimization models that focus on optimizing bus timetables to maximize the probability of bus trips where buses arrive at timepoint with delay within a desired on-time range (e.g. one minute early and five minutes late), which is widely used as a key indicator of bus performance in the United States [7]. Timepoints are special bus stops that transit agencies use to record and coordinate the bus arrival times along a trip. Studying the travel time of timepoint segments can be an effective way to set bus timetables, however, because of the monthly and seasonal variation in historical monthly patterns, generating one timetable for all months may not be the best solution, and how to divide months into clusters and optimize timetable for each month cluster remains an open problem.

Contributions. This paper focuses on creating and implementing a mechanism to improve the on-time performance of bus services with fixed schedules at the re-planning stage (re-planning stage is when transit agencies adjust the existing bus schedules to make a future timetable). Specific contributions are 1) We describe an unsupervised mechanism to find out how months can be divided to generate new timetables. We apply outlier analysis and clustering analysis on bus travel times to identify monthly patterns, and then generates new timetables for month clusters that have similar patterns. The feature vectors we use include mean, median and standard deviation of the historical travel time aggregated by route, trip, direction, timepoint segment and month. 2) We present a genetic algorithm to optimize the scheduled arrival and departure time at timepoints to maximize the probability of bus trips that reach the desired on-time range. A greedy algorithm is also developed for comparison purpose. 3) We evaluate the proposed mechanism via simulation. Results show that the genetic algorithm outperforms the greedy algorithm in on-time performance and the month grouping method that generates separate bus schedules for clustered months can further improve the optimization. The average on-time performance on all bus routes was improved from 62.9% to 74.7%.

Paper outline. Section II compares our mechanism with related work; Section III presents the problem description and formulation, and key research challenges; Section IV outlines the details of the unsupervised mechanism; Section V uses real-world data from the Nashville transit system as a case study to evaluate our methodology's performance; Section VI presents conclusion remarks and future work.

II. Related Work

A wide range of studies have been conducted on the bus on-time performance optimization problem. Friedman et al. [15] formulated a mathematical model of a general transportation network and presented a procedure to optimize bus departure times for minimizing the average waiting time of passengers by changing decision variables (i.e. bus departure time). Hora et al. [13] applied a Mixed Integer Linear Programming (MILP) model to obtain robust bus schedules that minimize the differences between scheduled times and actual arrival time Their solution works on allocating the slack time of two subsequent stops. Guihaire et al. [16] presented a classification of 69 approaches dealing with route design, bus frequency and timetabling.

Genetic algorithms (GAs) are search and optimization methods based on the evolutionary ideas of natural selection. Chakroborty et al. [17] first used genetic algorithms to develop optimal schedules for urban transit systems. The problem is formulated as a mathematical program that minimizes the sum of total time transferring from one route to another route for all transferring passengers and initial waiting time for all passengers at the origin. Later, Pattnaik et al. [8] proposed a genetic algorithm for designing urban bus transit route network. Their research focuses on selecting a set of optimum route sets using a GA. Charkroborty et al. [9] developed genetic algorithm based procedures for route planning and scheduling. Zhao et al. [14] presented a mathematical stochastic methodology to minimize transfer and user cost. Yang et al. [10] proposed an improved genetic algorithm to optimize timetables that passenger transfer time is minimized using constraints of traffic demand and departure time and maximum headway.

Naumann et al. [18] presented a stochastic programming approach for robust vehicle scheduling in public bus transportation. Szeto et al. [12] proposed a genetic algorithm for route design problem and a neighborhood search heuristic for bus frequency setting problem. Their goal is to reduce the number of transfers and the total travel time of the users. Tilahun et al. [19] modeled single frequency route bus timetabling as a fuzzy multi-objective optimization problem

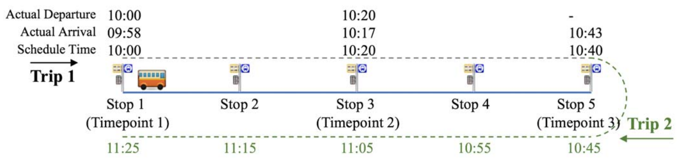

Fig. 1. A bus route example: two consecutive trips in the same block.

using preference-based genetic algorithm. Nayeem et al. [11] presented a genetic algorithm based optimization model for maximizing the number of satisfied passengers, minimizing the total number of transfers and minimizing the total travel time of all served passengers.

Using genetic algorithms for transit optimization is well studied. However, to the best of the authors' knowledge, even though the on-time arrival range (e.g. one minute earlier and six minute later than advertised schedule) is widely used as a key transit reliability indicator by transit agencies for analyzing and timetabling in the United States [7], there are few stochastic optimization models focusing on optimizing bus timetables to increase bus trips within the on-time performance range. Also, they didn't realize that grouping months according to the travel time patterns and generating cluster-specific schedule can further increase the on-time performance. The difference between existing approaches and our approach is that since traffic and delay patterns change over seasons and different times, we generate clusters based on unsupervised learning and develop optimization models for these clusters. In this paper, we propose an unsupervised mechanism with genetic algorithm to solve this problem. We show how we formulate the problem and set up the solution population for the genetic algorithm. A greedy algorithm is developed as a comparison. We also study the seasonal variations on bus delay patterns, which help to build a robust bus timetable.

III. System Model

Public transit bus service in a city typically consists of multiple routes. Each route contains a set of trips that depart at different times according to a published public timetable. A timepoint is a special transit stop that can accurately record the departure and arrival time of buses [20] (In Figure 1, Stop 1, 3 and 5 are timepoints). Transit agencies use timepoints to coordinate the buses by constraints that (1) a bus should wait at a timepoint until the scheduled time if it arrives early (2) a bus should departure as soon as possible if it arrives on time or late. There are some other key concepts that are involved in the problem:

- **Block:** A *block* consists of a group of sequential trips that use the same vehicle. Transit Authorities divide trips in a day into several *block*s by choosing the first trip and connect it with next trip that leaves from the end of the same line. In Figure 1, Trip 1 and 2 belong to the same block. The bus of Trip 1 will only continue trip 2 after it has arrived at stop 5. Thus, the delay in trip 1 will affect trip 2.
- **Slack Time:** The *slack time* is the layover time between the scheduled arrival time at the last timepoint of a trip and the scheduled departure time at the first timepoint of the next

TABLE I. Notations used in the optimization problem

h	a bus trip that departures at the same time in different days. In Figure 1, there are two bus trips that scheduled to depart at 10:00 and 10:45.
b	a bus schedule that defines the departure and arrival time at bus stops and timepoints
s	a timepoint
$t_{h,s}^{arrival}$	the actual arrival time at a timepoint s on trip h
$t_{h,s}^{departure}$	the actual departure time at a timepoint s on trip h
t_{h,s_i,s_j}^{travel}	the actual travel time between two adjacent timepoints s_i and s_j on trip h
$T_{h,s}^{arrival}$	the scheduled arrival time at timepoint s on trip h
$T_{h,s}^{departure}$	the scheduled departure time at timepoint s on trip h
$[t_{early}, t_{late}]$	the time window that the arrival delay on-time bus should satisfy within
$t_{s_j}^{dwell}$	the dwell time (in simulation) at timepoint s_j that caused by riders getting on/off

trip in the same block. The scheduled layover between Trip 1 and 2 in the example is 5 minutes.

Timepoint Schedule Adherence (TSA) is an important indicator that calculates how often buses adhere to their schedule. TSA is widely used by transit agencies to estimate the historical bus on-time performance on different routes [21]. The main purpose of this study is to create a methodology to improve the on-time performance of a given set of routes by optimizing the scheduled time at timepoints.

A. Problem Formulation

In order to formulate an optimization problem that aims to obtain a timetable that maximizes on-time performance at timepoints, we define the notations in Table I.

Let $H = \{h_1, h_2, ..., h_m\}$ be a set of m historical trips of a given bus trip schedule b. Each trip passes a set of n timepoints $\{s_1, s_2, ..., s_n\}$. The *on-time performance* of the bus trip schedule b can be expressed as :

$$P = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} I(h_i, s_j)}{m \times n} \tag{1}$$

where h_i denotes a historical trip and s_j denotes a timepoint on the trip. The indicator function $I(h_i, s_j)$ is defined as:

$$I(h_i, s_j) = \begin{cases} 1, & \text{if } d_{i,j} \in [t_{early}, t_{late}] \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$d_{i,j} = t_{h_i,s_j}^{arrival} - T_{h_i,s_j}^{arrival} \tag{3}$$

where $d_{i,j}$ is the actual delay that a bus from the historical trip h_i arrives at a timepoint s_j, t_{early} and t_{late} are two time parameters that transit authority has pre-defined to rate the schedule adherence of the bus at that timepoint. The goal of the schedule optimization problem is to generate new $T_{h,s}^{departure}$, such that the on-time performance is maximized. However, any updated schedule must satisfy the following constraints:

- **Constraint 1.** The scheduled slack time between two adjacent bus trips that belong to the same block must be greater than or equal to zero minute i.e. $T_{s_1'} - T_{s_n} \geq 0$, where s_n is the last timepoint of the current trip and s_1' is the first timepoint of the next trip in the same block.
- **Constraint 2.** The actual departure time at a timepoint should be greater than or equal to the scheduled departure time i.e. $t_s^{departure} \geq T_s^{departure}$
- **Constraint 3.** The scheduled departure time at a timepoint should be equal to the scheduled arrival time at the timepoint i.e. $T_s^{arrival} = T_s^{departure}$. How we handle dwell time at timepoints is described in Section IV-E.

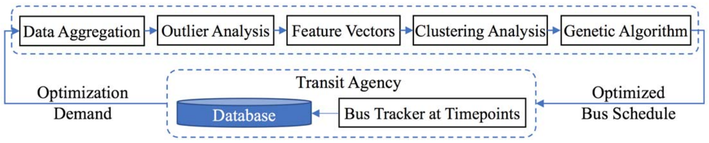

Fig. 2. The overall work flow of the unsupervised bus timetable optimization mechanism.

IV. Methodology

This section describes: (1) a genetic algorithm to solve the optimization problem described in sectionIII-A (details related to solution representation, initialization, evaluation, selection, crossover, mutation and termination are discussed), (2) a greedy algorithm as a comparison to the GA. Working code can be found in our public repository [22].

The overall work flow of the unsupervised bus timetable optimization mechanism is shown in Figure 2. First, outlier analysis is applied to identify and remove the outlier data from historical dataset. Clustering analysis is used to cluster months according to the feature vectors generated for each month. This is important because travel time during different seasons have different patterns as shown in Figure 3, which plots the [mean, standard deviation, median] vector of the monthly travel time for a segment (WE23-MCC5_5) on a bus trip of route 5. It should be noted that we provide an upper bound on the number of clusters as an algorithmic parameter. Setting the upper bound to one will ensure that only one schedule is generated for the whole year.

A. Data Aggregation

We have been collaborating with the Nashville Metropolitan Transit Authority (MTA) to access the bus schedules and real-time bus data feeds in Nashville. Also, we are integrating data from multiple other data sources to collect the real-time traffic and weather data in the city. The data sets that we have integrated into our system are as follows:

- Bus schedule datasets are the static public transportation schedules and associated geographic information of routes, trips, stop times, physical route layout in General Transit Feed Specification (GTFS) format [23] for all the 57 bus routes in Nashville.
- Real-time transit feeds are the real-time updates of transit fleet information in real-time GTFS format [23], including three types of information: (1) trip updates: bus delays and changes, (2) service alerts: routes and buses that are affected by unforeseen events, (3) vehicle position: bus locations with timestamps.

TABLE II. Real-time and Static Datasets Collected in the System.

Bus Schedules		Real-time Transit	
Format	Static GTFS	Format	Real-time GTFS
Source	Nashville MTA	Source	Nashville MTA
Update	Every public release	Update	Every minute
Size	30.6 MB (used version)	Size	411 GB

Timepoints		Real-time Traffic	
Format	Excel	Format	JSON
Source	Nashville MTA	Source	Here API
Update	Every month	Update	Every minute
Size	300,000 entries/month	Size	49.5 GB

- Time-point feed provides the historical bus operating details, including each bus's route, trip and vehicle ID, accurate arrival and departure time at timepoints, etc. Nashville MTA releases the time-point data sets at the end of each month.

B. Outlier Analysis

Median Absolute Deviation (MAD) [24] is a robust measure of statistical dispersion . For a data set $[x_1, x_2, ..., x_n]$, the MAD of the data set can be calculated using the following equation: $MAD = median(|x_i - median(x)|)$ where function $median(X)$ returns the median of data set X. For normal distribution, the scaled MAD is defined as (MAD/0.6745), which is approximately equal to the standard deviation. For any x_i, if the difference between x_i and median is larger than 3 times of standard deviation (i.e. scaled MAD), then we consider, x_i as an outlier.

C. Feature Vector

To cluster the months, a representation of the data distribution in each month is needed. For a bus trip consists of n timepoints, there are $n-1$ timepoint segments. Since the historical travel time for each timepoint segment in each month will have a data distribution (which is represented as the mean value μ, the median value m and the standard deviation σ), the feature vector for each month can be represented as:

$$[\mu_1, m_1, \sigma_1, \mu_2, m_2, \sigma_2, ..., \mu_{n-1}, m_{n-1}, \sigma_{n-1}] \tag{4}$$

D. Clustering Analysis

The trip data per month was clustered using the feature vector in Equation 4 by K-Means algorithm:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{5}$$

where μ_i denotes the mean of all points in cluster S_i.

If the upper bound on the number of clusters is not set, then it is set to the number of months for which the data is available. The gap statistic [25] is used to find the optimal number of clusters. Figure 3 plots the [mean, standard deviation, median] vectors of the monthly travel time for a segment (WE23-MCC5_5) on a bus trip of route 5 (Figure 7). It clearly shows the variation between monthly data and these 5 months can be clustered into two groups: [April, May, June] and [July, August]. This variation is used to produce different schedule

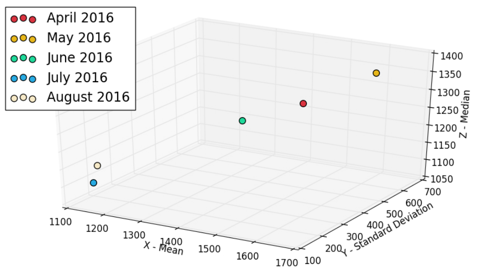

Fig. 3. The feature vectors (mean, standard deviation, median]) of the travel time in 5 months for a segment (WE23-MCC5_5) on a bus trip of route 5.

for these clusters. It should be noted that if the upper bound of number of clusters is set to 1 then only one schedule is generated. However, in our analysis we have seen that generating the schedule per cluster is better. This is shown later in section V-A.

E. A Genetic Algorithm to Optimize Bus Schedules

Since in our problem there are constraints that (1) the scheduled time at the first timepoint in each trip should not be changed, (2) the scheduled arrival time and departure time at the proceeding timepoints should remain in the same (dwell time is included in the expected travel time of the next segment, plus the range of [-1,+6] of on-time performance is able to account for dwell time variations as well) , the timetable for each trip can be decided by (1) the scheduled departure time at first timepoint, which is fixed, and (2) the scheduled arrival times at other timepoints, which are decided by the scheduled travel time between any two subsequent timepoints along the trip. Particularly, the following genetic algorithm terms are used:

- chromosome/individual: a solution in the genetic algorithm, which is a vector of integers representing travel time between subsequent timepoints.
- population: a set of solutions in each iteration.

In order to reduce the search space and match the real-world scenarios, the travel time in each individual is re-sampled to a multiple of 60 seconds.

Initialization When designing a genetic algorithm, estimating a good initial state is critical. Population size determines how many chromosomes are there in one population and affects the ultimate performance and computation efficiency [26]. Smaller population makes iterations faster but less various in chromosome crossover. Larger population will have the opposite effects. We chose 50 as the population size $pSize$.

In order to initialize the first population, the actual travel time between timepoints is aggregated from the historical datasets. Then the travel time in each individual is randomly selected between the maximum and minimum of historical data. We observed that the seeding in the initial population with heuristic solutions such as original scheduled travel time or optimized results from the greedy algorithm (presented in Section IV-F) would only affect the fitness of initial population and had little effects on the final optimality, so the initial population is generated at random.

Selection At the beginning of each iteration step, a portion of the existing population needs to be selected as parents to breed a new generation. A fitness function is required to determine how fit a solution is and a selection strategy is needed to select the solutions with better fitness. In our case, the objective function, defined in equation 1 is used as the fitness function. Since the fitness function contains an indicator function $I(h_i, s_j)$, and the value of the indicator function is related to the arrival delay at timepoint s_j, a simulation mechanism is needed to evaluate the on-time performance of the new schedule using historical data. To simulate the bus arrival and departure activities at timepoints, historical travel times between two consecutive timepoints and historical dwell time at timepoints are used.

To estimate the historical dwell time caused by passengers, we consider the following two scenarios in historical data: (1) if a bus arrives earlier than scheduled time, the waiting time between the scheduled time and actual departure time is used, (2) if a bus arrives later than scheduled time, the waiting time between the actual arrival time and departure time is used. For example, for the Timepoint 2 in Figure 1 with scheduled time of 10:20:

- If a historical bus arrived earlier at 10:17 and departed at 10:25, since the bus would always wait there for 3 minutes (between actual arrival time 10:17 and schedule time 10:20) regardless of there were passengers or not, we assume the dwell time caused by passengers is the extra time after the scheduled time (10:25 - 10:20 = 5 minutes).
- If a historical bus arrived later at 10:23 and departed at 10:25, then the dwell time caused by passengers is the extra time after the actual arrival time (10:25 - 10:23 = 2 minutes).

In the simulation, historical dwell time caused by passengers is added to the simulated arrival time at a timepoint, if the sum time is still earlier than the new scheduled time, then the simulation waits for extra time until the new scheduled time. The simulated departure time $st_{h,s_{j+1}}^{depart}$ at a timepoint s_{j+1} can be calculated using the simulated departure time st_{h,s_j}^{depart} at previous timepoint s_j, the actual travel time $t_{s_{j+1}}^{arrive} - t_{s_j}^{depart}$ between s_j and s_{j+1}, the dwell time $t_{s_{j+1}}^{dwell}$.

Thus the new schedule time $T_{h,ss_{j+1}}^{depart}$ at s_{j+1} is calculated using the following equation:

$$st_{h,s_{j+1}}^{depart} = \max(T_{h,ss_{j+1}}^{depart}, st_{h,s_j}^{depart} + (t_{s_{j+1}}^{arrive} - t_{s_j}^{depart}) + t_{s_{j+1}}^{dwell}) \tag{6}$$

For example, if the scheduled time at timepoint 2 in Figure 1 is changed to 10:16, since the bus of Trip 1 took 17 minutes to arrive at timepoint 2 and the historical dwell time by passengers is 0, the bus will be simulated to arrive 1 minutes later than the new schedule time and depart immediately after arrival.

Our genetic algorithm uses tournament selection [27] to randomly select new solutions. Each time, we select 2 individuals at random from the current population and pick the one with better fitness to become a parent. This process is repeated until the number of parents reaches the population size. The specifics are:

Crossover Using crossover, sub-solutions on different chromosomes are combined at random. A uniform crossover [28] technique is used for the crossover operation in gene level. Unlike one point or multi-point crossover, uniform crossover treats each gene separately. Two parents are randomly selected and their genes are exchanged (the scheduled travel time between two successive timepoints with another on the same places of the solution vector. The individual travel times between two parents are swapped with a fixed probability of 60%. An example illustrating the crossover is shown in Figure 4.

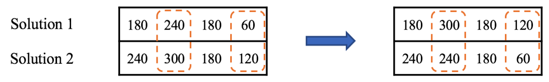

Fig. 4. Crossover: two genes are swapped between two individuals.

Mutation Mutation generates genetic diversity from one generation of a population of chromosomes to the next. The mutation works in two steps: (1) a schedule travel time between two timepoints in a solution is selected at random, (2) randomly add or minus 60 seconds to the time with the requirement that the new time should be within the historical travel time distribution range. The mutation probability is set as 0.005 and the population size ps is 50. Suppose each individual has 5 genes, 250 genes in total should lead to the result that one gene will mutate in each iteration.

Termination The termination condition of a genetic algorithm is critical to determine whether the algorithm should end or not. According to the study of stopping criteria for genetic algorithm [29], the following three types of conditions are mostly employed: (1) an upper limit of generation number is reached, (2) an upper limit of fitness function value is reached, (3) the change or achieving significant changes in the next generation is excessively low. Since the best on-time performance that the GA can achieve for each bus trip varies, setting the upper limit of the fitness function value does not work here. So we choose 1,000 as the upper generation number limit. At the same time, if the difference between the average fitness value of the solutions in the current generation and previous generation is below a pre-defined threshold 0.00001, then the algorithm will also terminate.

The pseudo code of the genetic algorithm is given in Algorithm 1. We utilize historical timepoint datasets to conduct the genetic algorithm for this optimization problem. The input includes on-time range, number of generation limit, number of solutions in the population, termination threshold, crossover and mutation probability, bus trip and upper limit of number of month clusters.

F. *Using A Greedy Algorithm to Optimize Bus Schedules*

We also used a greedy algorithm to compare the computation efficiency and optimization performance with the genetic algorithm. The basic idea of the greedy algorithm to optimize a bus trip's timetable is to adjust the scheduled arrival time greedily from the first timepoint to the last timepoint. Based on historical data, this algorithm will deal with each timepoint one by one. The first timepoint will not change. Then for the second timepoint, newly scheduled time that can maximize the percentage of on-time arrival delay within range $[t_{early}, t_{late}]$ at the current timepoint will be chosen. The process remains the same for subsequent timepoints.

Algorithm 1: Genetic algorithm for bus on-time performance optimization

```
Data: $D \leftarrow$ Historical timepoint datasets
Input : (1) $[t_{early}, t_{late}] \leftarrow$ on-time range , (2) $maxGen \leftarrow$
        maximum number of generations $maxGen$, (3) $pSize$
        $\leftarrow$ number of solutions in the population $pSize$, (4) $tt$
        $\leftarrow$ termination threshold, (5) $cP \leftarrow$ crossover
        probability, (6) $mP \leftarrow$ mutation probability, (7) $h \leftarrow$
        bus trip for optimization, (8) $upperLimit \leftarrow$ upper
        limit of the number of clusters
Output: Optimized schedule b at timepoints for bus trip $h$
GetAllTimepoints($D$, $h$);
GetHistoricalData($D$, $h$);
$monthClusters \leftarrow$ ClusterMonthData($upperLimit$);
for $monthCluster \in monthClusters$ do
    $P \leftarrow$ [];
    for $population\ size\ pSize$ do
        $P \leftarrow P \cup InitialIndividual()$;
    end
    $i_{population} \leftarrow 0$;
    while $maxGen$ is reached or AverageFitness($P_{i_{population}}$)
    - AverageFitness($P_{i_{population-1}}$) $\leq tt$ do
        $P \leftarrow$ TournamentSelect($P$);
        $P \leftarrow$ UniformCrossover($P$, $cP$);
        $P \leftarrow$ Mutation($P$, $mP$);
    end
end
```

Initialization The initialization step prepares the data for following steps. The actual travel time data between any two consecutive timepoints is aggregated using the historical dataset.

Optimization In the optimization step, the scheduled arrival time from the second timepoint to the last timepoint in a trip is optimized sequentially. Our goal is to pick new schedule time for two consecutive timepoints that can maximize the bus arrivals with delay within desired range $[t_{early}, t_{late}]$. It's a greedy algorithm because when adjusting the schedule time for a timepoint, only the on-time performance of the preceding timepoints and the current timepoint is considered. Figure 5 shows an example of the travel time distribution between timepoints on a bus trip in May 2016. We can visually observe that the travel time data distributions do not identically follow any fixed distribution. Based on the observation, instead of assuming the data follows any specific distribution (e.g. Gaussian distribution), we decide to utilize the empirical cumulative distribution function (CDF) to evaluate the percentage of historical delay in desired range.

An empirical CDF is a non-parametric estimator of the CDF of a random variable. The empirical CDF of variable x is defined as:

$$\hat{F}_n(x) = \hat{P}_n(X \leq x) = n^{-1} \sum_{n=1}^{n} I(x_i \leq x) \tag{7}$$

where $I()$ is an indicator function:

$$I(x_i \leq x) = \begin{cases} 1, & \text{if } x_i \leq x \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Then the CDF of x in range $[x + t_{early}, x + t_{late}]$ can be calculated using the following equation:

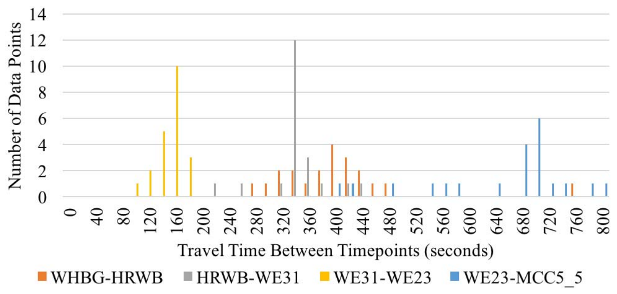

Fig. 5. Travel time distribution between consecutive timepoints on a bus trip in May 2016.

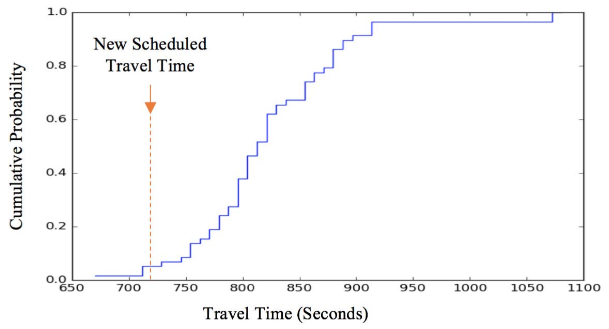

Fig. 6. Empirical cumulative distribution function (CDF) of historical travel time between two timepoints (MCC5_5 and WE23) on route 3 in May, June, July 2016.

$$\begin{aligned} &\hat{F}_n(x + t_{late}) - \hat{F}_n(x + t_{early}) \\ &= n^{-1} \sum_{n=1}^{n} I(x + t_{early} \leq x_i \leq x + t_{late}) \end{aligned} \tag{9}$$

Figure 6 illustrates an example of the empirical cumulative distribution function (CDF) of historical travel time between two timepoints (MCC5_5 and WE23) on route 3 in May, June, July 2016. Choosing a new scheduled travel time of 720 seconds between these two timepoints could maximize the percentage of historical data points within range $[720 + t_{early}, 720 + t_{late}]$.

1) Evaluation: The on-time performance of optimized schedule is evaluated using simulation. The simulated new arrival time using the new schedule is calculated using the simulated travel time equation described in the selection phase of section IV-E. Algorithm 2 shows the greedy algorithm's pseudo code.

V. Simulation Results and Discussion

The data involved are static bus schedule in General Transit Feed Specification (GTFS) format from Nashville MTA and recorded timepoint dataset in excel sheet files. The timepoint datasets contains historical data between April 2016 to August 2016. All data for each month is divided into two subsets at random: (1) 75% of the data is in the training set for generating new schedule (2) the rest 25% data is in the validating set for validating the new schedule.

Algorithm 2: Greedy algorithm for bus on-time performance optimization

```
Data: $D \leftarrow$ Historical timepoint datasets
Input : (1) $[t_{early}, t_{late}] \leftarrow$ on-time range, (2) $h \leftarrow$ bus trip
        for optimization, (3) $upperLimit \leftarrow$ upper limit of
        the number of clusters
Output: Optimized schedule b at timepoints for bus trip $h$
$[s_1, ..., s_n] \leftarrow$ GetAllTimepoints($D$, $h$);
GetHistoricalData($D$, $h$);
$monthClusters \leftarrow$ ClusterMonthData($upperLimit$);
for $monthCluster \in monthClusters$ do
    $b \leftarrow []$;
    for $s_i \in [s_1, ..., s_n]$ do
        $maxCDF \leftarrow 0$;
        $optimizedTime \leftarrow 0$;
        for candidate schedule time set $x$ do
            if $maxCDF \leq$
            CalculateEmpiricalCDF$(x, t_{early}, t_{late})$ then
                $maxCDF \leftarrow$
                CalculateEmpiricalCDF($x, t_{early}, t_{late}$);
                $optimizedTime \leftarrow x$
            end
        end
        $b \leftarrow b + optimizedTime$
    end
end
```

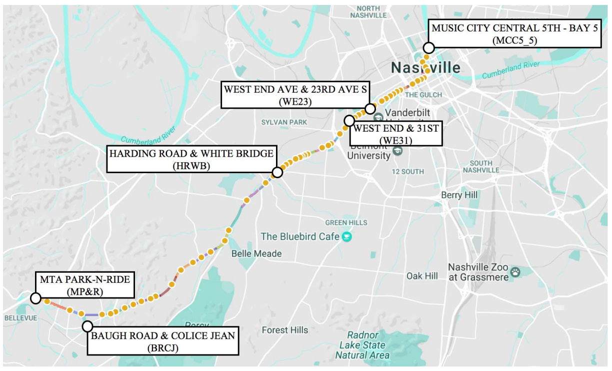

Fig. 7. Timepoints on bus route 5 in Nashville

A. Comparing Single Schedule vs Cluster Specific Schedule

For a trip, if its historical travel time patterns is clustered into 2 groups (e.g. [April, May] and [June, July, August]), then two separate bus schedule will be generated for each month cluster using the corresponding month data. The on-time performance using the new schedule is then simulated for each month using the validation dataset. Route 5, which connects the downtown and south west communities in the city, is one of the major bus routes in Nashville. It contains six timepoints (MCC5_5, WE23, WE31, HRWB, BRCJ, MP&R) and five timepoint segments along the route (shown in Figure 7). We use route 5 that runs between downtown (Stop: Music City Central 5th Cir) and southwest (Stop: Coley Davis-Shelter-Park N Ride) of Nashville to study the two strategies: (1) using clustered month data to generate separate bus schedules and (2) using all available data to build a uniform bus schedule. Table III and Table IV show the original on-time performance, optimized

TABLE III. Simulated results by using all month data to generate a single timetable.

	April	May	June	July	August
Original	70.13%	71.41%	69.87%	68.38%	70.52%
Greedy	74.82%	71.12%	76.55%	73.44%	71.87%
Genetic	79.08%	77.87%	79.62%	77.36%	77.56%

TABLE IV. Simulated results by using month grouped data to generate cluster specific timetables

	April	May	June	July	August
Original	70.13%	71.41%	69.87%	68.38%	70.52%
Greedy	74.22%	73.42%	74.86%	73.74%	71.38%
Genetic	79.98%	78.03%	80.79%	79.55%	79.50%

performance using greedy algorithm and optimized performance using genetic algorithm using the two strategies. If the month data is not grouped, the average on-time performance in these five months improved to 78.29% from 70.06% using the genetic algorithm. By grouping the months with similar patterns, the average on-time performance after optimization is increased further to 79.57%.

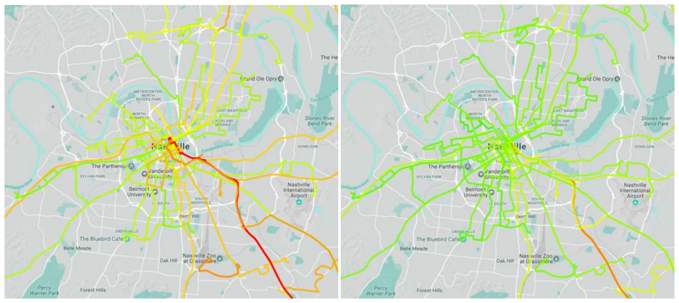

Fig. 8. Route heatmap shows the original (left) and optimized (right) on-time percentages of historical trips between April and August 2016 where bus arrival delay at timepoints are between 1 mins early and 6 min late

B. Comparing optimization results using genetic algorithm and greedy algorithm

In the second experiment we apply the proposed unsupervised mechanism to optimize the on-time performance for all the bus routes in the city of Nashville. For each bus trip, the trip's is grouped by using historical timepoint data in April, May, June, July and August. The original on-time performance, optimized performance using greedy algorithm and optimized performance using genetic algorithm are shown in Figure 9. The results validate our assumption that while both algorithms can improve the on-time performance, the genetic algorithm will outperform the greedy algorithm because it can optimize the schedule for all timepoint segments on each trip all together. The original on-time performance of all bus routes in origin is 62.9%. The greedy algorithm improved it to 67.8% and the genetic algorithm improved it further to 74.7%. Figure 8 visually illustrates the on-time performance on each route before and after optimization using heatmaps. The color on the path of each route is from red to green depending on the percentage of on-time buses at timepoints. We evaluated the greedy and genetic algorithms on a MacBook laptop (2.0 GHz Intel Core i7 processor and 8 GB 1600MHz DDR3 memory). The computation time of the genetic algorithm is roughly 4 times than the greedy algorithm (9477.29 seconds

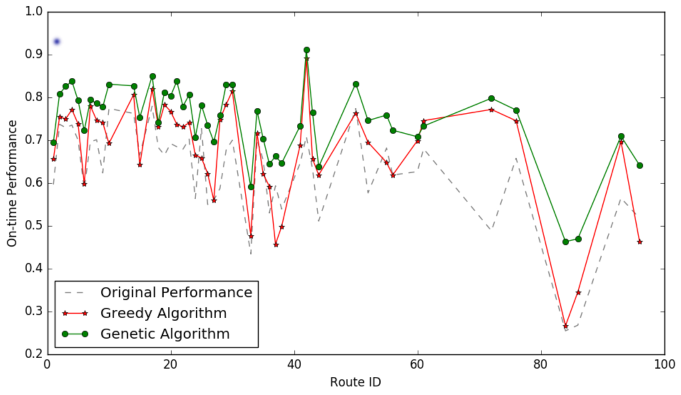

Fig. 9. Actual and simulated on-time performance using data of between Aprial and August 2016 by (1) original schedule, (2) optimized schedule using greedy algorithm, (3) optimized schedule using genetic algorithm

v.s. 2406.91 seconds) to optimize all the 57 bus routes. Even though the genetic algorithm takes much longer time, it is still worth to choose it because the schedule optimization can be executed offline and the optimization performance of the genetic algorithm is better.

VI. Conclusion

In this paper, we formulate the bus on-time performance optimization problem, propose an unsupervised mechanism that clusters historical data on different months based on the travel time patterns, and develop a genetic algorithm to generate new timetables for different month groups. Our goal is to maximize the probability of bus trips that can reach the desired on-time range at timepoints. Simulation results show that the on-time performance on bus routes are improved by 11.8% on average. In the future, we plan to utilize more data to extend the work so that we can study the seasonal variations and optimize the algorithms further.

Acknowledgments

This work is supported by The National Science Foundation under the award numbers CNS-1528799 and CNS-1647015 and a TIPS grant from Vanderbilt University. We acknowledge the support provided by our partners from Nashville Metropolitan Transport Authority.

References

[1] A. P. T. A. (APTA), “Americans took 10.6 billion trips on public transportation in 2015,” 2016.

[2] J. Lin, P. Wang, and D. T. Barnum, “A quality control framework for bus schedule reliability,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 6, pp. 1086–1098, 2008.

[3] F. Sun, Y. Pan, J. White, and A. Dubey, “Real-time and predictive analytics for smart public transportation decision support system,” in *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[4] A. Oruganti, F. Sun, H. Baroud, and A. Dubey, “Delayradar: A multivariate predictive model for transit systems,” in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1799–1806.

[5] F. Sun, A. Dubey, J. White, and A. Gokhale, “Transit-hub: A smart public transportation decision support system with multi-timescale analytical services,” *Cluster Computing*, 2017.

[6] H. Benn, “Bus route evaluation standards, transit cooperative research program, synthesis of transit practice 10,” *Transportation Research Board, Washington, DC*, 1995.

[7] S. A. Arhin, E. C. Noel, and O. Dairo, “Bus stop on-time arrival performance and criteria in a dense urban area,” *International Journal of Traffic and Transportation Engineering*, vol. 3, no. 6, pp. 233–238, 2014.

[8] S. Pattnaik, S. Mohan, and V. Tom, “Urban bus transit route network design using genetic algorithm,” *Journal of transportation engineering*, vol. 124, no. 4, pp. 368–375, 1998.

[9] P. Chakroborty, “Genetic algorithms for optimal urban transit network design,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 18, no. 3, pp. 184–200, 2003.

[10] Y. Hairong and L. Dayong, “Optimal regional bus timetables using improved genetic algorithm,” in *Intelligent Computation Technology and Automation, 2009. ICICTA’09. Second International Conference on*, vol. 3. IEEE, 2009, pp. 213–216.

[11] M. A. Nayeem, M. K. Rahman, and M. S. Rahman, “Transit network design by genetic algorithm with elitism,” *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 30–45, 2014.

[12] W. Y. Szeto and Y. Wu, “A simultaneous bus route design and frequency setting problem for tin shui wai, hong kong,” *European Journal of Operational Research*, vol. 209, no. 2, pp. 141–155, 2011.

[13] J. Hora, T. G. Dias, and A. Camanho, “Improving the robustness of bus schedules using an optimization model,” in *Operations Research and Big Data*. Springer, 2015, pp. 79–87.

[14] F. Zhao and X. Zeng, “Simulated annealing–genetic algorithm for transit network optimization,” *Journal of Computing in Civil Engineering*, vol. 20, no. 1, pp. 57–68, 2006.

[15] M. Friedman, “A mathematical programming model for optimal scheduling of buses’ departures under deterministic conditions,” *Transportation Research*, vol. 10, no. 2, pp. 83–90, 1976.

[16] V. Guihaire and J.-K. Hao, “Transit network design and scheduling: A global review,” *Transportation Research Part A: Policy and Practice*, vol. 42, no. 10, pp. 1251–1273, 2008.

[17] P. Chakroborty, K. Deb, and P. Subrahmanyam, “Optimal scheduling of urban transit systems using genetic algorithms,” *Journal of transportation Engineering*, vol. 121, no. 6, pp. 544–553, 1995.

[18] M. Naumann, L. Suhl, and S. Kramkowski, “A stochastic programming approach for robust vehicle scheduling in public bus transport,” *Procedia-Social and Behavioral Sciences*, vol. 20, pp. 826–835, 2011.

[19] S. L. Tilahun and H. C. Ong, “Bus timetabling as a fuzzy multiobjective optimization problem using preference-based genetic algorithm,” *Promet-Traffic&Transportation*, vol. 24, no. 3, pp. 183–191, 2012.

[20] A. Ceder, *Public transit planning and operation: Modeling, practice and behavior*. CRC press, 2016.

[21] M. Mandelzys and B. Hellinga, “Identifying causes of performance issues in bus schedule adherence with automatic vehicle location and passenger count data,” *Transportation Research Record: Journal of the Transportation Research Board*, no. 2143, pp. 9–15, 2010.

[22] F. Sun and A. Dubey, “T-hub timetable optimization project repository,” https://github.com/visor-vu/thub-timetable-optimization.

[23] “General transit feed specification (gtfs) and gtfs realtime,” https://developers.google.com/transit/, 2017, accessed: 2017-02-10.

[24] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, “Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median,” *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.

[25] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[26] J. J. Grefenstette, “Optimization of control parameters for genetic algorithms,” *IEEE Transactions on systems, man, and cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[27] B. L. Miller and D. E. Goldberg, “Genetic algorithms, tournament selection, and the effects of noise,” *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[28] G. Syswerda, “Uniform crossover in genetic algorithms,” 1989.

[29] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole, “On stopping criteria for genetic algorithms,” in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 405–413.