Journal of Computer and Communications, 2020, 8, 14-54
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

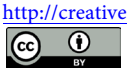
Parallel Multiple Tabu Search for Multiobjective Urban Transit Scheduling Problem

Vikneswary Uvaraja[1], Lai Soon Lee[1,2*], Nor Aliza Abd Rahmin[1], Hsin Vonn Seow[3]

[1]Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, Serdang, Malaysia
[2]Institute for Mathematical Research, Universiti Putra Malaysia, Serdang, Malaysia
[3]Nottingham University Business School, University of Nottingham Malaysia Campus, Semenyih, Malaysia
Email: *lls@upm.edu.my

How to cite this paper: Uvaraja, V., Lee, L.S., Rahmin, N.A.A. and Seow, H.V. (2020) Parallel Multiple Tabu Search for Multiobjective Urban Transit Scheduling Problem. *Journal of Computer and Communications*, **8**, 14-54.
https://doi.org/10.4236/jcc.2020.85002

Received: April 3, 2020
Accepted: May 6, 2020
Published: May 9, 2020



Abstract

Urban Transit Scheduling Problem (UTSP) is concerned with determining reliable transit schedules for buses and drivers by considering the preferences of both passengers and operators based on the demand and the set of transit routes. This paper considered a UTSP which consisted of frequency setting, timetabling, and simultaneous bus and driver scheduling. A mixed integer multiobjective model was constructed to optimize the frequency of the routes by minimizing the number of buses, passenger's waiting times and overcrowding. The model was further extended by incorporating timeslots in determining the frequencies during peak and off-peak hours throughout the time period. The timetabling problem studied two different scenarios which reflected the preferences of passengers and operators to assign the bus departure times at the first and last stop of a route. A set covering model was then adopted to minimize the number of buses and drivers simultaneously. A parallel tabu search algorithm was proposed to solve the problem by modifying the initialization process and incorporating intensification and diversification approaches to guide the search effectively from the different feasible domain in finding optimal solutions with lesser computational effort. Computational experiments were conducted on the well-known Mandl's and Mumford's benchmark networks to assess the effectiveness of the proposed algorithm. Competitive results are reported based on the performance metrics, as compared to other algorithms from the literature.

Keywords

Urban Transit Scheduling, Multiple Tabu Search, Parallel, Frequency Setting, Timetabling, Big Data

1. Introduction

Urban public transportation is an alternative mode for travelers in major cities to commute between destinations. It is undeniable that the evolution of the public transportation system is based on its demand and current technologies. It is also an efficient approach to enhance the ridership and combat the threat posed by motor vehicles, mainly traffic congestion and air pollution. The development of public transportation service must be consistent with the requirement of both passengers and operators in order to create a sustainable system. Some of the major disadvantages of public transportation are the unavailability of the service at desirable times, inconvenience while traveling, and inaccurate transit schedules.

The study on constructing effective schedules for urban public transportation can be formulated as an optimization problem which is known as the Urban Transit Scheduling Problem (UTSP). UTSP can be divided into a few sub-problems that consist of frequency setting, timetabling, vehicle scheduling and crew scheduling since it is usually difficult to solve them simultaneously due to its numerous objectives and decision variables [1]. The determination of headways or frequency serves as tactical planning to satisfy the irregular demand that changes according to time, days and seasons. This process is important to maintain the quality of transit service, because the unexpected arrival and departure times can affect the service reliability and also cause congestion in the transit. Besides, providing higher frequency is redundant at some time periods which reduce the efficiency of the transit service. Usually, the timetables are constructed based on the frequency predetermined while considering the synchronization at transfer nodes. Meanwhile, the vehicles and crews are assigned according to the industry regulation and resources available.

Due to the complexity of the problem, metaheuristic approaches have been widely applied to find near-optimal solutions in a reasonable time. However, the parallel algorithm is seldom being implemented in urban transportation problem which may due to the complication involved in how and where to do parallelism. The parallelization of an algorithm can be done in several ways depending on the problem structure and computer hardware. It is important to consider the type of parallelism to reduce cost and avoid unnecessary communication during data transfer which might increase the execution time for some cases.

Apart from the genetic algorithm (GA), tabu search (TS) and simulated annealing (SA), multiple tabu search (MTS) is an emerging approach in solving the combinatorial optimization problem [2]. However, to the best of our knowledge, MTS has not been applied in solving multiobjective UTSP to date. In this paper, a parallel MTS (PMTS) with systematic neighborhood selection approach is developed by modifying the initialization process and incorporating intensification and diversification procedures to produce optimal solutions. The proposed algorithm is verified and validated using well-known Mandl's and Mumford's benchmark datasets. This paper extends the previous sequential methodology in

[3] to study multiobjective UTSP using parallel implementation. The main contribution of this paper is the development of parallel MTS algorithm in reducing the computational time of solving the UTSP.

This paper is organized as follows. In Section 2, the review of the related work in this study is presented. The model formulations for frequency optimization as well as bus and driver scheduling are described in Section 3. In Section 4, the development of PMTS algorithm for UTSP is proposed. The computational experiments and results for benchmark datasets are reported in Section 5. Section 6 gives the conclusions and recommendations for future research.

2. Literature Review

Over the years, the evolution of operational research and the development of computing infrastructure have created high interest in tackling UTSP. Various optimization algorithms are proposed to search for the optimal solution effectively. Nevertheless, the ability of the algorithm to obtain optimal or near-optimal solutions in polynomial time for hard optimization problem such as UTSP is still very challenging. Review of the existing approaches in UTSP reveals the shortage of advanced optimization algorithm in handling the large and complex problems [4] [5] [6] [7]. Moreover, the deficiency of multiobjective approaches to solving UTSP is also reflected in the analysis. Due to the scope of this paper, only the study of multiobjective UTSP and the applications of parallel algorithms in public transportation design are discussed.

A multiobjective metaheuristic approach based on TS and GA is developed by [8] to solve bus driver scheduling problem and produce Pareto optimal solutions. Multi-objective set covering model is used to define the problem by including some measures of service quality needed by different companies in the objective function. GRASP has been used as a subroutine for the metaheuristic. A TS with three types of neighborhood selection and intensification strategies for a certain number of iterations are applied to find the best solution. On the other hand, [9] studied the transit route network design using a parallel genetic algorithm (PGA). It includes the determination of a set of transit routes and the associated frequencies that minimize the sum of the operating cost and the generalized travel cost. Two PGA models are proposed which based on the global parallel virtual machine and global message passing interface.

Bus transit network is optimized by developing a model with parallel ant colony optimization [10] [11]. The objectives are to achieve minimum transfers and maximum passenger flow per unit length with line length and non-linear rate as constraints. A heuristic pheromone distribution rule is applied, by which ants' path searching activities are altered based on the objective value. A parallel auction algorithm is proposed by [12] for bus rescheduling problem (BRP) that occur when a trip is disrupted. A single-depot BRP model is built to minimize the operating and delay cost. The sequential and parallel auction algorithms are developed to solve the BRP. The combined forward and backward auction itera-

tions are used with the implementation of ϵ-scaling to improve the performance of the auction algorithm.

Reference [13] presented a model for optimizing bus route headways that maximize the service quality and minimize the operational cost. The relative weights between the passenger cost and operator cost are determined by an integrated approach. A PGA based on coarse-grain strategy and a local search based on TS are incorporated to improve the performance of the GA. While [14] investigated the transit network design of large urban area (City of Rome) with the objectives to minimize the sum of operator cost, user cost and penalty measuring the level of unsatisfied demand. The contributions of the paper are the introduction of flow concentration procedure in a wider route generation process and the application of transit network design methodology to a large real-life urban area.

A methodology is proposed by [15] to create Pareto solutions for minimizing the number of vehicles and drivers to satisfy a given schedule. Multiple block subsets are chosen from a set of candidate vehicle blocks by improved multiobjective GA with departure-time adjustment procedure. While [16] investigated a multi-objective re-synchronizing of bus timetable problem that characterized by headway-sensitive passenger demand, uneven headways, service regularity, flexible synchronization and involvement of existing bus timetable. A multi-objective optimization model is derived to make a trade-off between the total number of passengers benefited by smooth transfers and the maximal deviation from the departure times of the existing timetable.

The problem of vehicle scheduling in urban public transport systems taking into account the vehicle-type with different capacity is studied by [17]. A heuristic-based on the multi-objective cellular evolutionary algorithm is proposed to solve the problem considering restrictions of government agencies. The objectives of minimizing the total operating cost, waiting time and congestion in the bus and maximizing the quality of service are considered to produce a set of non-dominated solutions that represent different assignments of vehicles to cover the trips of a specific route. While [18] presented an optimization-based approach to simultaneously solve the network design and the frequency setting phases on the context of railway rapid transit networks. A Lexicographic goal programming model is introduced, together with a solving strategy.

Most recently, [19] proposed a single framework that simultaneously considers the restrictions and objectives of the users and operator of a bus rapid transit system. Routes and frequencies were searched for this system by minimizing waste bus capacities (operation costs) and minimizing users' travel time (maximizing satisfaction). A Multiobjective Global-Best Harmony Search heuristic algorithm is implemented.

There are many studies that investigate UTSP by the different methodological approaches but the time complexity is usually overlooked with the effectiveness of the algorithm. Although several metaheuristic algorithms are derived to solve UTSP, the implementation of parallel techniques to enhance the difficulties of

solution procedure while reducing the computational time are seldom considered which is the research gap to be satisfied in this paper.

3. Problem Formulation

UTSP is formulated as a minimization problem taking into consideration of the preferences of passengers and operators. Generally, passengers would prefer to wait with a shorter time while maintaining their comfort and convenience while travelling. On the contrary, operators would be preferred to provide fewer buses and drivers to reduce the operating cost. These objectives are important to improve the overall performance of the public transportation system.

Many researchers combine these objectives into a single function under the resources constraints such as fleet size [20] [21] [22], bus loading [23] [24] and frequency boundary [25]. While, the multiobjective optimization model is employed by [15] [26] [27] to produce Pareto optimal solutions. Similarly, this study also optimizes all the conflicting objectives simultaneously to yield various solutions with different tradeoff levels.

In this study, UTSP is tackled consequently starting from frequency setting where the frequencies of each route are optimized based on the passengers' demand and total travel time between origin and destination. Then, a timetable is constructed by setting expected departure times for each route using their headways. The timetabling procedure does not involve separate model formulation since its objectives such as demand satisfaction and fleet size are already included in frequency optimization. Besides, the transfer synchronization in constructing timetables is excluded due to the lack of data on passengers' demand at every bus stop. Finally, the buses and drivers are optimized and assigned correspondingly to all departure times of the routes.

3.1. Frequency Optimization

The frequency optimization procedure can be best represented as a bi-level process. The first level explains the passenger assignment procedure and the second level describes the frequency optimization procedure by PMTS. Passengers' demand obtained from the first level is used to find the frequency of the route in the second level. The frequency set obtained is used to update the initial frequency and to restart the process. This process is reiterated until the convergence pattern of the frequency set is observed.

Two cases are considered to determine the optimal frequency for each route. The first case assumed that the demand and the frequency of a route are similar throughout the time period studied whereas, for the second case, the problem is extended to change the demand and frequencies of the routes according to the peak and off-peak hours. The total demand is divided into 18 timeslots (1-hour periods) that have been categorized based on the assumed passengers' traffic. The first case is to test the effectiveness of the proposed algorithm in comparison to other methods in the literature using a similar model. The second case is to

evaluate the proposed algorithm using additional criteria such as time-dependent demands that represent a more realistic situation. Since there is no benchmark data available on time-dependent demand for transportation network studied, the demand during peak hours is assumed to be double of the off-peak hours. Let assume the passenger demands for a route is 300 passengers. It is divided in the ratio of 1:2 that represent the demand on off-peak hours to peak hours. Note that, the second case is tested on Mandl's network only and its results are used to study the bus and driver scheduling problem in the latter section.

To begin with, all routes are initialized with similar frequency before the passenger demands are assigned based on their route choice. Two passenger assignment methods are adopted from [28] [29] as they use a realistic representation of passenger's behavior and the methods are applied in the comparative studies mentioned in Section 5.2.1. These methods allocate the passengers' demand based on the frequency share rule and multinomial logit model respectively.

Generally, the passengers are set to travel in a path with at most two transfers and the demand is considered unsatisfied if more than two transfers are needed. Based on [28], when more than one path exists for the same number of transfers, the demands are allocated to the routes within prespecified travel times such that each route carries a proportion of the flow equals to the ratio of its frequency to the sum of frequencies of all acceptable routes. Based on [29], the demands are distributed according to the travel time utility of each path if parallel paths exist with one or two transfers.

After the passenger assignment process, the maximum load of each route is obtained from its list of link flows and used as the input for the frequency optimization procedure. In order to optimize the frequency, a mixed integer programming model is built with the objectives of minimizing the total number of buses, passengers waiting times and overcrowding in the bus. This model is adapted from the literature by considering the overcrowding as one of the objectives rather than constraints with the same formulation.

Minimize

$$F_1 = \sum_{k \in R} \left[\frac{2 t_k f_k}{T} \right] \tag{1}$$

$$F_2 = \sum_{i \in N} \sum_{j \in N} \left[d_{ij} \frac{T}{2 \sum_{k \in R_{ij}} f_k} \right] \tag{2}$$

$$F_3 = \sum_{k \in R} \left[Q_k - \left(CAP(LF) f_k \right) \right] \tag{3}$$

subject to

$$f_{\min} \le f_k \le f_{\max} \quad \text{for all} \quad k \in R \tag{4}$$

where,

d_{ij} : number of passengers travelling between nodes *i* and *j*,

f_k : frequency of route *k*,

$f_{\min}$: minimum frequency for a time period,
$f_{\max}$: maximum frequency for a time period,
Q_k : maximum load (passengers) of route *k*,
t_k : vehicle travel time of route *k*,
CAP: seating capacity of a bus,
LF: load factor of a bus,
R: set of bus routes,
R_{ij} : set of potential routes between node *i* and *j*,
T: time horizon,
N: set of nodes in transit network.

Equation (1) calculates the number of buses needed for each route $k \in R$ that obtained by dividing the total round trip times with the time horizon (mins). Equation (2) measures the total waiting times for all the passengers. The waiting time is assumed to be half of the headway. Equation (3) determines the total number of passengers that exceed the maximum capacity of the bus. Equation (4) ensures that the frequency of each route is within the minimum and the maximum frequency for a time period. This model is further extended by including timeslot to find the frequencies of the routes at specific timeslot based on the variable demand. The representation of the new adapted model is as follows.

Minimize

$$F_1 = \sum_{k\in R}\left[\max_{s\in S}\left(\frac{2t_k f_{k,s} + Q_{k,s} DWP + l_k f_{k,s}}{T}\right)\right] \tag{5}$$

$$F_2 = \sum_{i\in N}\sum_{j\in N}\sum_{s\in S}\left[d_{ij,s}\frac{T}{2\sum_{k\in R_{ij}} f_{k,s}}\right] \tag{6}$$

$$F_3 = \sum_{k\in R}\sum_{s\in S}\left[Q_{k,s} - \left(CAP(LF) f_{k,s}\right)\right] \tag{7}$$

subject to

$$f_{\min} \le f_k \le f_{\max} \quad \text{for all} \quad k \in R \tag{8}$$

where,

$d_{ij,s}$: number of passengers travelling between nodes *i* and *j* in timeslot *s*,
$f_{k,s}$: frequency of route *k* in timeslot *s*,
l_k : layover time of route *k*,
$Q_{k,s}$: maximum load (passengers) of route *k* in timeslot *s*,
DWP: dwell time for a passenger,
S: set of timeslots.

Equation (5) determines the maximum number of buses needed for each route. Note that, the total round trip times includes dwell times and layover times. Dwell time (*DWP*) states the approximate time taken for boarding or deboarding passengers at every scheduled bus stop whereas layover time (l_k) refer to the time spent by a bus at the terminals (first and last stop) of a route without moving. Equation (6) measures the total waiting times for all the passengers in

every timeslot $s \in S$ while equation (7) calculates the total number of passengers that exceed the maximum capacity of the bus in all timeslots. Equation (8) represents the constraint on the frequency of each route.

Based on the two models, the decision variable is the frequency of the route which is represented as f_k and $f_{k,s}$ correspondingly and the travel time of a bus between two nodes is shown as t_k. The parameters such as bus capacity (*CAP*) and load factor (*LF*) denote the maximum number of seats available and maximum number of passengers that can be occupied in a bus, respectively. Meanwhile, Q_k and $Q_{k,s}$ represent the passengers demand at each route that obtained from the passenger assignment procedure respectively. For further explanation on the models, refer to [3].

3.2. Bus and Driver Scheduling

The proposed technique for solving this problem is inspired by [15] who studied the vehicle scheduling problem. The solution approach is revised by incorporating the elements for bus driver scheduling. Following are the definition of terms related to this sub problem. A trip is a one-way route that begins at a specific time from the starting terminal (bus stop) to ending terminal of a route. A vehicle block is the schedule of a bus that consists of consecutive trips allocated to it. There are two types of vehicle blocks used in this study which are represented as long and short blocks. The elapsed time of a short block is the total work duration whereas, for a long block, the time is doubled to be fulfilled by two drivers. The main idea of this assumption is to simplify the representation of the bus and driver scheduling problem.

At first, a set of vehicle blocks are generated to cover all the departure times according to the total work duration, driver's break duration and maximum working duration without break, which are set approximately as 9 hours, 1 hour and 4 hours, respectively. Then, a subset of candidate blocks with minimum objective functions values is selected using the proposed algorithm. All selected candidate blocks are reconstructed to minimize further the values of the objective function. The detailed explanation of this procedure is given in Section 4.2.

A set covering model is adapted from [15] to represent the problem and calculate the objective values in order to produce the optimal blocks. The set covering model generates a set of buses that cover all the trips with minimum cost. Moreover, it allows some of the trips to be included in more than one buses which consequently provides various combinations of trips for each bus. The set covering model for simultaneous bus and driver scheduling is formulated as follows:

Minimize

$$F_1 = \sum_{y=1}^{q} z_y \tag{9}$$

$$F_2 = \sum_{y=1}^{q} C_y z_y \tag{10}$$

subject to

$$\sum_{y=1}^{q} v_{xy} z_y \geq 1 \tag{11}$$

$$z_y = \{0,1\},\ C_y = \{1,2\},\ v_{xy} = \{0,1\},\ y = \{1,\cdots,q\},$$

where,

$$v_{xy} = \begin{cases} 1, & \text{if block } y \text{ has a trip starting from } x\text{th departure time} \\ 0, & \text{otherwise} \end{cases}$$

$$z_y = \begin{cases} 1, & \text{if block } y \text{ is in the solution} \\ 0, & \text{otherwise} \end{cases}$$

$$C_y = \begin{cases} 1, & \text{if } b_y \text{ is a short block} \\ 2, & \text{if } b_y \text{ is a long block} \end{cases}$$

Equations (9) and (10) minimize the number of buses and drivers respectively and Equation (11) ensures that every departure time is covered by at least one trip. The v_{xy} is a binary matrix that records the availability of departure times in a block whereas the binary decision variable z_y shows the presence of certain blocks in the chosen solution. The parameter C_y defines the number of drivers for each block and b_y is the yth block in a set of vehicle blocks such that q is the total.

The proposed solution approach covers all the departure times by producing extra vehicle blocks at the beginning rather than altering the existing departure times in the optimal blocks to include more departure times. This is due to the adjustment of departure times may affect the headways and layover time determined earlier which consequently have an impact on the robustness of the schedule.

It is important to note that the formulation to calculate the number of buses in this sub problem is different from the frequency optimization problem although both problems are closely related. In the frequency setting, the total buses are computed based on the round trip time of a bus and passengers demand without considering drivers work preferences. In this sub problem, several work rules such as driver's break duration have been included which can increase the round-trip times of buses. This consequently will increase the number of buses required to maintain the headways found in previous frequency setting problem.

4. Parallel Multiple Tabu Search

The idea of MTS is first proposed by [2] in designing the optimal fuzzy logic proportional-integral controller. The basic TS algorithm might require longer computational time to search for the expected solution if the initial solution is further away from the promising region. Thus, the adapted MTS algorithm is developed to guide the search to the optimal region in less computation time. The basic structure of MTS consists of initialization, adaptive searches, multiple searches, replacing and restarting procedure. MTS algorithm begins by generating several initial solutions to increase the possibility of reaching the optimal region quickly. Then, adaptive search mechanism is applied to alter the step size of

the neighborhood during the search and multiple TS algorithms are performed sequentially based on its initial solution.

In this study, the MTS algorithm is adapted such that each of the initial solutions is selected from different feasible domain to examine the search space thoroughly in finding better solutions that minimize all the objective values. The domains are obtained by partitioning the search space into a fixed range of values. This technique can increase the chance of evaluating every possible solution in a reasonable computational time. Besides, intensification and diversification processes are incorporated to exploit and explore the search space when there is no promising solution available. Moreover, two-dimensional tabu list is created to record the availability of elements in an organized memory structure [3].

The proposed PMTS is derived to make use of modern technology with multiple processors for exploring the search space more effectively in less computational time. It works differently for continuous optimization (frequency setting) and discrete optimization (bus and driver scheduling) problems. However, the basic idea of PMTS that finds feasible solutions in a systematic way by dividing the search space into several domains and handling them at the same time is applied in both problems. This research focuses on exploiting the data parallelism in order to partition the data optimally to numerous processors. Each processor performs the same task at the same time but using different data sets. Specifically, the input of the problem is partitioned into a fixed number of processors such that each processor has a distinct range of domain.

The PMTS algorithm is altered according to a master-slave strategy with multiple initial points and single strategy configuration. The parallel search is controlled by a single processor called master which distributes the data equally into several processors called slaves to execute the search in parallel. The slaves will start their search in different search space with different initial points from the data range given and perform similar functions independently. There is some communication between the master and slaves at the beginning to allocate the tasks and data and also at the end of executions when the optimal solutions are gathered at the master processor. Note that the slaves do not communicate directly to each other as there is no data transfer involved between them. The selection of this approach is motivated by the natural design of the MTS algorithm which guides the search separately with specific domain.

The PMTS begins with several initial solutions such that each of them is selected from various domains. All the starting solutions run simultaneously in different processors to search for the best solution in every domain. Explicitly, the search space is divided into a number of domains and each of the domains is allocated with different range/set of values. After the initialization process in continuous optimization, the adaptive search mechanism is applied to find variable step sizes for locating the neighborhood of the current solution to move them to another feasible solution. By referring to **Figure 1**, the neighborhood is formed by adding and subtracting the step size from each variable in the initial solution. Let Δ_n be the step size values such that n represent the domain.

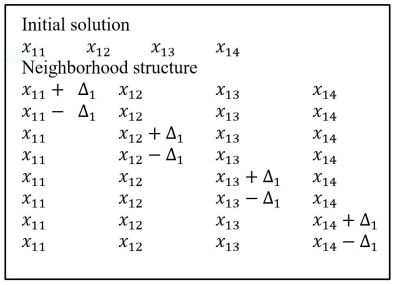


Figure 1. Neighborhood formation for continuous optimization.

Meanwhile, for discrete optimization, three classifications of the neighborhoods are formed based on *add*, *drop coarse-grain* and *swap* moves from where the best solution is chosen at every iteration. Based on **Figure 2**, the solutions in first 4 rows are formed by dropping a block from the initial solution; the following solutions are created by swapping and the last solution is formed by adding a new solution. Based on **Figure 1** and **Figure 2**, x_{nm} represent decision variables such that m is the elements. For example, x_{11} is the decision variable of element 1 in first domain.

The use of single or multiple tabu lists depends on the types of move involved during the search that directly influenced by the type of the problem. There are several moves being added or dropped at the same time to create the neighborhood of the current solution. This study applied two-dimensional tabu lists with the same tabu tenure to record the moves with their positions in the list. This approach inhibits repeated moves and enables the search to explore a variety of solutions using organized memory structure. An array, tabulist (a) (b) stores recently chosen solution to mark it as tabu for certain iterations such that a, the row represents the tabu tenure and b, the column represents the maximum number of moves that can be added or dropped. For all moves in the tabu list, the value of a is reduced by 1 at every iteration. The number of columns occupied depends on the number of moves involved in an iteration and it is different for every row.

Intensification is conducted if there is no promising solution available and diversification occurs when the intensification is not possible to be applied. This is because the normal search procedure in the proposed PMTS is good enough to analyze a portion of the whole neighborhood since it is divided earlier. Furthermore, it is unnecessary to spend extra time to examine the regions which are already been visited previously. Intensification is performed based on intermediate memory that functions by recording and updating some best trial solutions produced during the search based on the objective function values of the solutions. Diversification is performed based on long-term memory that counts and stores the number of times that a solution is involved in the current solution. The aspiration criteria allow the tabu move if the neighbor solution yields better result compared to the best-known solution. Termination criteria stop the search if there is no improvement in the objective values after a fixed number of iterations.

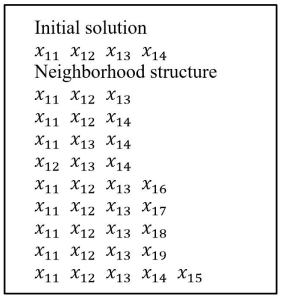


Figure 2. Neighborhood formation for discrete optimization.

4.1. Parallel Multiple Tabu Search for Frequency Optimization

The main idea of frequency optimization is to find the optimal frequency set for each route in a transit network based on the passengers' demand during a time period. Different frequency sets give different tradeoff between the objectives since this is a multiobjective problem. The specific steps of the PMTS algorithm are described as follows. The pseudo-code of the algorithm for solving the frequency optimization problem is given in **Algorithm 1**.

STEP 1: Allocate the demand, travel times and routes between the nodes based on the bus network studied. Assign the minimum and maximum frequencies for all routes. Initialize the parallel environment for communication between the master processor and slaves using MPI.

STEP 2: Divide the range of frequency for each route into m number of domains equally such that every domain (processor) has the same interval except the last domain which might be lower if m is not a factor of the frequency range. Choose a random frequency for all the routes to begin the passenger assignment procedure.

STEP 3: Allocate the demand to the routes based on the frequency obtained and their route choice behavior. Send the demand to all the slaves (processors).

STEP 4: Begin the MTS initialization for each slave processor. Assign the random frequencies within the domain boundaries to the routes and set it as the current solution. Represent the initial solutions as a vector of $X_n^0 = \left\{ f_{n1}^0, f_{n2}^0, \cdots, f_{mw}^0 \right\}$, such that w is the number of routes. Set the tabu list and intermediate-term memory as empty.

STEP 5: Increase or decrease the frequencies based on the step sizes to find the neighborhood of the current solutions. Check the feasibility, tabu restriction and dominance for each solution in the neighborhood.

STEP 6: If the solution satisfies the constraint, non tabu and dominates the previous current solution, save it in a set of non-dominated solutions, else if the solution is in tabu list, check aspiration criteria before saving the solution in the non-dominated set. Update the set by removing the worst solution each time after a solution is added. Choose a new solution randomly from the set to be assigned as the next current solution. Update the tabu list and intermediate-term memory.

```
BEGIN PROGRAM

p = number of processor
Variables and functions declaration.
Initialize parallel environment (MPI_Init)
If rank = 0 (master)
    compute travel times for all routes
    divide range of frequencies into p − 1 processors
    While the frequencies not converge
        passenger assignment procedure
        broadcast the demands and send the specific range of frequency to the slaves
        receive the best solutions from slave
        synchronize with other processors
    End While
    Return best solution
Else (slaves)
    While the frequencies not converge
        receive the specific frequency set for all routes
        do MTS procedure until no improvement in objective values
        send the best solution to master
        synchronize with other processors
    End While
Terminate parallel environment (MPI_Finalize)

END PROGRAM
```

Algorithm 1. PMTS for frequency optimization.

STEP 7: When no dominated solutions are available in the neighborhood, conduct the intensification process by choosing a solution from intermediate-term memory. Alternatively, if the intermediate-term memory is empty, select the least bad solution. Otherwise, initiate the diversification process. If there is no feasible solution in the neighborhood, restart the search from the feasible region.

STEP 8: Repeat **STEP 5 - 7** until there is no improvement in the best-known solution for a predefined consecutive iteration. Every slave processor produces a different set of frequency and sent to the master processor.

STEP 9: Check the convergence of the frequencies and if the pattern is not observed, repeat **STEP 3 - 8** until the frequency set converged. Else, stop the optimization procedure and record the best solution among the processors.

Following the output obtained from the frequency optimization problem, the timetabling process is initiated by considering two different scenarios such that the first scenario assigns equal departure times to both terminals of the routes to favor the passengers. Meanwhile, the second scenario considers operator's preference by allocating the times at the starting node only.

Based on the frequency set obtained from PMTS, the headway at each timeslot s of a route k is computed by dividing the total time period with the route's frequency per timeslot. The departure times for each route are calculated using the headways. Besides that, the one-way travel time at a timeslot is found by the summation of dwell time, vehicle travel time and layover time (assumed to be 10 percent of the travel time). The procedure to determine the departure times for each route are as follows:

STEP 1: Let $k = 1$.

STEP 2: Set $s = 1$. Set the first departure time to 300 minutes (5 a.m.) for route k.

STEP 3: Compute the headway of the route k and add the value to the first departure time in route k.

STEP 4: Continue adding the headway consecutively to its previous departure time and store the time in a set after every addition until the number of departure times at timeslot s equal to the frequency of route k.

STEP 5: Repeat **STEP 3** and **4** for all $s \in S$, where S is the set of timeslots. Let $k = k + 1$.

STEP 6: Repeat **STEP 2 - 5** for all $k \in R$, where R is the set of routes.

4.2. Parallel Multiple Tabu Search for Bus and Driver Scheduling

In this procedure, both buses and drivers are assigned simultaneously to the departure times by assuming that the drivers are assigned to the same bus throughout the working time. Besides, the vehicle blocks formed consist of long and short blocks such that the long blocks need two drivers and short blocks require only one driver which is deduced based on the duration of the blocks. Note that, the lengths of long blocks are equal to the time horizon studied in this research while the short blocks are reduced by half. The bus and driver scheduling are conducted consecutively for all the routes involved. The complete steps in this optimization process are discussed as follows. The pseudo-code of PMTS for bus and driver scheduling is given in Algorithm 2.

STEP 1: Initialize the parallel environment for communication between the master and slaves' processors using MPI. Assign the departure time and layover time at each timeslot for all the routes.

STEP 2: Determine the set of vehicle blocks that covers all the departure time and the number of drivers for each block. Divide the set of departure time into m number of domains equally such that every domain (processor) has the same number of departure time.

STEP 3: Begin MTS initialization for each slave processor. Select randomly the vehicle blocks that cover each departure time within the domain and set it as the current solution and best solution. Represent the initial solutions as a vector of $X_n^0 = \{b_{n1}^0, b_{n2}^0, \cdots, b_{nc}^0\}$, such that each vehicle

```
BEGIN PROGRAM
1.  p = number of processor, r = number of routes
2.  Variables and functions declaration
3.  Initialize parallel environment (MPI_Init)
4.  For i = 0 to r,
5.      If rank = 0 (master)
6.          set the departure time for the route i
7.          construct vehicle blocks that cover all the departure time
8.          set the number of driver for each block
9.          broadcast vehicle blocks, total drivers and other related data to the slaves
10.         send the departure times equally into p − 1 processors
11.         receive the best solutions from slaves
12.         eliminate similar blocks and reconstruct the candidate blocks to minimize further the objective values
13.         record the optimal solutions for route i
14.     Else (slaves)
15.         receive an array of departure times to be covered
16.         find initial blocks that satisfy the constraints
17.         Do
18.             TS procedure
19.         While no improvement in objective values for certain iteration
20.         send the best solution to master.
21.         synchronize every processors.
22. End for
23. Terminate parallel environment (MPI_Finalize)
END PROGRAM
```

Algorithm 2. PMTS for bus and driver scheduling.

block covers a subset of departure time, $b^0 = \{h_1^0, h_2^0, \cdots, h_e^0\}$ where c is the number of blocks included and e indicates the total departure time covered by each block.

STEP 4: Create two tabu lists to store the *add*, *drop* and *swap* moves. Set it as empty together with intermediate-term memory. Create a neighborhood of the current solution by adding, dropping and swapping corresponding blocks. Check each solution in the neighborhood for its feasibility, tabu restriction, and dominance.

STEP 5: If the solution satisfies the constraint, non-tabu, and dominates the previous current solution, record it in a set of non-dominated solutions, else if, the solution is in tabu list, conduct aspiration criteria before moving the solution to the non-dominated set. Update the set by removing the worst solution each time a solution is added. Then, randomly choose a new solution from the set to be assigned as the next current solution. Update the tabu lists and intermediate-term memory.

STEP 6: When no dominated solution is available in the neighborhood, conduct the intensification process by choosing a solution from intermediate-term memory. Alternatively, if the intermediate-term memory is empty, select the least bad solution. Otherwise, initiate the diversification process. When there is no feasible solution in the neighborhood, perform the constraint handling procedure to modify the neighborhood by adding the blocks that contain the uncovered departure times with another new block which is not included in the current solution and this trial solution.

STEP 7: Repeat **STEP 5 - 6** until there is no improvement in the best-known solution for a predefined iteration. Every slave processor produces a set of vehicle blocks and sends them to the master processor.

STEP 8: Combine all the blocks from each processor and remove the similar blocks. As the departure time can be covered by more than one block, eliminate its replicates from the blocks selected randomly. Reconstruct all the blocks to minimize further the number of buses and drivers and record the final solutions.

STEP 9: Repeat STEP 2 - 8 for all the routes in the network.

5. Results and Discussion

5.1. Benchmark Data and Experimental Design

The effectiveness of the proposed PMTS algorithm is assessed on benchmark Mandl's Swiss network (**Figure 3**) and Mumford's large network (**Figure 4**) based on the parameters given in **Table 1** using the following performance metrics as suggested by [30]: total number of buses, total waiting times, average route headways, and maximum route headways.

Computational experiments are conducted for several route sets and compared to various algorithms: heuristic by [28], [31], and [32]; GA with ant-system

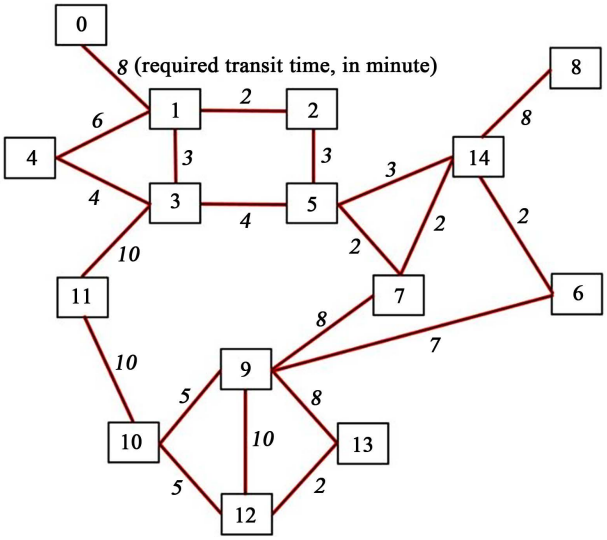


Figure 3. Mandl's Swiss network.

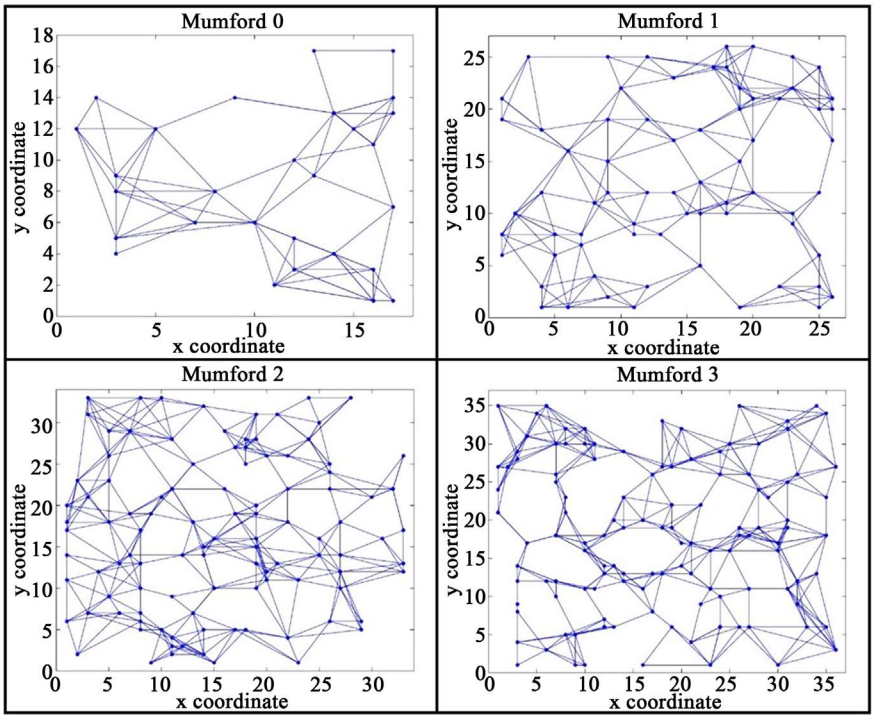


Figure 4. Mumford's network.

Table 1. Parameters for benchmark instances.

Instances	Nodes	Links	Routes	Min-Max nodes	Demand
Mandl	15	21	4 - 12	2 - 8	15,570
Mumford 0	30	90	12	2 - 15	342,160
Mumford 1	70	210	15	10 - 30	1,926,170
Mumford 2	110	385	56	10 - 22	4,487,900
Mumford 3	127	425	60	12 - 25	6,394,950

(GA-AS) by [33]; bee colony optimization (BCO) by [34]; GA by [30]; memetic algorithm (MA) by [35], differential evolution (DE) by [36] and sequential multiple tabu search (SMTS) by [3]. Some researchers also contributed their own route sets for this study together with other authors such as [37] [38] [39].

The proposed algorithm is verified with the respective passenger assignment methods and transfer penalties used by the previous authors. The PMTS algorithm is coded in ANSI-C language and executed on a Linux operating system using a High Performance Computer (HPC) system. It has an IBM System X3650 M4 Cluster operating on PUTRAGRID cluster distribution. The algorithm is executed for 10 runs to assess the robustness of the algorithm and each run is able to produce the results for all domains. The parameters for UTSP and PMTS used in this study are given in **Table 2**.

5.2. Experimental Results of PMTS

5.2.1. Frequency Optimization

Based on **Tables 3-8**, the first column indicates the source of the route sets of the Mandl's Swiss network benchmark data. The second column indicates the algorithms used to obtain the optimal results. The next five columns specified the performance metrics values mentioned earlier with the addition of overcrowding. Additionally, CPU execution time are shown to indicate the speed of the algorithm. The solution with lower number of buses, lesser total waiting times, lesser average and maximum headways without overcrowding is considered as the best solution.

The importance of the performance metrics is ranked from the number of buses followed by the total waiting times, average route headways and maximum route headways, consecutively. The solution with the lowest number of buses is given the highest priority to be chosen as the best among the 10 solutions produced. Note that, the average values for the number of buses and the total waiting time are rounded up to the nearest integer.

Table 2. Parameter configuration for UTSP and PMTS.

Description	Value
Transfer penalty for one transfer	5 minutes
Maximum number of transfers allowed	2 transfer/passenger
Bus capacity	50 passenger (40 seating capacity)
Load factor	1.25
Time horizon	1080 minutes (18 hours)
Minimum allowable frequency of buses on any route	18 (1 per hour)
Maximum allowable frequency of buses on any route	360 (20 per hour)
Consecutive iteration with non-improving solution	100
Tabu list size	2 × number of routes
Number of domains	10
Number of processors	10

Table 3. Comparison results for 4 routes (Mandl's Swiss network).

Source of route set	Algorithm	Number of buses	Total waiting Time (min)	Overcrowding	Maximum route headways (min)	Average route headways (min)	CPU time (sec)
[31]	[<1>]	103[a]	n/a	n/a	6.67[a]	3.48[a]	n/a
	[<2>]	99[b]	18,194[b]	n/a	n/a	n/a	n/a
	[<3>]	99[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	54[a]	27,563[a]	0[a]	3.14[a]	3.00[a]	59.445[a]
		54[b]	24,746[b]	0[b]	3.14[b]	3.07[b]	84.627[b]
	[<10>]	54[a]	27,398[a]	0[a]	3.15[a]	3.07[a]	1.919[a]
		54[b]	27,595[b]	0[b]	3.14[b]	3.07[b]	2.037[b]
	[<11>]	53[a]	28,591[a]	0[a]	3.19[a]	3.12[a]	1.318[a]
		54[b]	27,564[b]	0[b]	3.15[b]	3.07[b]	1.945[b]
[37]	[<1>]	105[a]	n/a	n/a	9.00[a]	4.06[a]	n/a
	[<9>]	80[a]	19,247[a]	0[a]	3.36[a]	3.30[a]	40.982[a]
	[<10>]	80[a]	19,610[a]	0[a]	3.46[a]	3.35[a]	1.709[a]
	[<11>]	79[a]	19,476[a]	0[a]	3.46[a]	3.35[a]	2.010[a]
[38]	[<1>]	86[a]	n/a	n/a	9.33[a]	4.43[a]	n/a
	[<9>]	79[a]	22,110[a]	0[a]	3.93[a]	3.78[a]	61.675[a]
	[<10>]	80[a]	22,183[a]	0[a]	3.96[a]	3.80[a]	2.545[a]
	[<11>]	77[a]	22,839[a]	0[a]	4.04[a]	3.91[a]	1.834[a]
[39]	[<1>]	87[a]	n/a	n/a	5.11[a]	3.64[a]	n/a
	[<9>]	86[a]	19,440[a]	0[a]	3.61[a]	3.50[a]	64.409[a]
	[<10>]	89[a]	19,050[a]	0[a]	3.49[a]	3.39[a]	2.232[a]
	[<11>]	86[a]	19,519[a]	0[a]	3.61[a]	3.50[a]	1.178[a]
[40]	[<1>]	94[a]	n/a	n/a	4.32[a]	3.41[a]	n/a
	[<9>]	86[a]	18,767[a]	0[a]	3.59[a]	3.42[a]	62.297[a]
	[<10>]	87[a]	18,914[a]	0[a]	3.48[a]	3.38[a]	2.936[a]
	[<11>]	87[a]	18,721[a]	0[a]	3.42[a]	3.37[a]	2.547[a]
[30]	[<1>]	79[a]	n/a	n/a	8.60[a]	4.60[a]	n/a
	[<9>]	76[a]	20,780[a]	0[a]	3.93[a]	3.77[a]	52.430[a]
	[<10>]	76[a]	20,807[a]	0[a]	3.92[a]	3.78[a]	1.434[a]
	[<11>]	74[a]	21,501[a]	0[a]	4.03[a]	3.90[a]	1.929[a]
[34]-(passenger)	[<4>]	94[b]	21,147[b]	n/a	n/a	n/a	n/a
	[<9>]	88[b]	19,489[b]	0[b]	3.47[b]	3.35[b]	43.805[b]
	[<10>]	88[b]	19,744[b]	0[b]	3.51[b]	3.40[b]	2.918[b]
	[<11>]	85[b]	20,132[b]	0[b]	3.62[b]	3.48[b]	2.777[b]
[34]-(operator)	[<1>]	67[b]	26,057[b]	n/a	n/a	n/a	n/a
	[<3>]	67[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	54[b]	24,711[b]	0[b]	4.50[b]	4.24[b]	48.886[b]
	[<10>]	53[b]	25,362[b]	0[b]	4.57[b]	4.35[b]	2.977[b]
	[<11>]	52[b]	25,644[b]	0[b]	4.66[b]	4.43[b]	1.369[b]
[36]	[<5>]	95[b]	24,098[b]	n/a	n/a	n/a	n/a
	[<9>]	86[b]	21,095[b]	0[b]	3.61[b]	3.41[b]	48.790[b]
	[<10>]	87[b]	21,149[b]	0[b]	3.55[b]	3.39[b]	2.091[b]
	[<11>]	86[b]	21,099[b]	0[b]	3.61[b]	3.38[b]	1.299[b]

Note: [<1>] = GA by [30]; [<2>] = Heuristic$_1$ by [31]; [<3>] = MA by [15]; [<4>] = BCO by [34]; [<5>] = DE by [36]; [<6>] = Heuristic$_2$ by [28]; [<7>] = Heuristic$_3$ by [32]; [<8>] = GA-AS by [33]; [<9>] = SMTS by [3]; [<10>] = proposed PMTS (average); [<11>] = proposed PMTS (best). a = Passenger assignment based on multinomial logit model; b = Passenger assignment based on frequency share rule; n/a = not available.

Table 4. Comparison results for 5 and 6 routes (Mandl's Swiss network).

Source of route set	Algorithm	Number of buses	Total waiting time (min)	Overcrowding	Maximum route headways (min)	Average route headways (min)	CPU time (sec)
5 routes							
[30]	[<1>]	75[a]	n/a	n/a	9.56[a]	5.39[a]	n/a
	[<9>]	64[a]	26,262[a]	0[a]	5.48[a]	5.26[a]	54.776[a]
	[<10>]	67[a]	25,446[a]	0[a]	5.32[a]	5.11[a]	3.789[a]
	[<11>]	64[a]	26,989[a]	0[a]	5.54[a]	5.36[a]	3.812[a]
6 routes							
[28]	[<6>]	89[b]	20,920[b]	n/a	n/a	n/a	n/a
	[<1>]	87[a]	n/a	n/a	11.33[a]	4.11[a]	n/a
	[<3>]	89[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	76[a] 76[b]	20,782[a] 19,558[b]	0[a] 0[b]	3.44[a] 3.44[b]	3.35[a] 3.35[b]	309.517[a] 345.220[b]
	[<10>]	77[a] 76[b]	20,798[a] 19,810[b]	0[a] 0[b]	3.42[a] 3.51[b]	3.34[a] 3.38[b]	3.415[a] 4.379[b]
	[<11>]	76[a] 75[b]	20,756 19,677[b]	0[a] 0[b]	3.44[a] 3.45[b]	3.34[a] 3.36[b]	3.235[a] 3.082[b]
[32]	[<7>]	84[b]	20,058[b]	n/a	n/a	n/a	n/a
	[<3>]	84[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	82[b]	19,869[b]	0[b]	3.04[b]	3.09[b]	161.875[b]
	[<10>]	81[b]	19,153[b]	0[b]	3.11[b]	3.05[b]	3.991[b]
	[<11>]	80[b]	19,070[b]	0[b]	3.11[b]	3.05[b]	3.701[b]
[38]	[<1>]	98[a]	n/a	n/a	8.00[a]	5.06[a]	n/a
	[<9>]	88[a]	18,433[a]	0[a]	5.24[a]	5.08[a]	188.846[a]
	[<10>]	89[a]	18,696[a]	0[a]	5.27[a]	5.03[a]	3.900[a]
	[<11>]	88[a]	18,310[a]	0[a]	5.12[a]	5.04[a]	3.207[a]
[39]	[<1>]	110[a]	n/a	n/a	8.00[a]	4.86[a]	n/a
	[<9>]	101[a]	17,457[a]	0[a]	4.74[a]	4.56[a]	272.325[a]
	[<10>]	104[a]	16,994[a]	0[a]	4.56[a]	4.36[a]	3.526[a]
	[<11>]	102[a]	16,812[a]	0[a]	4.62[a]	4.40[a]	5.375[a]
[40]	[<1>]	102[a]	n/a	n/a	10.29[a]	5.25[a]	n/a
	[<9>]	100[a]	18,147[a]	0[a]	4.70[a]	4.52[a]	145.578[a]
	[<10>]	104[a]	19,267[a]	0[a]	5.08[a]	4.78[a]	3.933[a]
	[<11>]	101[a]	17,846[a]	0[a]	4.70[a]	4.46[a]	3.712[a]
[34]-(passenger)	[<4>]	99[b]	21,766[b]	n/a	n/a	n/a	n/a
	[<9>]	98[b]	19,697[b]	0[b]	3.87[b]	3.76[b]	242.867[b]
	[<10>]	98[b]	19,909[b]	0[b]	3.98[b]	3.80[b]	3.448[b]
	[<11>]	98[b]	19,713[b]	0[b]	3.93[b]	3.78[b]	3.413[b]

Continued

[34]-(operator)	[<4>]	66[b]	31,500[b]	n/a	n/a	n/a	n/a
	[<3>]	66[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	61[b]	25,946[b]	0[b]	4.43[b]	4.27[b]	192.858[b]
	[<10>]	61[b]	26,305[b]	0[b]	4.55[b]	4.33[b]	4.283[b]
	[<11>]	60[b]	26,126[b]	0[b]	4.50[b]	4.30[b]	4.249[b]
[30]	[<1>]	77[a]	n/a	n/a	9.56[a]	6.42[a]	n/a
	[<9>]	63[a]	26,728[a]	0[a]	6.71[a]	6.31[a]	108.120[a]
	[<10>]	65[a]	26,678[a]	0[a]	6.69[a]	6.21[a]	4.976[a]
	[<11>]	64[a]	26,478[a]	0[a]	6.59[a]	6.22[a]	4.360[a]
[36]	[<5>]	92[b]	24,705[b]	n/a	n/a	n/a	n/a
	[<9>]	85[b]	23,091[b]	0[b]	5.57[b]	5.04[b]	74.578[b]
	[<10>]	86[b]	23,027[b]	0[b]	5.41[b]	5.06[b]	4.586[b]
	[<11>]	85[b]	22,915[b]	0[b]	5.29[b]	5.06[b]	3.523[b]

Table 5. Comparison results for 7 routes (Mandl's Swiss network).

Source of route set	Algorithm	Number of buses	Total waiting time (min)	Overcrowding	Maximum route headways (min)	Average route headways (min)	CPU time (sec)
[38]	[<1>]	102[a]	n/a	n/a	6.80[a]	5.32[a]	n/a
	[<9>]	101[a]	16,446[a]	0[a]	5.51[a]	5.25[a]	128.616[a]
	[<10>]	106[a]	16,200[a]	0[a]	5.46[a]	5.07[a]	4.789[a]
	[<11>]	102[a]	16,584[a]	0[a]	5.60[a]	5.27[a]	4.016[a]
[39]	[<1>]	110[a]	n/a	n/a	8.00[a]	4.86[a]	n/a
	[<9>]	108[a]	17,077[a]	0[a]	4.74[a]	4.50[a]	331.880[a]
	[<10>]	112[a]	16,460[a]	0[a]	4.55[a]	4.34[a]	3.999[a]
	[<11>]	111[a]	16,358[a]	0[a]	4.74[a]	4.33[a]	4.560[a]
[40]	[<1>]	98[a]	n/a	n/a	17.5[a]	7.00[a]	n/a
	[<9>]	82[a]	21,458[a]	0[a]	6.63[a]	6.13[a]	337.381[a]
	[<10>]	83[a]	21,623[a]	0[a]	6.52[a]	6.05[a]	3.482[a]
	[<11>]	81[a]	21,732[a]	0[a]	6.67[a]	6.13[a]	3.105[a]
[30]	[<1>]	77[a]	n/a	n/a	12.8[a]	7.58[a]	n/a
	[<9>]	76[a]	21,955[a]	0[a]	6.84[a]	6.12[a]	140.000[a]
	[<10>]	77[a]	22,044[a]	0[a]	6.70[a]	6.08[a]	5.930[a]
	[<11>]	75[a]	22,472[a]	0[a]	6.63[a]	6.19[a]	4.902[a]
[28]	[<6>]	82[b]	22,804[b]	n/a	n/a	n/a	n/a
	[<3>]	82[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	71[b]	23,901[b]	0[b]	3.20[b]	3.04[b]	245.449[b]
	[<10>]	71[b]	23,927[b]	0[b]	3.13[b]	3.04[b]	1.911[b]
	[<11>]	70[b]	24,142[b]	0[b]	3.20[b]	3.07[b]	1.733[b]

Continued

[34]-(passenger)	[<4>]	99[b]	23,157[b]	n/a	n/a	n/a	n/a
	[<9>]	96[b]	20,169[b]	0[b]	4.70[b]	4.44[b]	470.926[b]
	[<10>]	100[b]	19,455[b]	0[b]	4.58[b]	4.34[b]	6.556[b]
	[<11>]	97[b]	20,115[b]	0[b]	4.70[b]	4.49[b]	4.793[b]
[34]-(operator)	[<4>]	63[b]	35,481[b]	n/a	n/a	n/a	n/a
	[<3>]	63[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	63[b]	33,700[b]	0[b]	6.75[b]	6.38[b]	302.340[b]
	[<10>]	68[b]	31,363[b]	0[b]	6.50[b]	6.00[b]	5.280[b]
	[<11>]	66[b]	32,444[b]	0[b]	6.63[b]	6.11[b]	5.207[b]
[36]	[<5>]	90[b]	25,587[b]	n/a	n/a	n/a	n/a
	[<9>]	90[b]	25,584[b]	0[b]	6.67[b]	6.01[b]	132.450[b]
	[<10>]	88[b]	26,440[b]	0[b]	6.68[b]	6.19[b]	7.332[b]
	[<11>]	88[b]	26,141[b]	0[b]	6.63[b]	6.14[b]	7.433[b]

Table 6. Comparison results for 8 routes (Mandl's Swiss network).

Source of route set	Algorithm	Number of buses	Total waiting time (min)	Overcrowding	Maximum route headways (min)	Average route headways (min)	CPU time (sec)
[28]	[<6>]	77[b]	27,064[b]	n/a	n/a	n/a	n/a
	[<1>]	78[b]	n/a	n/a	10.67[b]	4.65[b]	n/a
	[<3>]	77[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	73[a] 72[b]	23,596[a] 22,200[b]	0[a] 0[b]	4.52[a] 4.60[b]	4.23[a] 4.31[b]	403.832[a] 371.028[b]
	[<10>]	71[a] 73[b]	24,415[a] 21,968[b]	0[a] 0[b]	4.70[a] 4.57[b]	4.44[a] 4.28[b]	6.287[a] 5.311[b]
	[<11>]	71[a] 73[b]	24,415[a] 21,827[b]	0[a] 0[b]	4.68[a] 4.52[b]	4.44[a] 4.23[b]	6.588[a] 4.871[b]
[32]	[<7>]	68[b]	26,455[b]	n/a	n/a	n/a	n/a
	[<3>]	68[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	62[b]	23,227[b]	0[b]	5.14[b]	4.98[b]	313.981[b]
	[<10>]	62[b]	23,231[b]	0[b]	5.16[b]	4.98[b]	6.790[b]
	[<11>]	62[b]	23,141[b]	0[b]	5.14[b]	4.96[b]	7.132[b]
[38]	[<1>]	101[a]	n/a	n/a	15.00[a]	6.91[a]	n/a
	[<9>]	96[a]	18,510[a]	0[a]	6.59[a]	6.09[a]	231.979[a]
	[<10>]	98[a]	18,708[a]	0[a]	6.58[a]	6.04[a]	7.286[a]
	[<11>]	96[a]	18,641[a]	0[a]	6.59[a]	6.11[a]	7.912[a]
[39]	[<1>]	88[a]	n/a	n/a	31.00[a]	9.67[a]	n/a
	[<9>]	86[a]	23,843[a]	0[a]	6.51[a]	6.02[a]	262.43[a]
	[<10>]	87[a]	24,206[a]	0[a]	6.55[a]	6.04[a]	4.910[a]
	[<11>]	86[a]	24,163[a]	0[a]	6.55[a]	6.04[a]	4.373[a]

Continued

[40]	[<1>]	104[a]	n/a	n/a	29.00[a]	9.66[a]	n/a
	[<9>]	95[a]	19,880[a]	0[a]	6.79[a]	6.15[a]	327.811[a]
	[<10>]	96[a]	20,224[a]	0[a]	6.64[a]	6.06[a]	6.821[a]
	[<11>]	96[a]	19,323[a]	0[a]	6.59[a]	5.99[a]	6.972[a]
[34]-(passenger)	[<4>]	99[b]	24,726[b]	n/a	n/a	n/a	n/a
	[<9>]	95[b]	20,804[b]	0[b]	6.35[b]	5.96[b]	397.550[b]
	[<10>]	94[b]	21,301[b]	0[b]	6.58[b]	6.09[b]	19.743[b]
	[<11>]	94[b]	21,133[b]	0[b]	6.47[b]	6.07[b]	12.890[b]
[34]-(operator)	[<4>]	63[b]	34,931[b]	n/a	n/a	n/a	n/a
	[<3>]	63[b]	n/a	n/a	n/a	n/a	n/a
	[<9>]	65[b]	25,479[b]	0[b]	6.51[b]	5.87[b]	519.204[b]
	[<10>]	65[b]	25,975[b]	0[b]	6.56[b]	5.94[b]	7.112[b]
	[<11>]	65[b]	25,708[b]	0[b]	6.43[b]	5.81[b]	6.780[b]
[30]	[<1>]	69[a]	n/a	n/a	10.33[a]	7.02[a]	n/a
	[<9>]	71[a]	23,803[a]	0[a]	6.84[a]	6.11[a]	156.901[a]
	[<10>]	73[a]	23,660[a]	0[a]	6.51[a]	6.07[a]	9.203[a]
	[<11>]	72[a]	23,602[a]	0[a]	6.47[a]	6.06[a]	8.329[a]
[36]	[<5>]	94[b]	25,487[b]	n/a	n/a	n/a	n/a
	[<9>]	93[b]	25,083[b]	0[b]	6.79[b]	6.11[b]	103.613[b]
	[<10>]	94[b]	25,405[b]	0[b]	6.62[b]	6.12[b]	4.992[b]
	[<11>]	94[b]	25,078[b]	0[b]	6.63[b]	6.01[b]	4.263[b]

Table 7. Comparison results for 9, 10 and 11 Routes (Mandl's Swiss network).

Source of route set	Algorithm	Number of buses	Total waiting time (min)	Overcrowding	Maximum route headways (min)	Average route headways (min)	CPU time (sec)
			9 routes				
[30]	[<1>]	66[a]	n/a	n/a	20.00[a]	9.14[a]	n/a
	[<9>]	58[a]	34,381[a]	0[a]	8.06[a]	7.34[a]	158.800[a]
	[<10>]	59[a]	34,527[a]	0[a]	8.22[a]	7.35[a]	10.180[a]
	[<11>]	58[a]	34,463[a]	0[a]	8.06[a]	7.31[a]	9.535[a]
			10 routes				
	[<1>]	72[a]	n/a	n/a	20.00[a]	9.44[a]	n/a
	[<9>]	81[a]	23,903[a]	0[a]	7.88[a]	7.36[a]	280.615[a]
	[<10>]	81[a]	24,184[a]	0[a]	8.51[a]	7.52[a]	10.108[a]
	[<11>]	80[a]	24,137[a]	0[a]	8.37[a]	7.51[a]	9.978[a]
			11 routes				
	[<1>]	68[a]	n/a	n/a	14.4[a]	8.76[a]	n/a
	[<9>]	67[a]	26,863[a]	0[a]	8.12[a]	7.56[a]	241.700[a]
	[<10>]	71[a]	26,040[a]	0[a]	8.07[a]	7.40[a]	14.492[a]
	[<11>]	68[a]	26,612[a]	0[a]	8.06[a]	7.51[a]	13.002[a]

Table 8. Comparison results for 12 routes (Mandl's Swiss network).

Source of route set	Algorithm	Number of buses	Total waiting time (min)	Overcrowding	Maximum route headways (min)	Average route headways (min)	CPU time (sec)
[33]	[<1>]	78[a]	n/a	n/a	11.33[a]	7.23[a]	n/a
	[<8>]	87[b]	24,951[b]	n/a	n/a	n/a	n/a
	[<9>]	73[a]	22,520[a]	0[a]	7.83[a]	7.24[a]	1282.227[a]
		70[b]	20,576[b]	0[b]	8.71[b]	7.63[b]	584.546[b]
	[<10>]	73[a]	23,032[a]	0[a]	8.16[a]	7.41[a]	13.636[a]
		73[b]	19,855[b]	0[b]	8.11[b]	7.44[b]	8.233[b]
	[<11>]	72[a]	22,933[a]	0[a]	8.06[a]	7.39[a]	12.974[a]
		72[b]	19,720[b]	0[b]	8.31[b]	7.36[b]	8.109[b]
[30]	[<1>]	73[a]	n/a	n/a	15.33[a]	10.58[a]	n/a
	[<9>]	62[a]	31,162[a]	0[a]	12.13[a]	10.01[a]	215.700[a]
	[<10>]	63[a]	31,307[a]	0[a]	11.34[a]	9.97[a]	13.400[a]
	[<11>]	61[a]	32,091[a]	0[a]	12.27[a]	10.32[a]	14.492[a]
[34]-(passenger)	[<4>]	98[b]	23,867[b]	n/a	n/a	n/a	n/a
	[<9>]	95[b]	24,675[b]	0[b]	6.71[b]	6.09[b]	846.067[b]
	[<10>]	97[b]	24,798[b]	0[b]	6.67[b]	6.10[b]	13.224[b]
	[<11>]	96[b]	24,588[b]	0[b]	6.84[b]	6.14[b]	12.092[b]
[34]-(operator)	[<4>]	65[b]	36,051[b]	n/a	n/a	n/a	n/a
	[<9>]	52[b]	33,828[b]	0[b]	11.61[b]	10.01[b]	1124.808[b]
	[<10>]	54[b]	34,102[b]	0[b]	11.92[b]	10.00[b]	17.096[b]
	[<11>]	52[b]	33,922[b]	0[b]	11.87[b]	10.19[b]	16.740[b]
[36]	[<5>]	88[b]	25,670[b]	n/a	n/a	n/a	n/a
	[<9>]	72[b]	42,468[b]	0[b]	12.13[b]	10.31[b]	321.287[b]
	[<10>]	74[b]	43,431[b]	0[b]	11.76[b]	10.00[b]	8.695[b]
	[<11>]	73[b]	42,304[b]	0[b]	12.27[b]	10.06[b]	7.822[b]

In the case of 4 routes (refer **Table 3**), the proposed PMTS algorithm able to minimize the number of buses, total waiting times, average route headways and maximum route headways produced by other authors for all the route sets except [31]. The total waiting times is higher as compared to [31], although the number of buses is reduced by almost half. The increase in waiting times may due to the lower upper bound value fixed for the frequency parameter in this study. A similar pattern can be observed for SMTS algorithm [3] but the CPU time is much higher as compared to the PMTS algorithm. Besides, the average values of total waiting times from all the route sets except [36] [37] [40] are lower than the best solution because there is a possibility to reduce the waiting times by increasing the number of buses scheduled. Note that, the solutions from the two passenger assignment models are also comparable to each other. For [37],

only route set of 4 is included for comparison because there are no previously published results are available considering the aforementioned performance criteria for 5 - 12 routes.

For 5 and 6 routes (see **Table 4**), PMTS algorithm improved all the performance metrics values mentioned earlier for all the previously published results. The CPU time for PMTS is reduced to more than 90 percent of the SMTS by [3] which shows the effectiveness of PMTS algorithm to compute faster. For the routes in [28], two passenger assignment methods using a multinomial logit model and frequency share rule are applied to compare with the respective solutions from the literature. Note that, all the best values of the number of buses and total waiting times are better than average solution even though there is some contradiction for average and maximum route headways.

In the case of 7 routes, **Table 5** shows that the proposed PMTS algorithm outperformed the solutions from [30] for their own route sets and the routes from [40]. The PMTS algorithm improved the solution of [30] based on the performance criteria for [34]-(passenger) and [38] but not in a Pareto sense. Similarly, the results from PMTS algorithm is similar to the values from [28] and [36] with the increase in total waiting times and also [34]-(operator) with more buses. Overall, the solutions of the PMTS algorithm for all the route sets are comparable to [3] but with much shorter CPU time. Moreover, all the best values of PMTS algorithm require lesser number of buses as compared to the average values except for [36].

In the case of 8 routes (see **Table 6**), the solutions from the proposed PMTS algorithm are better than the previous solutions for the routes from all the previous studies except [34]-(operator) and [30]. The number of buses produced by the proposed PMTS algorithms is higher but the total waiting time, the average and maximum route headways are lower for these studies. Alternatively, the proposed PMTS algorithm performed better than [3] for [32] only while producing equivalent results for the others. The values of all performance metrics for the best result are lower compared to average results for most of the route sets.

Furthermore, the proposed PMTS algorithm generates best results as compared to [30] for 9 and 11 routes (see **Table 7**). For route set of size 10, the results are improved in term of average and maximum route headways although the number of buses increased. The performance of PMTS algorithm is similar to the SMTS algorithm for 9 and 10 routes with slightly longer total waiting times and for 11 routes with a higher number of buses. By considering all the performance metric, the best results of PMTS are superior to its average results for 9 and 10 routes. The total waiting times of best solution are higher for 11 routes since the number of buses is reduced.

Based on **Table 8**, for 12 routes, our results are preferable as compared to [33], [34]-(operator), and [30] for their respective route sets. Alternatively, the results in this study are comparable with [30] for the routes from [33] and with [34]-(passenger) and [36] for their own route sets. The PMTS algorithm can

produce good solutions as SMTS algorithm for all route sets with short CPU time.

The proposed algorithm is further validated using an extended model considering different demands and travel times throughout the time period from 5.00 am to 11.00 pm. The route sets from [36] are experimented by implementing a multinomial logit model for passenger assignment procedure. The average solutions of every domain for the route sets of size 4, 6, 7, 8 and 12 are presented in Tables 9-13.

Table 9. Results obtained from the extended model for 4 routes.

Domain	Number of buses		Total waiting time (min)		Overcrowding	
	SMTS	PMTS	SMTS	PMTS	SMTS	PMTS
1	17	17	260,559	257,354	6034	6266
2	31	29	139,851	148,586	731	1150
3	44	43	99,934	101,785	0	0
4	58	56	76,488	79,624	0	0
5	70	70	62,809	62,720	0	0
6	83	83	52,725	52,908	0	0
7	97	97	45,625	45,779	0	0
8	109	110	40,419	40,554	0	0
9	122	123	36,097	35,782	0	0
10	136	135	29,406	32,695	0	0
CPU time (seconds)—390.992 sec (8.207 sec) * (PMTS)						

Table 10. Results obtained from the extended model for 6 routes.

Domain	Number of buses	Total waiting time (min)	Overcrowding
1	25	222,435	6264
2	45	119,891	491
3	62	85,340	0
4	83	65,256	0
5	100	54,322	0
6	120	44,568	0
7	142	38,335	0
8	159	34,109	0
9	180	29,993	0
10	196	27,719	0
CPU time (seconds)—7.314 sec			

Table 11. Results obtained from the extended model for 7 routes.

Domain	Number of buses	Total waiting time (min)	Overcrowding
1	29	216,488	5092
2	53	117,394	272
3	77	82,073	0
4	101	64,051	0
5	123	52,205	0
6	148	43,226	0
7	173	37,473	0
8	194	33,684	0
9	219	29,475	0
10	240	27,118	0
CPU time (seconds)—8.764 sec			

Table 12. Results obtained from the extended model for 8 routes.

Domain	Number of buses	Total waiting time (min)	Overcrowding
1	32	200,618	2682
2	57	95,470	38
3	81	74,298	0
4	107	57,785	0
5	133	46,744	0
6	158	38,892	0
7	184	33,760	0
8	209	26,226	0
9	236	26,226	0
10	258	24,199	0
CPU time (seconds)—9.750 sec			

Table 13. Results obtained from the extended model for 12 routes.

Domain	Number of buses	Total waiting time (min)	Overcrowding
1	39	191,747	2886
2	73	101,086	166
3	103	72,525	0
4	135	56,855	0
5	170	44,889	0
6	198	38,456	0
7	232	32,881	0
8	263	29,179	0
9	297	25,794	0
10	325	23,678	0
CPU time (seconds)—10.690 sec			

Generally, the round trip time of a bus affects the number of buses required. Thus, the total buses needed are high as compared to the previous model (without timeslot) since the layover time and dwell time increase the round trip time for a trip. Moreover, as the frequency for peak and off-peak hours is different, the total waiting time also increases. Based on **Tables 9-13**, there is no overcrowding in the solutions from domain 3 to 10 and domain 3 has the least number of buses among them. Thus, it is more preferable to be chosen for studying bus and driver scheduling problem in the next subsection.

Referring to **Table 9**, the PMTS algorithm is compared with SMTS of [3] and it shows similar results but with much lesser CPU time. Since the previous study does not include the solutions from other route sets (6, 7, 8 and 12), the comparison between SMTS and PMTS results for the route sets are not available.

Table 10 presents the values for each objective functions with respect to the domains. The first domain which carries lower frequency needs lesser buses butlonger waiting time with high overcrowding as compared to the last domain. The number of passengers exceeds the maximum load factor at some time period due to inadequate frequency.

Based on **Table 11**, the frequency and number of buses required increases while the total waiting time and overcrowding decreases which reflects the conflicting nature of the objectives. The total buses are up to 240 with 27,118 minutes of waiting time for the headways around 3 minutes from domain 10.

According to **Table 12**, around 32 buses are needed to satisfy the frequency from domain 1 with respective waiting time and overcrowding. Meanwhile, domain 10 carries the highest number of buses (258) with the least passengers waiting time without overcrowding.

Table 13 displays the 10 non-dominated solutions from all the domains respectively. The highest number of buses is 325 with 23,678 minutes of waiting time. Alternatively, the lowest number of buses required for 12 routes is 39 with 191,747 minutes of total waiting time and 2886 passengers who exceed the fixed load factor of the bus.

On the other hand, the proposed algorithm is also tested for Mumford's network using the route sets from [41]. Mumford 0, Mumford 1, Mumford 2 and Mumford 3 are different in term of the number of nodes and links, route size and total demands. These parameter values are increasing according to the instances from 0 to 3 such that Mumford 3 has a more complex network and highest number of routes. The average results of 10 runs for each domain are presented in **Table 14**. Since there is no previous solution for Mumford's network considering the number of buses, total waiting time, overcrowding, average route headways, and maximum route headways; the comparison of the results is not possible. Therefore, the values from all the 10 domains are presented.

The computational time for Mumford's network is quite long which is around 4 days for each run. This is due to the passenger assignment procedure that has to be conducted at every iteration to calculate the total waiting times. Moreover, as the number of routes increases, the CPU time also increases.

Table 14. Average solutions of 10 domains for mumford's network.

Instance	Domain	Number of buses	Total waiting time (min)	Overcrowding	Average route headways (min)	Maximum route headways (min)
Mumford 0	1	46	3,823,981	177,092	33.35	50.22
	2	89	1,869,231	155,882	15.35	19.00
	3	130	1,298,821	135,873	11.85	9.37
	4	171	992,384	106,623	7.74	8.53
	5	210	799,953	99,223	6.27	6.81
	6	261	656,328	80,128	5.07	5.25
	7	291	575,334	69,678	4.54	4.93
	8	337	499,331	54,761	3.91	4.07
	9	378	443,829	45,167	3.45	3.52
	10	412	408,721	36,884	3.13	3.22
Mumford 1	1	152	27,272,231	889,400	32.11	51.92
	2	251	1,038,267	865,813	16.81	20.32
	3	380	6,619,500	838,356	11.68	12.52
	4	505	5,451,290	894,010	8.71	9.55
	5	635	4,351,790	783,128	5.25	5.84
	6	754	3,648,891	789,003	6.18	6.45
	7	890	3,120,588	731,700	3.87	4.15
	8	1010	2,728,672	776,742	3.68	3.97
	9	1257	2,440,521	679,423	3.16	3.37
	10	1255	2,204,810	654,888	3.09	3.13
Mumford 2	1	363	70,175,671	2,383,652	34.51	59.08
	2	703	32,236,820	2,283,577	15.71	20.12
	3	1041	21,477,720	2,186,647	10.40	12.23
	4	1387	15,942,405	2,086,622	6.96	7.85
	5	1729	12,788,639	1,989,147	6.18	6.77
	6	2070	10,675,905	1,889,902	5.14	5.56
	7	2413	9,159,933	1,792,892	4.41	4.73
	8	2748	8,008,142	1,696,337	3.48	3.69
	9	3093	7,126,956	1,598,492	3.43	3.60
	10	3395	6,489,612	1,514,318	3.43	3.56
Mumford 3	1	428	74,000,032	3,235,742	36.04	60.00
	2	867	33,544,319	3,123,442	15.79	20.38
	3	1299	22,491,812	3,017,792	10.35	12.27
	4	1724	16,909,705	2,910,592	7.70	8.64
	5	2143	13,579,161	2,808,442	6.19	6.79
	6	2557	11,322,699	2,703,492	5.15	5.48
	7	2987	9,693,081	2,597,763	4.40	4.68
	8	3413	8,477,324	2,493,563	3.85	4.11
	9	3816	7,627,143	2,401,554	3.46	3.61
	10	4210	6,904,463	2,308,445	3.13	3.22

5.2.2. Bus and Driver Scheduling

As mentioned earlier in Section 3.2, two different scenarios favoring passengers (Scenario 1, S1) and operators (Scenario 2, S2) are studied regarding the allocation of the departure times at the origin and destination of a route. Each scenario is performed for 10 runs and the average and best solution (marked with "*") are recorded. The solution with a lower number of buses compared to the average value is marked as best although the number of drivers is higher. This is because fewer buses indicate more long blocks are produced which consequently need more drivers. The performance of the proposed PMTS algorithm is compared with SMTS by [3]. The input data such as frequency and one-away travel time of the routes are obtained using the solutions of domain 3 in **Tables 9-13**.

Based on **Table 15** and **Table 16**, the total buses and drivers are greater for S1 as compared to S2 since the departure times to be covered are higher for the former. Both scenarios are equivalent to each other by considering different perspectives of passengers and operators respectively. It provides more choices for the decision makers to implement in a real system. The solution from S1 is presented as an example to show the ability of reconstruction procedure to allocate more departure times for buses and drivers. On the other hand, there is no significant difference between the effectiveness of SMTS and PMTS algorithms except the CPU times is reduced to more than 50 percent by the parallel implementation.

The bus and driver schedule of S1 for 4 routes from [36] are presented in Appendix A (**Tables A1-A4**). It shows the bus number and drivers that cover the

Table 15. Results for S1.

Source of route set	Routes	Algorithm	Total buses	Total drivers	CPU time (sec)
[36]	4	SMTS	106	153	9.000
		PMTS	112	147	2.334
		PMTS*	106	154	2.541
	6	SMTS	148	203	14.000
		PMTS	147	202	3.567
		PMTS*	146	200	3.450
	7	SMTS	191	254	20.000
		PMTS	190	251	4.871
		PMTS*	188	258	5.124
	8	SMTS	202	268	19.000
		PMTS	205	272	5.881
		PMTS*	203	265	5.723
	12	SMTS	256	349	27.000
		PMTS	267	345	7.812
		PMTS*	262	348	7.923

Table 16. Results for S2.

Source of route set	Routes	Algorithm	Total buses	Total drivers	CPU time (sec)
[36]	4	SMTS	98	134	4.000
		PMTS	99	133	1.228
		PMTS*	98	130	1.932
	6	SMTS	134	179	4.000
		PMTS	135	183	2.078
		PMTS*	131	179	2.138
	7	SMTS	165	226	5.000
		PMTS	177	225	2.356
		PMTS*	176	217	2.348
	8	SMTS	179	239	6.000
		PMTS	193	238	3.223
		PMTS*	181	227	3.412
	12	SMTS	234	310	8.000
		PMTS	231	314	2.453
		PMTS*	228	314	2.378

respective departure times in the time period at the first stop (origin) and last stop (destination) of a route. Based on Table A1, the total number of buses and drivers needed for the first route are 23 and 34 respectively. Each departure time is assigned to a specific bus and driver according to the travel time and working period.

Table A2 shows that route 2 require 38 buses and 57 drivers to cover all the departure time. The buses and drivers are allocated according to the type of blocks. Table A3 indicates the buses and drivers that work at every departure time to ensure effective distribution of the resources. The schedule consists of 22 buses and 29 drivers. Similarly, Table A4 presents the sequence of departure times from the perspective of passengers for a time period of 18 hours. A total of 23 buses and 34 drivers are allocated to the times.

6. Conclusions

In this paper, a procedure for solving UTSP which consists of frequency optimization, timetabling, and bus and driver scheduling is proposed. A mixed integer multiobjective model is constructed to optimize the frequency of the routes by minimizing the number of buses, passenger's waiting times and overcrowding by considering the preferences of passengers and operators to find the optimal solution. The model is further extended by including timeslots to find the frequencies during peak and off-peak hours throughout the time period.

The main contribution of this paper is the development of PMTS algorithm by modifying the initialization process and incorporating intensification and diver-

sification approaches to guide the search effectively in order to obtain better solutions. The efficiency of the proposed PMTS algorithm is tested on Mandl's benchmark datasets using the route sets published from the literature. The results indicate that the frequency set obtained, improved the number of buses and total waiting times in most cases from the literature. Moreover, the extended model which includes timeslot produced a higher number of buses and longer waiting times since it includes dwelling time, layover times and generates different frequency according to the demands and travel times during the time period.

Acknowledgements

This research was supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2016/STG06/UPM/02/8).

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] Ceder, A. (2002) Urban Transit Scheduling: Framework, Review and Examples. *Journal of Urban Planning and Development*, **128**, 225-244. https://doi.org/10.1061/(ASCE)0733-9488(2002)128:4(225)

[2] Pothiya, S., Ngamroo, I., Runggeratigul, S. and Tantaswadi, P. (2006) Design of Optimal Fuzzy Logic Based Pi Controller Using Multiple Tabu Search Algorithm for Load Frequency Control. *International Journal of Control, Automation and Systems*, **4**, 155-164.

[3] Uvaraja, V. and Lee, L.S. (2019) Multiple Tabu Search for Multiobjective Urban Transit Scheduling Problem. *ASM Science Journal*, **12**, 150-173.

[4] Desaulniers, G. and Hickman, M.D. (2007) Public Transit. In: *Handbooks in Operation Research and Management Science*: *Transportation*, *Volume* 14, Elsevier, North Holland, 69-120. https://doi.org/10.1016/S0927-0507(06)14002-5

[5] Guihaire, V. and Hao, J.K. (2008) Transit Network Design and Scheduling: A Global Review. *Transportation Research Part A*: *Policy and Practice*, **42**, 1251-1273. https://doi.org/10.1016/j.tra.2008.03.011

[6] Ceder, A. (2016) Public Transit Planning and Operation: Modeling, Practice and Behavior. CRC Press, Boca Raton, FL. https://doi.org/10.1201/b18689

[7] Uvaraja, V. and Lee, L.S. (2017) Metaheuristic Approaches for Urban Transit Scheduling Problem: A Review. *Journal of Advanced Review on Scientific Research*, **34**, 11-25.

[8] Lourenço, H.R., José P.P. and Rita, P. (2001) Multiobjective Metaheuristics for the Bus Driver Scheduling Problem. *Transportation Science*, **35**, 331–343. https://doi.org/10.1287/trsc.35.3.331.10147

[9] Agrawal, J. and Mathew, T.V. (2004) Transit Route Network Design Using Parallel Genetic Algorithm. *Journal of Computing in Civil Engineering*, **18**, 248-256. http://doi.org/10.1061/(ASCE)0887-3801(2004)18:3(248)

[10] Yu, B., Yang, Z., Cheng, C. and Liu, C. (2005) Optimizing Bus Transit Network with

Parallel Ant Colony Algorithm. *Proceedings of the Eastern Asia Society for Transportation Studies*, **5**, 374-389.

[11] Yang, Z., Yu, B. and Cheng, C. (2007) A Parallel Ant Colony Algorithm for Bus Network Optimization. *Computer-Aided Civil and Infrastructure Engineering*, **22**, 44-55. https://doi.org/10.1111/j.1467-8667.2006.00469.x

[12] Li, J.Q., Mirchandani, P.B. and Borenstein, D. (2008) Parallel Auction Algorithm for Bus Rescheduling. In: *Computer-Aided Systems in Public Transport*, Springer, Berlin, Heidelberg, 281-299. https://doi.org/10.1007/978-3-540-73312-6_14

[13] Yu, B., Yang, Z., Sun, X., Yao, B., Zeng, Q. and Jeppesen, E. (2011) Parallel Genetic Algorithm in Bus Route Headway Optimization. *Applied Soft Computing*, **11**, 5081-5091. https://doi.org/10.1016/j.asoc.2011.05.051

[14] Cipriani, E., Gori, S. and Petrelli, M. (2012) Transit Network Design: A Procedure and an Application to a Large Urban Area. *Transportation Research Part C*: *Emerging Technologies*, **20**, 3-14. https://doi.org/10.1016/j.trc.2010.09.003

[15] Zuo, X., Chen, C., Tan, W. and Zhou, M. (2015) Vehicle Scheduling of an Urban Bus Line via an Improved Multiobjective Genetic Algorithm. *IEEE Transactions on Intelligent Transportation Systems*, **16**, 1030-1041.

[16] Wu, Y., Yang, H., Tang, J. and Yu, Y. (2016) Multi-Objective Re-Synchronizing of Bus Timetable: Model, Complexity and Solution. *Transportation Research Part C*: *Emerging Technologies*, **67**, 149-168. https://doi.org/10.1016/j.trc.2016.02.007

[17] Peña, D., Tchernykh, A., Nesmachnow, S., Massobrio, R., Feoktistov, A. and Bychkov, I. (2017) Multiobjective Vehicle-Type Scheduling in Urban Public Transport. In: *IEEE International Parallel and Distributed Processing Symposium Workshops*, 482-491. https://doi.org/10.1109/IPDPSW.2017.80

[18] López-Ramos, F., Codina, E., Marín, Á. and Guarnaschelli, A. (2017) Integrated Approach to Network Design and Frequency Setting Problem in Railway Rapid Transit Systems. *Computers and Operations Research*, **80**, 128-146. https://doi.org/10.1016/j.cor.2016.12.006

[19] Ruano-Daza, E., Cobos, C., Torres-Jimenez, J., Mendoza, M. and Paz, A. (2018) A Multiobjective Bilevel Approach Based on Global-Best Harmony Search for Defining Optimal Routes and Frequencies for Bus Rapid Transit Systems. *Applied Soft Computing*, **67**, 567-583. https://doi.org/10.1016/j.asoc.2018.03.026

[20] Scheele, S. (1980) A Supply Model for Public Transit Services. *Transportation Research Part B*, **14**, 133-146. https://doi.org/10.1016/0191-2615(80)90039-9

[21] Constantin, I. and Florian, M. (1995) Optimizing Frequencies in a Transit Network: A Nonlinear Bi-Level Programming Approach. *International Transactions in Operational Research*, **2**, 149-164. https://doi.org/10.1111/j.1475-3995.1995.tb00011.x

[22] Niu, H. (2011) Determination of the Skip-Stop Scheduling for a Congested Transit Line by Bilevel Genetic Algorithm. *International Journal of Computational Intelligence Systems*, **4**, 1158-1167. https://doi.org/10.2991/ijcis.2011.4.6.7

[23] Verbas, I. and Mahmassani, H.S. (2015) Exploring Trade-Offs in Frequency Allocation in a Transit Network Using Bus Route Patterns: Methodology and Application to Large Scale Urban Systems. *Transportation Research Part B*: *Methodological*, **81**, 577-595. https://doi.org/10.1016/j.trb.2015.06.018

[24] Li, Y., Xu, W. and He, S. (2013) Expected Value Model for Optimizing the Multiple Bus Headways. *Applied Mathematics and Computation*, **219**, 5849-5861. https://doi.org/10.1016/j.amc.2012.11.098

[25] Jiang, X., Guo, X. and Ran, B. (2014) Optimization Model for Headway of a Suburban Bus Route. *Mathematical Problems in Engineering*, **2014**. Article ID: 979062. https://doi.org/10.1155/2014/979062

[26] Fedorko, G. and Weiszer, M. (2012) Configuring the Parameters of Multi-Objective Evolutionary Algorithm for Integrated Timetabling and Vehicle Scheduling in Public Transport. In: *Carpathian Logistics Congress* (*CLC*) *Proceeding*, 435-440.

[27] Prata, B.D.A. (2016) A Multiobjective Metaheuristic Approach for the Integrated Vehicle and Crew Scheduling. *Journal of Transport Literature*, **10**, 10-14. https://doi.org/10.1590/2238-1031.jtl.v10n2a2

[28] Baaj, M.H. and Mahmassani, H.S. (1991) An AI-based Approach for Transit Route System Planning and Design. *Journal of Advanced Transportation*, **25**, 187-209. https://doi.org/10.1002/atr.5670250205

[29] Afandizadeh, S., Hassan, K. and Narges, K. (2013) Bus Fleet Optimization using Genetic Algorithm a Case Study of Mashhad. *International Journal of Civil Engineering*, **11**, 43-52.

[30] Arbex, R.O. and da Cunha, C.B. (2015) Efficient Transit Network Design and Frequencies Setting Multi-Objective Optimization by Alternating Objective Genetic Algorithm. *Transportation Research Part B*: *Methodological*, **81**, 355-376. https://doi.org/10.1016/j.trb.2015.06.014

[31] Mandl, C. (1980) Evaluation and Optimization of Urban Public Transportation Networks. *European Journal of Operational Research*, **5**, 396-404. https://doi.org/10.1016/0377-2217(80)90126-5

[32] Shih, M.C. and Mahmassani, H.S. (1994) A Design Methodology for Bus Transit Networks with Coordinated Operations. Technical Report.

[33] Bagloee, S.A. and Ceder, A.A. (2011) Transit-Network Design Methodology for Actual-Size Road Networks. *Transportation Research Part B*: *Methodological*, **45**, 1787-1804. https://doi.org/10.1016/j.trb.2011.07.005

[34] Nikolić, M. and Teodorović, D. (2014) A Simultaneous Transit Network Design and Frequency Setting: Computing with Bees. *Expert Systems with Applications*, **41**, 7200-7209. https://doi.org/10.1016/j.eswa.2014.05.034

[35] Zhao, H., Xu, W. and Jiang, R. (2015) The Memetic Algorithm for the Optimization of Urban Transit Network. *Expert Systems with Applications*, **42**, 3760-3773. https://doi.org/10.1016/j.eswa.2014.11.056

[36] Buba, A.T. and Lee, L.S. (2018) A Differential Evolution for Simultaneous Transit Network Design and Frequency Setting Problem. *Expert Systems with Applications*, **106**, 277-289. https://doi.org/10.1016/j.eswa.2018.04.011

[37] Chakroborty, P. (2003) Genetic Algorithms for Optimal Urban Transit Network Design. *Computer-Aided Civil and Infrastructure Engineering*, **18**, 184-200. https://doi.org/10.1111/1467-8667.00309

[38] Mumford, C.L. (2013) New Heuristic and Evolutionary Operators for the Multi-Objective Urban Transit Routing Problem. In: *IEEE Congress of Evolutionary Computation*, 939-946. https://doi.org/10.1109/CEC.2013.6557668

[39] Chew, J.S.C., Lee, L.S. and Seow, H.V. (2013) Genetic Algorithm for Biobjective Urban Transit Routing Problem. *Journal of Applied Mathematics*, **2013**, Article ID: 698645. https://doi.org/10.1155/2013/698645

[40] Nikolić, M. and Teodorović, D. (2013) Transit Network Design by Bee Colony Optimization. *Expert Systems with Applications*, **40**, 5945-5955.

https://doi.org/10.1016/j.eswa.2013.05.002

[41] Buba, A.T. and Lee, L.S. (2018b) Differential Evolution with Improved Sub-Route Reversal Repair Mechanism for Multiobjective Urban Transit Routing Problem. *Numerical Algebra, Control & Optimization*, **8**, 361-386. https://doi.org/10.3934/naco.2018023

Appendix A

Table A1. Schedule for route #1 of 4 routes.

Origin to Destination			Destination to Origin			Origin to Destination			Destination to Origin		
Time	Bus No.	Driver No.	Time	Bus No.	Driver No	Time	Bus No.	Driver No	Time	Bus No.	Driver No.
5:00	16	22	5:00	20	28	13:51	2	2	13:51	3	4
5:15	17	24	5:15	3	4	14:00	20	28	14:00	11	14
5:30	1	1	5:30	21	30	14:15	8	11	14:15	22	32
5:45	2	2	5:45	14	19	14:30	18	26	14:30	1	1
6:00	20	28	6:00	16	22	14:45	3	5	14:45	2	2
6:15	6	8	6:15	17	24	15:00	11	14	15:00	12	16
6:30	21	30	6:30	22	32	15:15	22	32	15:15	8	11
6:45	14	19	6:45	2	2	15:30	23	34	15:30	18	26
7:00	16	22	7:00	20	28	15:45	15	21	15:45	3	5
7:08	17	24	7:08	6	8	16:00	12	16	16:00	11	14
7:17	22	32	7:17	21	30	16:08	8	11	16:08	22	33
7:25	4	6	7:25	18	26	16:17	18	26	16:17	23	34
7:34	2	2	7:34	14	19	16:25	10	13	16:25	5	7
7:42	12	16	7:42	19	27	16:34	3	5	16:34	15	21
7:51	20	28	7:51	16	22	16:42	16	23	16:42	9	12
8:00	6	8	8:00	17	24	16:51	11	14	16:51	12	17
8:08	21	30	8:08	22	32	17:00	22	33	17:00	7	9
8:17	18	26	8:17	4	6	17:08	23	34	17:08	20	29
8:25	14	19	8:25	2	2	17:17	5	7	17:17	10	13
8:34	5	7	8:34	12	16	17:25	15	21	17:25	14	20
8:42	16	22	8:42	20	28	17:34	9	12	17:34	16	23
8:51	17	24	8:51	6	8	17:42	12	17	17:42	11	14
9:00	22	32	9:00	21	30	17:51	7	9	17:51	22	33
9:09	4	6	9:09	18	26	18:00	13	18	18:00	8	11
9:17	2	2	9:17	14	19	18:08	10	13	18:08	3	5
9:25	12	16	9:25	5	7	18:17	14	20	18:17	15	21
9:34	3	4	9:34	1	1	18:25	16	23	18:25	9	12
9:42	6	8	9:42	11	14	18:34	11	14	18:34	12	17
9:51	7	9	9:51	22	32	18:42	22	33	18:42	7	9
10:00	18	26	10:00	4	6	18:51	8	11	18:51	13	18
10:15	5	7	10:15	12	16	19:00	3	5	19:00	10	13
10:30	20	28	10:30	16	22	19:08	15	21	19:08	21	31
10:45	19	27	10:45	17	24	19:17	9	12	19:17	16	23
11:00	21	30	11:00	18	26	19:25	12	17	19:25	11	15
11:15	12	16	11:15	2	2	19:34	7	10	19:34	22	33
11:30	11	14	11:30	20	28	19:42	13	18	19:42	8	11

Continued

11:45	22	32	11:45	7	9	19:51	10	13	19:51	3	5
12:00	1	1	12:00	21	30	20:00	17	25	20:00	23	34
12:08	2	2	12:08	14	19	20:15	2	3	20:15	12	17
12:17	20	28	12:17	11	14	20:30	8	11	20:30	13	18
12:25	8	11	12:25	6	8	20:45	3	5	20:45	9	12
12:34	7	9	12:34	22	32	21:00	21	31	21:00	15	21
12:42	17	24	12:42	13	18	21:15	14	20	21:15	2	3
12:51	18	26	12:51	1	1	21:30	22	33	21:30	7	10
13:00	14	19	13:00	2	2	21:45	9	12	21:45	10	13
13:08	11	14	13:08	20	28	22:00	15	21	22:00	21	31
13:17	6	8	13:17	8	11	22:15	12	17	22:15	14	20
13:25	22	32	13:25	7	9	22:30	7	10	22:30	22	33
13:34	21	30	13:34	17	24	22:45	10	13	22:45	17	25
13:42	1	1	13:42	18	26						

Table A2. Schedule for route #2 of 4 routes.

Origin to Destination			Destination to Origin			Origin to Destination			Destination to Origin		
Time	Bus No.	Driver No.	Time	Bus No.	Driver No.	Time	Bus No.	Driver No.	Time	Bus No.	Driver No.
5:00	4	5	5:00	22	30	13:46	21	28	13:46	36	53
5:12	16	20	5:12	3	4	13:53	35	51	13:53	18	24
5:24	17	22	5:24	27	37	14:00	26	36	14:00	33	48
5:36	30	42	5:36	21	28	14:12	17	22	14:12	8	10
5:48	9	11	5:48	31	44	14:24	27	37	14:24	6	8
6:00	10	13	6:00	29	40	14:36	23	32	14:36	15	19
6:12	22	30	6:12	37	55	14:48	29	40	14:48	22	31
6:24	3	4	6:24	16	20	15:00	36	53	15:00	35	51
6:36	27	37	6:36	8	10	15:12	33	48	15:12	21	29
6:48	21	28	6:48	30	42	15:24	8	10	15:24	17	23
7:00	31	44	7:00	9	11	15:36	6	8	15:36	27	38
7:06	29	40	7:06	10	13	15:48	19	26	15:48	23	32
7:13	36	53	7:13	1	1	16:00	31	45	16:00	29	41
7:20	37	55	7:20	22	30	16:06	35	51	16:06	36	53
7:26	33	48	7:26	34	49	16:13	12	16	16:13	7	9
7:33	16	20	7:33	3	4	16:20	21	29	16:20	33	48
7:40	8	10	7:40	27	37	16:26	13	17	16:26	26	36
7:46	5	7	7:46	32	46	16:33	17	23	16:33	16	21
7:53	30	42	7:53	21	28	16:40	27	38	16:40	5	7
8:00	6	8	8:00	23	32	16:46	28	39	16:46	38	57
8:06	9	11	8:06	31	44	16:53	23	32	16:53	19	26
8:13	10	13	8:13	29	40	17:00	14	18	17:00	30	43

Continued

8:20	35	51	8:20	36	53	17:06	29	41	17:06	31	45
8:26	22	30	8:26	37	55	17:13	37	56	17:13	35	51
8:33	18	24	8:33	33	48	17:20	7	9	17:20	12	16
8:40	3	4	8:40	16	20	17:26	1	2	17:26	21	29
8:46	27	37	8:46	8	10	17:33	26	36	17:33	13	17
8:53	32	46	8:53	5	7	17:40	16	21	17:40	17	23
9:00	21	28	9:00	30	42	17:46	2	3	17:46	27	38
9:06	23	32	9:06	6	8	17:53	18	25	17:53	28	39
9:13	31	44	9:13	9	11	18:00	19	26	18:00	23	33
9:20	29	40	9:20	10	13	18:06	30	43	18:06	14	18
9:26	36	53	9:26	35	51	18:13	31	45	18:13	29	41
9:33	37	55	9:33	2	3	18:20	35	52	18:20	37	56
9:40	33	48	9:40	18	24	18:26	12	16	18:26	7	9
9:46	7	9	9:46	20	27	18:33	21	29	18:33	24	34
9:53	8	10	9:53	17	22	18:40	13	17	18:40	26	36
10:00	5	7	10:00	32	46	18:46	17	23	18:46	20	28
10:12	6	8	10:12	23	32	18:53	32	47	18:53	25	35
10:24	9	11	10:24	31	44	19:00	28	39	19:00	18	25
10:36	35	51	10:36	36	53	19:06	23	33	19:06	19	26
10:48	16	20	10:48	3	4	19:13	14	18	19:13	30	43
11:00	20	27	11:00	27	37	19:20	22	31	19:20	11	15
11:12	30	42	11:12	5	7	19:26	15	19	19:26	35	52
11:24	23	32	11:24	29	40	19:33	10	14	19:33	1	2
11:36	2	3	11:36	37	55	19:40	24	34	19:40	34	50
11:48	26	36	11:48	35	51	19:46	26	36	19:46	16	21
12:00	17	22	12:00	8	10	19:53	27	38	19:53	12	16
12:06	27	37	12:06	20	27	20:00	25	35	20:00	32	47
12:13	3	4	12:13	6	8	20:12	18	25	20:12	23	33
12:20	28	39	12:20	30	42	20:24	29	41	20:24	31	45
12:26	11	15	12:26	15	19	20:36	35	52	20:36	21	29
12:33	29	40	12:33	22	30	20:48	30	43	20:48	13	17
12:40	36	53	12:40	21	28	21:00	12	16	21:00	27	38
12:46	18	24	12:46	9	11	21:12	19	26	21:12	4	6
12:53	33	48	12:53	26	36	21:24	36	54	21:24	18	25
13:00	5	7	13:00	2	3	21:36	31	45	21:36	29	41
13:06	8	10	13:06	17	22	21:48	21	29	21:48	1	2
13:13	20	27	13:13	27	37	22:00	13	17	22:00	30	43
13:20	6	8	13:20	32	46	22:12	27	38	22:12	12	16
13:26	30	42	13:26	28	39	22:24	23	33	22:24	19	26
13:33	15	19	13:33	23	32	22:36	18	25	22:36	36	54
13:40	22	30	13:40	29	40	22:48	9	12	22:48	35	52

Table A3. Schedule for route #3 of 4 routes.

Origin to Destination			Destination to Origin			Origin to Destination			Destination to Origin		
Time	Bus No.	Driver No	Time	Bus No.	Driver No.	Time	Bus No.	Driver No.	Time	Bus No.	Driver No.
5:00	16	19	5:00	19	24	13:46	19	24	13:46	11	13
5:12	18	22	5:12	14	17	13:53	14	17	13:53	4	5
5:24	10	12	5:24	15	18	14:00	6	8	14:00	18	22
5:36	19	24	5:36	16	19	14:12	15	18	14:12	1	1
5:48	14	17	5:48	12	15	14:24	11	13	14:24	2	3
6:00	15	18	6:00	10	12	14:36	4	5	14:36	12	15
6:12	16	19	6:12	19	24	14:48	1	1	14:48	3	4
6:24	12	15	6:24	14	17	15:00	2	3	15:00	11	13
6:36	4	5	6:36	15	18	15:12	13	16	15:12	21	28
6:48	19	24	6:48	3	4	15:24	3	4	15:24	1	1
7:00	14	17	7:00	12	15	15:36	11	13	15:36	2	3
7:06	20	26	7:06	18	22	15:48	21	28	15:48	13	16
7:13	15	18	7:13	4	5	16:00	1	1	16:00	7	9
7:20	3	4	7:20	19	24	16:06	5	7	16:06	22	29
7:26	17	20	7:26	16	19	16:13	2	3	16:13	17	20
7:33	12	15	7:33	14	17	16:20	9	11	16:20	6	8
7:40	18	22	7:40	20	26	16:26	13	16	16:26	21	28
7:46	4	5	7:46	15	18	16:33	7	9	16:33	1	1
7:53	19	24	7:53	3	4	16:40	22	29	16:40	8	10
8:00	16	19	8:00	17	20	16:46	17	21	16:46	19	25
8:06	14	17	8:06	12	15	16:53	6	8	16:53	9	11
8:13	20	26	8:13	18	22	17:00	21	28	17:00	13	16
8:20	15	18	8:20	4	5	17:06	1	1	17:06	7	9
8:26	3	4	8:26	19	24	17:13	8	10	17:13	11	13
8:33	17	20	8:33	16	19	17:20	19	25	17:20	17	21
8:40	12	15	8:40	14	17	17:26	9	11	17:26	6	8
8:46	18	22	8:46	20	26	17:33	13	16	17:33	21	28
8:53	4	5	8:53	15	18	17:40	7	9	17:40	1	1
9:00	19	24	9:00	3	4	17:46	11	13	17:46	8	10
9:06	16	19	9:06	17	20	17:53	17	21	17:53	19	25
9:13	10	12	9:13	12	15	18:00	6	8	18:00	9	11
9:20	20	26	9:20	18	22	18:06	21	28	18:06	13	16
9:26	2	3	9:26	4	5	18:13	1	1	18:13	7	9
9:33	3	4	9:33	5	7	18:20	8	10	18:20	11	13
9:40	17	20	9:40	16	19	18:26	19	25	18:26	17	21

Continued

9:46	12	15	9:46	10	12	18:33	9	11	18:33	6	8
9:53	18	22	9:53	20	26	18:40	13	16	18:40	21	28
10:00	4	5	10:00	2	3	18:46	7	9	18:46	22	29
10:12	5	7	10:12	3	4	18:53	11	13	18:53	8	10
10:24	14	17	10:24	17	20	19:00	17	21	19:00	19	25
10:36	15	18	10:36	4	5	19:06	6	8	19:06	9	11
10:48	3	4	10:48	5	7	19:13	4	6	19:13	20	27
11:00	17	20	11:00	14	17	19:20	22	29	19:20	7	9
11:12	1	1	11:12	15	18	19:26	8	10	19:26	11	13
11:24	10	12	11:24	12	15	19:33	18	23	19:33	17	21
11:36	14	17	11:36	18	22	19:40	9	11	19:40	6	8
11:48	15	18	11:48	1	1	19:46	20	27	19:46	1	1
12:00	12	15	12:00	10	12	19:53	19	25	19:53	22	29
12:06	11	13	12:06	19	24	20:00	11	13	20:00	8	10
12:13	4	5	12:13	14	17	20:12	21	28	20:12	13	16
12:20	18	22	12:20	6	8	20:24	1	2	20:24	9	11
12:26	1	1	12:26	15	18	20:36	8	10	20:36	11	13
12:33	2	3	12:33	17	20	20:48	13	16	20:48	21	28
12:40	19	24	12:40	11	13	21:00	9	11	21:00	1	2
12:46	14	17	12:46	4	5	21:12	11	14	21:12	19	25
12:53	6	8	12:53	18	22	21:24	21	28	21:24	13	16
13:00	15	18	13:00	1	1	21:36	1	2	21:36	8	10
13:06	17	20	13:06	2	3	21:48	22	29	21:48	11	14
13:13	11	13	13:13	19	24	22:00	13	16	22:00	21	28
13:20	4	5	13:20	14	17	22:12	8	10	22:12	1	2
13:26	18	22	13:26	6	8	22:24	11	14	22:24	22	29
13:33	1	1	13:33	15	18	22:36	21	28	22:36	13	16
13:40	2	3	13:40	17	20	22:48	7	9	22:48	9	11

Table A4. Schedule for route #4 of 4 routes.

Origin to Destination			Destination to Origin			Origin to Destination			Destination to Origin		
Time	**Bus No.**	**Driver No**	**Time**	**Bus No.**	**Driver No**	**Time**	**Bus No.**	**Driver No**	**Time**	**Bus No.**	**Driver No**
5:00	18	28	5:00	16	25	13:46	2	2	13:46	8	13
5:12	2	2	5:12	17	26	13:53	13	19	13:53	4	5
5:24	1	1	5:24	5	7	14:00	1	1	14:00	9	14
5:36	16	25	5:36	18	28	14:12	5	7	14:12	6	9
5:48	19	29	5:48	12	17	14:24	12	17	14:24	15	23

Continued

6:00	17	26	6:00	14	21	14:36	4	5	14:36	13	19
6:12	18	28	6:12	16	25	14:48	6	9	14:48	5	8
6:24	12	17	6:24	22	32	15:00	15	23	15:00	12	18
6:36	14	21	6:36	17	26	15:12	13	19	15:12	4	5
6:48	6	9	6:48	18	28	15:24	5	8	15:24	6	9
7:00	22	32	7:00	12	17	15:36	12	18	15:36	15	23
7:06	13	19	7:06	19	29	15:48	4	5	15:48	13	19
7:13	17	26	7:13	14	21	16:00	6	10	16:00	5	8
7:20	18	28	7:20	6	9	16:06	10	15	16:06	20	30
7:26	3	4	7:26	15	23	16:13	15	23	16:13	12	18
7:33	12	17	7:33	22	32	16:20	11	16	16:20	17	27
7:40	4	5	7:40	13	19	16:26	13	20	16:26	4	5
7:46	14	21	7:46	17	26	16:33	5	8	16:33	6	10
7:53	6	9	7:53	18	28	16:40	20	30	16:40	10	15
8:00	15	23	8:00	3	4	16:46	12	18	16:46	15	24
8:06	22	32	8:06	12	17	16:53	17	27	16:53	23	34
8:13	13	19	8:13	4	5	17:00	4	6	17:00	13	20
8:20	17	26	8:20	14	21	17:06	6	10	17:06	5	8
8:26	18	28	8:26	6	9	17:13	8	13	17:13	20	30
8:33	3	4	8:33	15	23	17:20	15	24	17:20	12	18
8:40	12	17	8:40	22	32	17:26	23	34	17:26	17	27
8:46	4	5	8:46	13	19	17:33	13	20	17:33	4	6
8:53	14	21	8:53	17	26	17:40	5	8	17:40	6	10
9:00	6	9	9:00	18	28	17:46	20	30	17:46	8	13
9:06	15	23	9:06	3	4	17:53	12	18	17:53	15	24
9:13	22	32	9:13	12	17	18:00	9	14	18:00	23	34
9:20	13	19	9:20	4	5	18:06	4	6	18:06	13	20
9:26	17	26	9:26	14	21	18:13	6	10	18:13	11	16
9:33	7	11	9:33	6	9	18:20	8	13	18:20	20	30
9:40	3	4	9:40	15	23	18:26	15	24	18:26	12	18
9:46	12	17	9:46	22	32	18:33	23	34	18:33	9	14
9:53	4	5	9:53	13	19	18:40	13	20	18:40	4	6
10:00	14	21	10:00	17	26	18:46	7	12	18:46	6	10
10:12	6	9	10:12	7	11	18:53	20	30	18:53	8	13
10:24	15	23	10:24	3	4	19:00	11	16	19:00	15	24
10:36	18	28	10:36	4	5	19:06	9	14	19:06	23	34
10:48	7	11	10:48	6	9	19:13	4	6	19:13	5	8
11:00	3	4	11:00	15	23	19:20	6	10	19:20	7	12

Continued

11:12	4	5	11:12	18	28	19:26	8	13	19:26	20	30
11:24	20	30	11:24	12	17	19:33	15	24	19:33	11	16
11:36	17	26	11:36	14	21	19:40	23	34	19:40	9	14
11:48	18	28	11:48	16	25	19:46	5	8	19:46	4	6
12:00	12	17	12:00	22	32	19:53	17	27	19:53	21	31
12:06	8	13	12:06	2	2	20:00	12	18	20:00	2	3
12:13	19	29	12:13	13	19	20:12	11	16	20:12	15	24
12:20	9	14	12:20	1	1	20:24	4	6	20:24	5	8
12:26	6	9	12:26	18	28	20:36	22	33	20:36	12	18
12:33	15	23	12:33	12	17	20:48	21	31	20:48	9	14
12:40	2	2	12:40	8	13	21:00	5	8	21:00	6	10
12:46	13	19	12:46	4	5	21:12	12	18	21:12	22	33
12:53	1	1	12:53	9	14	21:24	14	22	21:24	13	20
13:00	18	28	13:00	6	9	21:36	6	10	21:36	5	8
13:06	12	17	13:06	15	23	21:48	15	24	21:48	12	18
13:13	8	13	13:13	2	2	22:00	13	20	22:00	4	6
13:20	4	5	13:20	13	19	22:12	5	8	22:12	6	10
13:26	9	14	13:26	1	1	22:24	12	18	22:24	15	24
13:33	6	9	13:33	18	28	22:36	4	6	22:36	13	20
13:40	15	23	13:40	12	17	22:48	10	15	22:48	17	27