Final Report

Multi-depot and Multi-school Bus scheduling Problem with School Bell Time Optimization

Qinglian He, Ph.D.
University of Maryland, College Park

Ali Haghani, Ph.D.
University of Maryland, College Park

Prepared for the Urban Mobility & Equity Center, Morgan State University, CBEIS 327, 1700 E. Coldspring Lane, Baltimore, MD 21251

ACKNOWLEDGMENT

This research was partially funded by the Urban Mobility & Equity Center, Morgan State University.

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Multi-depot and Multi-school Bus scheduling Problem with School Bell Time Optimization		**5. Report Date** 12 April 2022
		6. Performing Organization Code
7. Author(s) *Include ORCID #* 0000-0003-3181-7155		**8. Performing Organization Report No.**
9. Performing Organization Name and Address University of Maryland College Park, MD 20742		**10. Work Unit No.**
		11. Contract or Grant No. 69A43551747123
12. Sponsoring Agency Name and Address US Department of Transportation Office of the Secretary-Research UTC Program, RDT-30 1200 New Jersey Ave., SE Washington, DC 20590		**13. Type of Report and Period Covered** Final
		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

The school bus transportation system is responsible for transporting students to and from schools safely and promptly. This research aims to optimize the school bus schedules and the school bell times simultaneously for improving the efficiency of the school bus system operation. We consider the school bus scheduling problem in a multi-depot multi-school bus system and incorporate the bell time optimization to make bus operations more efficient. We propose four different methods, including one exact method and three heuristic methods, to solve the Multi-depot and Multi-school Bus Scheduling Problem with School Bell Time Optimization (MDSBSPTW). Besides, they can also solve the Single-depot Multi-school Bus Scheduling Problem with School Bell Time Optimization problems (SDSBSPTW) or problems without bell time optimization regardless of the number of depots (i.e., MDSBSP or SDSBSP).

First, the MDSBSPTW is formulated as a mixed-integer programming model. Second, a two-phase heuristic method is proposed, namely the first-route second-assignment method. Then, we improve the first-route phase of the two-phase heuristic method with a Simulated Annealing-based Greedy Algorithm (SA-GDA) method. Combined with the same second-assignment phase, we have the improved two-phase heuristic method. Finally, we propose a Tabu Search-Based Ant Colony Optimization (TS-ACO) for solving the MDSBSPTW without dividing it into different phases.

Fourteen test problems with different characteristics derived from the real-world collected data are used to examine the performance of the proposed four methods. The results are compared and explained. The improved two-phase heuristic method and the TS-ACO method perform better than the other two methods when the problems are more complicated (e.g., more trips, depots, schools, or larger bell time window). Overall, the improved two-phase heuristic method is the best as it can achieve better solutions (i.e., fewer buses) much quicker, especially for large-size complicated problems. Generally, incorporating the school bell time optimization into the school bus scheduling problem can significantly reduce the total number of buses since it makes more trips compatible and thus link more trips together on a single bus route. Besides, the larger the school bell time window, the fewer buses are needed overall, but more computational resources and longer model running time are required.

17. Key Words :		**18. Distribution Statement** No Restriction	
19. Security Classif. (of this report) : Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 150	**22. Price**

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Problems

School Bus Scheduling Problem: SBSP

Single-depot Multi-school Bus Scheduling Problem: SDSBSP

Multi-depot Multi-school Bus Scheduling Problem: MDSBSP

School Bus Scheduling Problem with School Bell Time Optimization: SBSPTW

Single-depot Multi-school Bus Scheduling Problem with School Bell Time Optimization: SDSBSPTW

Multi-depot Multi-school Bus Scheduling Problem with School Bell Time Optimization: MDSBSPTW

Algorithms:

Simulated Annealing: SA

Genetic Algorithm: GA

Greedy Algorithm: GDA

Ant Colony Optimization: ACO

Tabu Search: TS

Simulated Annealing-based Greedy Algorithm: SA-GDA

Tabu Search-based Ant Colony Optimization: TS-ACO

Others:

ES: Elementary School

MS: Middle School

HS: High School

RT: Model Running Time

ODT: Original Dismissal Time

TOB: Total Number of Buses

TDT: Total Deadhead Time Between Trips (min)

TDTD: Total Deadhead Time Between Trips and Depots (min)

TD: Total Deadhead Time (min), including the deadhead between trips and the deadhead between trips and depots

Chapter 1: Introduction

1.1 Background

In the U.S., the expenditures for public student transportation keep increasing. According to the National Center for Education Statistics (2021), from 2010-11 to 2017-18, the expenditures have increased from about $22 billion to $26 billion. The average spending per student transported in 2017-18 has risen to $1079. However, since the economic recession in 2008, most states have cut school funding. In 2015, there were still 29 states that didn't restore their school funding to pre-recession levels (Leachman et al., 2017). In this case, public school districts are facing severe budget crises. Consequently, transportation spending, which mainly consists of bus-related costs and bus drivers' salaries, is often reduced.

Improving school transportation efficiency with only a limited transportation budget is crucial. Considering the very high cost of buying new buses (from $50,000 to $100,000 per bus) and the annual maintenance cost for each bus, minimizing the total number of buses can be the number one choice to lessen the financial strain. Another way to improve the efficiency is to cut down on deadhead time, namely the total time buses run without students on board. By doing so, school districts can achieve significant savings in fuel and wear on the buses. To achieve those goals, mathematical models combined with optimization technologies should be utilized to construct efficient school bus routes to serve all the students cost-effectively.

Moreover, adjusting school bell times can also improve bus operation efficiency. In a particular school district, the demand for school buses is usually unbalanced since the bell times of most schools are around the same time. If we can change the bell time of some schools, more trips will become compatible so that more trips can be linked together and be served on the same bus. As a result, one single bus route becomes longer, thus reducing the total number of buses.

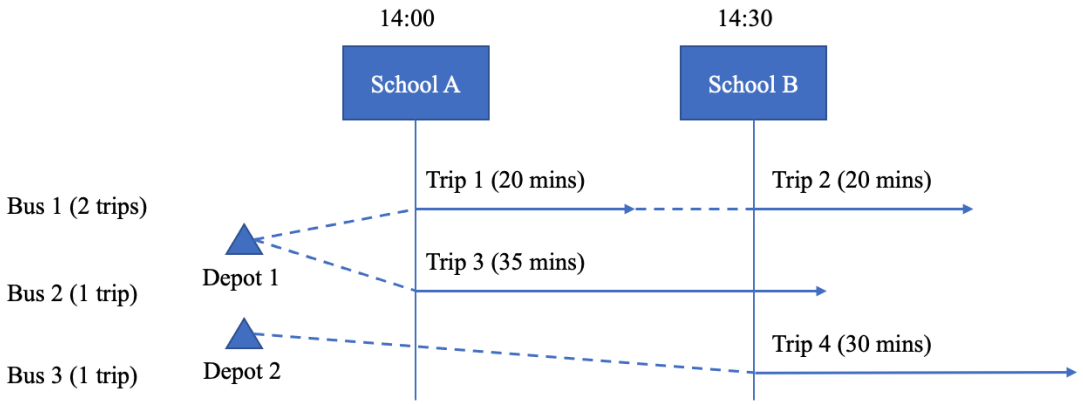


Figure 1. Illustration of bus routes before bell time optimization

A simple example illustrates this point. Two schools and two depots (i.e., Depot 1 and Depot 2) are in the hypothetical school bus system shown in Figure 1. Please note that we only show the starting depot for each bus, but each bus must start and end at the same depot. For example, Bus 1 starts from Depot 1, and it should return to Depot 1 once finishing its journey. We consider the PM case where school buses transport students from schools to their homes. The dismissal time for School A and School B are 14:00 and 14:30, respectively. School A has two trips, Trip 1 with a travel duration of 20 minutes and Trip 3 with a travel duration of 35 minutes. Trip 2 and Trip 4 belong to School B, and their travel times are 20 minutes and 30 minutes, respectively.

Three buses are needed to serve those four trips. Bus 1 is a two-trip bus. It starts from Depot 1 and first arrives at School A to serve Trip 1. Then, it takes 10 minutes to drive from the last stop of Trip 1 to School B (the first stop of Trip 2). At 14:30, Bus 1 starts to serve Trip 2 and will go back to Depot 1 after finishing Trip 2. Bus 2 is a single-trip bus that starts from Depot 2 and arrives at School A to serve Trip 3. After finishing Trip 3, Bus 2 will go back to Depot 2. Similarly, Bus 3 starts from Depot 1, arrives at School B to serve Trip 4 and will return to Depot 1 at the end.

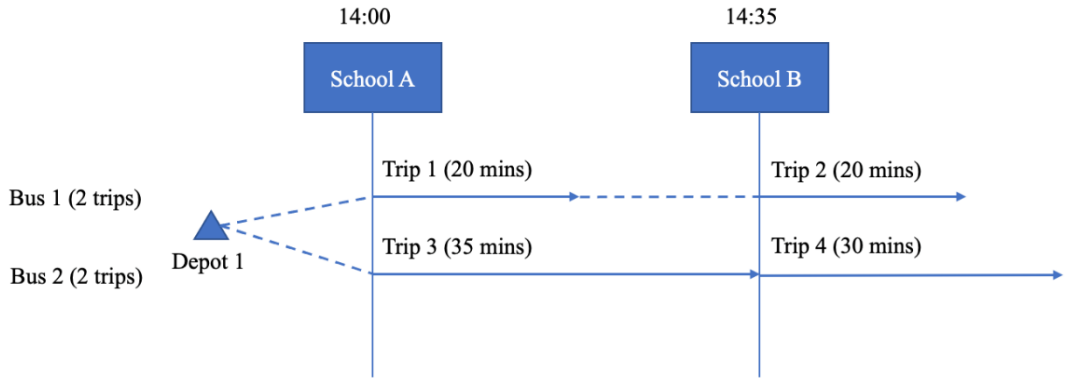


Figure 2. Illustration of bus routes after bell time optimization

If we change the dismissal time of School B from 14:30 to 14:35, as shown in Figure 2, then the total number of buses in this system can be reduced to two. Because Trip 3 and Trip 4 can be served on the same bus now (i.e., Bus 2), and thus Bus 3 is eliminated. Therefore, bell time optimization can reduce the number of buses needed overall.

Given the trip information and school bell time plans, the School Bus Scheduling Problem (SBSP) aims to optimize the school bus schedules for a given school district. This research focuses on the SBSP with the objective to minimize the number of buses

needed overall and the total deadhead time. Besides, to potentially reduce the total number of buses, the school bell time optimization is incorporated into the SBSP. Therefore, the integrated model is used to optimize the school bell times and provide an efficient bus schedule to improve the efficiency of the school bus system.

1.2 Motivation

Most existing studies consider the school bus scheduling problem as a single-depot problem in which buses are required to start and end at the only depot (or garage). However, this cannot represent the real-world situations where the school buses usually start from multiple garages, serve several routes in the morning or afternoon, and return to the garages after finishing their work. Therefore, it is essential to formulate the school bus scheduling problem as a multi-depot problem.

Besides, bell time optimization is also challenging for such a multi-depot multi-school system. On the one hand, the number of related studies is small, and those mainly work on single-depot bus systems. On the other hand, unlike the Multi-Depot Vehicle Routing Problem with Time Window (MDVRPTW), where the time window for each customer is independent of other customers, the time windows for schools are coupled with each other because multiple trips may be associated with each school. If the school bell times are changed, the trip starting times should be adjusted accordingly. And when the schools have multiple trips, to ensure that all the students can come to the school (or return home) on time, the starting times of the trips belonging to the same school

should be around the same time. Therefore, synchronizing the change in the starting time of the schools and trips is a major challenge.

Though doing the school bus scheduling and bell time optimization are beneficial, we don't want to solve them as two subproblems sequentially as they are highly related to each other. Therefore, we should incorporate the bell time optimization into the multi-depot multi-school bus scheduling problem to simultaneously provide a good bus schedule and school bell times. However, the integrated model, which deals with more features, is much more complicated and can be very hard to solve, especially for large-size problems. Therefore, solving the integrated model efficiently, especially for large MDSBSPTWs, is a critical task in this study.

1.3 Research Contributions

This study aims to solve the multi-depot multi-school bus scheduling problem with bell time optimization (MDSBSPTW) for improving the efficiency of the given school bus system. The solution to the problem is the best bus schedule that minimizes the total number of buses, the total deadhead duration, and the corresponding school bell time for each school. This study tries to fulfill several existing research gaps:

1. Formulate a mathematical model for the MDSBSPTW.

This study will develop an exact method, a Mixed-Integer Programming (MIP) model, to solve the MDSBSPTW. The proposed MIP model will also be able to solve the single-depot multi-school bus scheduling problems with bell time optimization

(SDSBSPTW) or the problems without bell time optimization regardless of the total number of depots in the system (i.e., MDSBSP and SDBSP).

2. Develop heuristic methods for large-size MDSBSPTW.

The proposed MIP model may lose its power when dealing with large-scale instances, and heuristic methods will become the methodology of choice. We will develop several heuristic methods from different perspectives to find suitable solutions within reasonable time for large-scale problems. All the proposed heuristic methods will be capable of solving SDSBSPTWs, SDSBSPs, and MDSBSPs as well.

3. Test the performance of all the proposed methods.

Real-world data will be collected for testing the performance of all the proposed methods. We will generate multiple test problems with different characteristics (e.g., different depot size, trip size, school size, and different sizes for the school bell time window) from the collected data. The performance of all proposed methods on each test problem will be presented, compared, and analyzed. We will also test the performance of all the proposed methods on the largest problem that uses the entire collected data. Their performances will be examined and compared. Finally, sensitivity analysis will be conducted on some key parameters of the MDSBSPTW.

1.4 Report Structure

The organization of the report is as follows:

- Chapter 1 introduces the background and the motivation of this research. It also presents the problem statement and the contributions.

- Chapter 2 is the literature review. It first summarizes the current studies on the vehicle routing problem, multi-depot vehicle routing problem, and multi-depot vehicle routing problem with time windows. Then, it reviews the existing studies on the school bus scheduling problem and the school bell time optimization. Finally, the research gaps are discussed.
- Chapter 3 presents the mathematical formulation for the multi-depot multi-school bus scheduling problem with bell time optimization. The model assumptions are first introduced. Then, the mixed-integer programming model is presented. The objective function and the constraints are clearly stated.
- Chapter 4 presents a two-phase heuristic method, the first-route second-assignment method, for solving the multi-depot multi-school bus scheduling problem with bell time optimization through two phases sequentially. The first-route phase converts the original problem into a single-depot multi-school bus scheduling problem with bell time optimization and is formulated as a mixed-integer programming model. The second-assignment phase is a bus-depot assignment problem formulated as an integer programming model. Then, we propose the improved two-phase heuristic method. It keeps the second-assignment phase unchanged while replacing the MIP model in the first-route phase with the Simulated Annealing-based Greedy Algorithm (SA-GDA) method. All the models and the SA-GDA method are presented in detail.
- Chapter 5 presents the Tabu Search-based Ant Colony Optimization (TS-ACO) method for solving the MDSBSPTW without dividing it into different phases. The proposed ACO algorithm and the TS method are described in detail.

- Chapter 6 presents the performances of all proposed methods on fourteen test problems derived from real-world collected data. They all are applied to the largest test problem based on all collected data at the end. The performances of different methods on all test problems are examined and compared, followed by the sensitivity analysis results on two key model parameters.
- Chapter 7 concludes this research and discusses the future research directions.

Chapter 2: Literature Review

The multi-depot multi-school bus scheduling problem with the school bell time optimization problem (MDSBSPTW) can be considered a real-world application of the Multi-depot Vehicle Routing Problem with Time Windows (MDVRPTW), which is a variant of the classic Vehicle Routing Problem. Therefore, first, an overview of the classic Vehicle Routing Problem and the Multi-depot Vehicle Routing Problem is provided. Then, the Multi-depot Vehicle Routing Problem with Time Windows research is discussed. Finally, the School Bus Scheduling Problem and the School Bell Time Optimization studies are reviewed.

2.1 Vehicle Routing Problem

Given a set of customers, Vehicle Routing Problems (VRPs) aim to find a set of optimal routes that serve all the customers at minimum cost. It is one of the most important and studied optimization problems and has wide applications in real-world systems (Toth and Vigo, 2002). The basic configurations of the VRPs include a central depot, deterministic demand, and a homogeneous fleet of vehicles. However, real-world situations are often much more complicated. Therefore, different VRP variants are proposed to accommodate different situations. The VRP variants are the basic VRP with extra features such as heterogeneous vehicle capacity, customer time windows, and multiple depots. Some of them are shown in Figure 3.

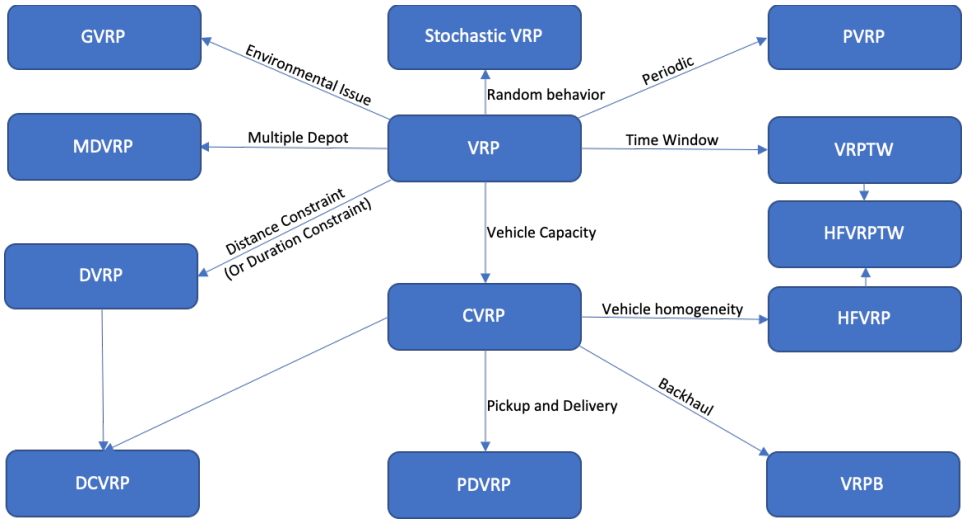


Figure 3. Variants of VRP

VRP variants can better describe the actual situation. For example, Dantzig and Ramser (1959) first introduced the CVRP for the gasoline delivery truck dispatching problem. They proposed a procedure based on a linear programming formulation to obtain a near-optimal solution. Later, Clarke and Wright (1964) came up with a heuristic method that is based on the "saving" concept to improve the solution. Since then, a large number of methods and algorithms have been proposed to solve the VRP and its variants. There are mainly three types of approaches, namely the exact methods, heuristic methods, and metaheuristic methods.

Some popular exact methods include branch-and-bound, branch-and-cut, and branch-and-cut-and-price. However, both the VRP and its variants are NP-hard such that the exact methods are only powerful when the problem size is small. In general, these small instances involve around 100 customers (Laporte et al., 2014). Otherwise, solving the problem with the exact methods will become super time-consuming. Take CVRP, for example. To the best of the author's knowledge, the largest instance solved to

optimality with the exact method has 360 customers. The problem was solved with an improved branch-and-cut-and-price algorithm by Pecin et al. (2017). However, it took a very long time to reach optimality (162,405 seconds).

The heuristic or metaheuristic methods are proposed and widely used for large-scale instances in practice, considering the tradeoff between computational speed and solution quality. They are capable of quickly finding near-optimal solutions. Classical heuristic methods include construction heuristics and improvement heuristics (Laporte et al., 2014). As for metaheuristics, Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA), and Ant Colony Optimization (ACO) are most commonly used. Those heuristics and metaheuristics can solve large-scale VRPs with nearly 500 customers (Kytöjoki et al., 2007; Laporte et al., 2014).

More recently, the interest in hybrid methods has grown rapidly due to their ability to combine the advantages of different algorithms or techniques together (Subramanian et al., 2013). Meta-meta hybridization is the commonly used one. For example, Kytöjoki et al. (2007) presented a Variable Neighborhood Search (VNS) method combined with a Guided Local Search (GLS) metaheuristic to prevent the local minima. They can efficiently solve very large-scale real-life VRPs in a reasonable time. The proposed hybrid method can solve the CVRP with up to 20,000 customers in about 144 minutes. Another hybridization is the metaheuristic, which combines the exact method (i.e., the mathematical programming technique) and the heuristics (or the metaheuristics). For example, Subramanian (2013) proposed a hybrid method that

combines a set partitioning model with an Iterated Local Search (ILS) heuristic for solving a class of VRPs. The idea is to find and store sufficiently high-quality routes (i.e., the columns of the set-partitioning model) based on the ILS and then solve the set partitioning model with a MIP solver. Results showed that the solutions of some benchmark examples could be improved based on the proposed metaheuristic. Other best performing published methods include Knowledge-Guided Local Search (KGLS) (Arnold and Sörensen, 2019), Slack Induction by String Removals (SISRs) (Christiaens and Vanden Berghe, 2020), and the Partial Optimization Metaheuristic Under Special Intensification Conditions (POPMUSIC) (Queiroga et al., 2021).

2.2 Multi-depot Vehicle Routing Problem

The Multi-depot VRP (MDVRP) is an extension of the basic VRP. In the MDVRP, vehicles start from multiple depots to serve all the customers and are often required to return to the same depot from which they start (Renaud et al., 1996). It has important applications in the field of transportation, logistics, and distribution. For example, logistics companies often operate from more than one distribution center (commonly referred to as a depot) to efficiently transport goods to customers.

Let $G(V, A)$ be a complete graph. V is the vertex set which consists of a depot set $S = \{s_1, s_2, \cdots, s_n\}$ and a customer set $W = \{w_{n+1}, w_{n+2}, \cdots, w_{n+m}\}$. In this case, there is a total number of n depots and m customers. $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ is the arc set and is associated with a travel time matrix ($c_{v_i v_j}, v_i, v_j \in V, i \neq j$). Based on this graph, The MDVRP aims to optimize the vehicle routes such that (1) each customer

must be visited exactly once by any vehicle in the fleet, and (2) each vehicle starts from a depot s_n and is required to go back to s_n after finishing its route. The objective can be a single objective, such as minimizing the total traveling cost (i.e., traveling time or distance) or minimizing the total number of vehicles. Or it can be a bi-objective function that minimizes the total number of buses and the total traveling cost. Figure 4 is an example of a feasible solution for an MDVRP. There are 15 customers and three depots. Five routes (i.e., the solid blue lines) are built to serve all those 15 customers exactly once. And every vehicle starts and ends at the same depot.

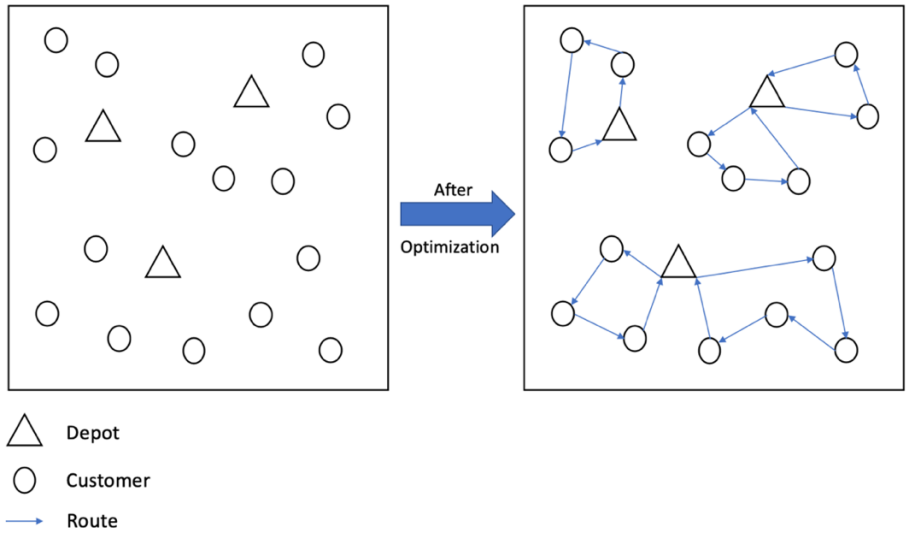


Figure 4. Illustration of the MDVRP

As one variant of the VRP, the MDVRP is NP-hard. Thus, most studies worked on developing heuristic and metaheuristic methods to find good solutions for relatively large-scale instances in a short time. Despite this, some research studies focused on exact methods. Pioneer works would be two branch-and-bound algorithms for solving the symmetric version MDVRP (1894) and asymmetric version MDVRP (1988), respectively (Laporte et al., 1984, 1988). More recently, Baldacci and Mingozzi (2009)

presented a unified exact method for solving several different classes of the VRP, including MDVRP. The proposed method is based on the set partitioning formulation with new proposed tight lower bounds. It can solve the instance with 199 customers and four depots to optimality. Contardo and Martinelli (2014) also used the set-partitioning formulation. Combined with a vehicle-flow formulation, the proposed exact method can solve the MDVRP with capacity and route length constraints. Results showed that it can provide stronger lower bounds and could solve some previously unsolved open instances with the exact method. The largest instance it could solve optimally had 240 customers and six depots. More recently, Ramos et al. (2020) proposed a two-commodity flow formulation for the MDVRP considering a heterogeneous vehicle fleet and maximum routing time. The proposed model was tested on some benchmark instances and achieved better performances.

As for the heuristic methods, the cluster first-route second heuristic methods are widely used. Specifically, “Cluster first” refers to decomposing the MDVRP into multiple single-depot VRPs based on different customer allocation strategies, and then “route-second” solves a VRP for each depot. The most straightforward customer allocation strategy would be the nearest-depot assignment approach, firstly proposed by Tillman (1969). It assigns the customers to their nearest depot and then adds refinements to improve the solution (Salhi and Sari, 1997). Later, researchers took the customer geographical distribution features into consideration and improved customer allocation with some clustering methods or some assignment algorithms. The clustering methods include the adaptive genetic clustering method (Salhi and Sari, 1997), the geometric

shape-based genetic clustering algorithm (Yücenur and Demirel, 2011), and the bi-level Voronoi diagram (Tu et al., 2014). As for the assignment algorithms, Giosa et al. (2002) proposed several measures for assigning a customer to a depot, such as an assignment through urgencies, cyclic assignment, and assignment by clusters.

Metaheuristics can be the method of choice for efficiently solving large-scale instances in a reasonable time and providing good solutions (Montoya-Torres et al., 2015). Two often-used metaheuristics for the MDVRPs are Tabu Search (TS) (Renaud et al., 1996; Escobar et al., 2014) and Genetic Algorithm (GA) (Ho et al., 2008; Vidal et al., 2012). The TS algorithm is a local search method designed to avoid a local optimum based on a flexible memory system. The GAs consist of several steps, such as mutation, crossover, and selection. They are robust and effective but may have a slow convergence speed. A nice review of the genetic algorithms for MDVRPs can be found in the paper written by Karakatič and Podgorelec (2015).

Some bio-inspired metaheuristic methods are also very popular for solving the MDVRPs. Those methods include the Ant Colony Optimization algorithm (ACO) and the Antlion Optimization (ALO). The ACO is inspired by the ants' foraging behavior, and the pheromone is the key parameter that affects the ants' route selection (Yu et al., 2011; Demirel and Yilmaz., 2012). While the ALO is based on the ant hunting mechanism of antlions which mainly consists of the random walk of the ants and the entrapment and catching behavior of the antlions (Barma et al., 2019). Besides, Bezerra et al. (2018) solved the MDVRP through a General Neighborhood Search (GVNS).

The GVNS has fewer parameters but can also achieve reasonable results. Sadati et al. (2021) also work on solving the MDVRP using the VNS. They combined it with the tabu shaking mechanism to improve both solution quality and mode running time.

Researchers also extend the basic MDVRP with additional constraints. Some frequently used constraints are (1) depot (or vehicle) capacity; (2) fleet type (homogeneous or heterogeneous); (3) time windows for customers; (4) route length (i.e., traveling time or distance); (5) vehicles are not required to go back to the depots from which they start (i.e., open VRP); (6) pickup and delivery demand at each customer. A summary of some MDVRP variants is presented in Table 1.

Table 1. Summary of some MDVRP variants

Authors and Year	Type of Problem	Main Constraints	Method Type	Method (Algorithm)	Number of Depots	Number of Customers
Polacek et al. (2004)	MDVRPTW	Time windows for customers	Heuristic	Variable Neighborhood Search	4≤S≤6	48≤N≤288
Nagy and Salhi (2005)	MDVRPPD	Vehicle capacity, minimum and maximum loads for a route	Heuristic	Integrated Heuristic	2≤S≤5	50≤N≤249
Crevier et al. (2007)	MDVRPI	Inter-depot routes	Heuristic	Tabu Search	5≤S≤7	48≤N≤288
Bettinelli et al. (2011)	MDHVRPTW	The time window for customers; Heterogeneous feet; route duration	Exact	Branch-and-Cut-and-Price	2≤S≤6	25≤N≤100
Vidal et al. (2012)	MDPVRP	Multiple periods (4 or 6 days), vehicle capacity, route length	Heuristic	Hybrid Genetic	4≤S≤6	48≤N≤288
Narasimha et al. (2013)	min–max MDVRP	Maximum route length	Heuristic	Ant Colony Optimization	3≤S≤5	80≤N≤140

Salhi et al. (2014)	MDHFVRP	Multiple vehicle types (5 types), vehicle capacity, route length	Heuristic	Variable Neighborhood Search	2≤S≤9	50≤N≤360
Wang et al. (2016)	min-max SDMDVRP-MSTR	A customer can be visited multiple times by different vehicles, and route duration	Heuristic	MD Heuristic	1≤S≤20	10≤N≤500
Alinaghian et al. (2018)	MDMCVRP	The cargo space of each vehicle has multiple compartments with a limited capacity, vehicle route duration, depot capacity	Heuristic	Hybrid Adaptive Large Neighborhood Search	1≤S≤6	30≤N≤360
Zhou et al. (2018)	MD-TEVRP-DO	Two levels of routing problems: depots-satellites and satellites-customers. Customers have two delivery options (pick up/direct delivery), vehicle capacity, satellite capacity	Heuristic	Hybrid Multi-population Genetic Algorithm	1≤S≤3 (4~12 satellites, 10~30 pickup facilities)	50≤N≤200
Zhang et al. (2019)	MDGVRP	Refill the alternative fuel-powered vehicles only at their original depots	Heuristic	Ant Colony System Algorithm	4≤S≤6	25≤N≤75

Brandão, J. (2020)	MDOVRP	Vehicles are not allowed to return to the depot, and vehicle capacity	Heuristic	Memory-based Iterated Local Search	4≤S≤6	48≤N≤288
Sadati et al. (2021)	MDVRPTW; MDOVRP	Vehicle capacity, time window, tour duration	Heuristic	Variable Tabu Neighborhood Search	4≤S≤6	48≤N≤288

Note:

- MDVRPTW: MDVRP with time windows
- MDVRPPD: MDVRP with pickups and deliveries
- MDVRPI: MDVRP with inter-depot routes
- MDHVRPTW: Multi-depot heterogeneous vehicle routing problem with time window
- MDPVRP: Multi-depot periodic VRP
- min–max MDVRP: minimize the maximum distance traveled by any vehicle
- MDHFVRP: Multi-depot heterogeneous vehicle routing problem
- min-max SDMDVRP-MSTR: min-max split delivery multi-depot vehicle routing problem with minimum service time requirement
- MDMCVRP: Multi-depot multi-compartment vehicle routing problem
- MD-TEVRP-DO: Multi-depot two-echelon vehicle routing problem with delivery options for the last mile distribution
- MDGVRP: Multi-depot green vehicle routing problem
- MDOVRP: Multi-depot open vehicle routing problem

2.3 Multi-depot Vehicle Routing Problem with Time Windows

The Multi-depot Vehicle Routing Problem with Time Windows (MDVRPTW) is one MDVRP variant. Besides the two basic constraints of the MDVRP, the additional constraint defines a time window $[e_m, l_m]$ for each customer m where e_m and l_m are the earliest service time and latest service time, respectively. In this case, the customer m can be served at any time in the given time window. Some real-life applications of MDVRPTW are package delivery, school bus routing, and waste collection.

The MDVRPTW is NP-hard, and thus metaheuristics are widely used. Luo and Chen (2014) proposed the Multi-Phase Modified Shuffled Frog Leaping Algorithm (MPMSFLA) framework for solving both MDVRP and MDVRPTW. Sadati et al. (2021) proposed an efficient Variable Tabu Neighborhood Search (VTNS) for solving a class of MDVRPs, including MDVRP itself, MDVRPTW, and MDOVRP.

From another perspective, the MDVRPTW is also one variant of the VRPTW. Therefore, some researchers designated a unified metaheuristic method for solving a large class of VRPTWs and then applied it to MDVRPTW. For example, Cordeau et al. (2001) proposed a unified Tabu Search (TS) algorithm and applied it to the VRPTW, PVRPTW, and MDVRPTW. Later in 2004, they improved the TS for solving the MDVRPTW with route duration constraints. More recently, they proposed a parallel iterated Tabu Search heuristic for solving VRPTW, PVRPTW, MDVRPTW, and site-dependent VRPTW. Experimental results showed that the proposed method could

improve the computational speed by fully using the multiple cores available on the computers and provide competitive solutions (Cordeau et al., 2012).

Vidal et al. (2013) introduced a Hybrid Genetic Search with Advanced Diversity Control (HGSADC) for a large class of VRPTW, including the MDVRPTW. The proposed HGSADC can improve the solutions of some MDVRPTW benchmark instances. Besides, the HCSADC also performs well on some newly proposed larger instances with up to 960 customers, 12 depots, and two types of time windows. Besides, they concluded that the distribution and tightness of the time windows might strongly affect the performance of the methods and the quality of the solutions.

As for studies solely focused on the MDVRPTW, Polacek et al. (2004, 2008) first applied the Variable Neighborhood Search (VNS) to the MDVRPTWs. Experiment results showed that the VNSs are also powerful in solving the MDVRPTWs. Bettinelli et al. (2011) worked on the MDVRPTW with a heterogeneous fleet, namely the MDHVRPTW. They presented an exact algorithm, the branch-and-cut-and-price algorithm, for solving the MDHVRPTW. The proposed exact method can solve the problems with three types of vehicles, 100 customers and two depots. Without vehicle type constraints, the largest instance it can solve has 96 customers and four depots. Li et al. (2016) studied the MDVRPTW under shared depot resources, which allow vehicles not to end at the depot from which they start. They solved the problem with a hybrid genetic algorithm with an adaptive local search algorithm.

2.4 School Bus Scheduling Problem

In a school bus transportation system, bus stops are the essential elements to which students are assigned. A trip is a sequence of bus stops and their designated school. A bus route links several trips from different schools together and is assigned to a school bus. Given the trip information and the school bell times, the Bus Scheduling Problem (SBSP) aims to optimize bus schedules to serve all the trips.

The SBSPs could be classified based on the characteristics of the school bus transportation system, such as (1) the total number of depots (single or multiple); (2) the total number of schools (single or multiple); (3) district locations (urban or rural) and (4) problem scope (morning or afternoon). Numerous works solve the single depot problems in which each bus is required to start and end at the only depot (Kim, 2012; Fügenschuh, 2009, 2011; Wang, 2019). Bektaş and Elmastaş (2007) also worked on a single depot problem but considered the problem as an open VRP in which buses can end their tours at any point other than the depot. Löbel (1998) worked on the multi-depot vehicle scheduling problems in public transit. He formulated the problem as a multi-commodity flow problem and solved it by column generation. However, to the best of the authors' knowledge, no existing study is working on the multi-depot school bus scheduling problem. As for the school settings of the SBSPs, early studies are mainly on single-school problems (Park and Kim, 2010). As for the multi-school problems, some research decomposed the problem into several single depot subproblems (Chen et al., 2015). Others applied the metaheuristics (Braca et al., 1997).

Three nice reviews can be found in Park and Kim (2010), Wang et al. (2017), and Ellegood et al. (2020).

The objective of the SBSPs could be set based on three criteria: efficiency, effectiveness, and equity (Savas, 1978). To be specific, efficiency is related to the cost of the school bus system. Numerous research studies have focused on efficiency. And the most commonly used objectives are (1) to minimize the total number of buses; (2) to minimize the total traveling cost (i.e., traveling distance or traveling time); or (3) to minimize the weighted sum of the number of buses and the total vehicle traveling cost (Wang, 2019). Effectiveness is evaluated by how well the demand is satisfied and thus is related to students (Park and Kim, 2010). For example, one objective considering effectiveness could be to minimize students' walking distance. As for equity, it requires the system to be balanced relative to the busload route duration (or distance). For example, Shafahi et al. (2018) set the objective to minimize the total number of buses while minimizing the maximum route duration.

A broad range of constraints can be added to the SBSP. One unique type of constraint used in the SBSP is called trip compatibility constraint. As mentioned in Wang (2019), the trip compatibility constraint ensures that only compatible trip pairs are considered in the problem and thus helps to reduce the problem size. Other constraints include vehicle capacity, time windows, and maximum bus route length (or duration). The time window constraints are usually associated with school bell times and could potentially reduce the total number of buses (Fügenschuh, 2009, 2011).

The SBSPs for homogeneous buses with fixed start and end times can be considered a modified assignment problem or modified transportation problem (Kim et al., 2012). However, it's impossible to enumerate all possible routes for a large SBSP in a reasonable time. Therefore, the column generation method, which only generates the routes that have the potential to improve the objective function, becomes the methodology of choice (Fügenschuh, 2011; Wang, 2019). Besides, some metaheuristic methods are also used. For example, Chen et al. (2015) proposed a simulated annealing algorithm to minimize the number of buses and the total travel distance for a single-depot multi-school bus scheduling problem.

More recently, researchers have begun to consider the SBSP with stochasticity. Yan et al. (2015) formulated the inter-school bus routing and scheduling problem with stochastic travel time as a special multiple commodity network flow model. A heuristic algorithm based on a problem decomposition technique and variable fixing method is proposed and successfully solved a real-life problem with 400 passengers. Babaei and Rajabi-Bahaabadi (2019) formulated the simultaneous school bus scheduling and routing problem with stochastic time-dependent travel times as a bi-level problem and solved by a heuristic method that is a combination of ant colony optimization and a proposed route decomposition heuristic method. Wang et al. (2020) proposed a column generation-based stochastic school bell time and bus scheduling optimization, which also takes the stochastic travel time into account. Caceres et al. (2017) proposed a chance-constrained programming approach for a general multi-depot multi-school bus scheduling problem (MDSBSP) with the consideration of three types of uncertainty,

namely (1) the bus overcrowding condition; (2) the probability that a bus is late to school; (3) the expected maximum ride time of a student on any bus. The MDSBSP is first decomposed into several multi-depot single-school subproblems. And each subproblem is then solved with a column-generation-based algorithm.

2.5 School Bell Time Optimization

In a particular district, the demand for school buses is usually unbalanced since the bell times of most schools are around the same time. If we can change the bell time of some schools, more trips will become compatible so that more trips can be linked together to create a route for each bus. Therefore, the number of buses needed overall can be potentially reduced. Though school bell time optimization is beneficial, only a few studies have focused on school bell time optimization. All of them are under the single-depot configuration to the best of the authors' knowledge. Most of those studies incorporated the school bell time optimization into the school bus scheduling problem and formed the integrated model, namely SBSPTW. The SBSPTW is very difficult because the schools and trips interact with each other. If the school bell times are changed, then the trip starting times should be adjusted accordingly (Fügenschuh, 2011). Besides, when the schools have multiple trips, to ensure that all the students can come to school (or go back home) on time, the starting times of the trips belonging to the same school should be around the same time (Wang, 2019).

Fügenschuh (2009) formulated the integrated optimization of school starting times and bus route schedules as an integer programming problem and solved it with branch-and-

cut techniques. The results showed that the integrated model could reduce the total number of buses by 10-25%. Besides, wider school starting time windows may potentially lower the number of buses needed overall but also make the problem more difficult to solve (i.e., longer solution time and larger optimal gap). Later in 2011, Fügenschuh presented a set partitioning relaxation formulation and a primal construction heuristic to improve the solutions to the same problem (Fügenschuh, 2011). The largest real-world dataset used in both studies is collected in a German county which contains 191 trips and 82 schools.

Kim et al. (2012) considered both SBSPTW with a homogeneous fleet and a heterogeneous fleet. They solved the problems with assignment problem-based exact method and heuristic method. Wang (2019) formulated an integrated deterministic model to solve school bus routing, bell time, and scheduling, considering the maximum ride time and vehicle time. Considering the stochasticity of the travel time, Wang and Haghani (2020) proposed a column generation-based stochastic school bell time and bus scheduling optimization. The proposed model has two stages in which the first stage optimizes the bus schedules, and the second stage optimizes the bell time. The model is tested on a real-world dataset from a public-school transportation system with 286 trips and 93 schools. Results showed that, on average, 20% of buses could be saved with bell time optimization. Miranda et al. (2021) proposed three bell time strategies for improving the efficiency of the rural school bus system in Brazil. Results showed that the proposed bell time adjustment strategies could achieve up to 9% savings on the total cost, including the fixed cost of buses and the traveling cost.

Bertsimas et al. (2019) consider the school bell time optimization as a subproblem of their proposed bi-objective Routing Decomposition (BiRD) algorithm for optimizing schools' start times and bus routes. They first constructed several routing scenarios for each school and then formulated the school bell time optimization as a multi-objective generalized quadratic assignment problem. The proposed algorithm has been implemented in Boston and has led to $5 million in yearly savings.

Changing school start times can also have some adverse effects on communities. For example, later school start times might negatively interfere with students' after-school activities, and thus schools prefer schedules with lower absolute deviation in start times from the current schedule (Banerjee and Smilowitz, 2019). Therefore, they aimed to reduce the disutilities associated with changing school start times using a minimax model and solved the model using a lexicographic minimax approach.

2.6 Research Gaps

This Chapter presented a literature review on the studies related to the school bus scheduling problem and school bell time optimization, including the VRP, MDVRP, MDVRPTW, SBSP, and SBSPTW. The developments of the solution methods and algorithms for each of those problems are described.

According to the literature review, most of the existing approaches to the SBSP formulate the problem as a single-depot problem to simplify the complicated real-world

situation. However, in reality, multiple depots may exist in the network, and vehicles are required to operate from different depots.

Second, only a few studies work on school bell time optimization. One study solved it as a subproblem, and the others incorporated the school bell time optimization into the SBSP, which led to the SBSPTW. However, all the existing studies are based on the single-depot configuration and solved as an SDSBSPTW. Therefore, incorporating the school bell time optimization into the multi-depot multi-school bus scheduling problem is a major challenge in this study. Although the existing studies can provide some insights on how to optimize the school bell time, it is still a difficult task because (1) each school may have multiple trips, and (2) schools and trips are coupled with each other while the customers in the VRPTW are independent. How to synchronize the starting time of the school and trips is also challenging.

Therefore, this research will try to address these problems. First, this study will focus on the multi-depot bus scheduling problem (MDSBSP). Then, the school bell time optimization will be incorporated into the multi-depot school bus scheduling problem and form the integrated model, namely, the MDSBSPTW. The mathematical formulation and the heuristic methods will be proposed to solve the MDSBSPTW. And the performance of the proposed methods will be tested on different test problems.

Chapter 3: Problem Description and Model Formulation

This chapter proposes a mathematical formulation to solve the multi-depot multi-school bus scheduling problem with school bell time optimization (MDSBSPTW). First, the problem is thoroughly described, and the model's assumptions are listed. Then, the problem is formulated as a mixed-integer programming (MIP) problem. The mathematical model formulation is presented along with the explanations. Depending on the settings of the depot and the bell time window, the proposed MIP model can solve the multi-depot problems with or without bell time optimization (MDSBSPTW or MDSBSP) and the single-depot multi-school scheduling problems with or without school bell time optimization (SDSBSPTW or SDSBSP).

3.1 Problem Description

In a particular multi-depot multi-school system, a bus trip includes a sequence of bus stops and their designated school. For example, a bus trip in the afternoon starts from a school and then visits several stops sequentially, dropping off students at each stop until the bus becomes empty. A morning trip visits bus stops first and has school as the last stop. This study assumes the trips are fixed (i.e., the visiting sequence of bus stops on each trip is known). Based on the trip information, a bus route is a sequence of bus trips from different schools that are linked together to be served by one bus. Each bus route is then assigned to a single bus. Buses usually park at different depots. And each bus is required to start and end at the same depot. Therefore, after each bus departs

from the depot, it serves one bus route by visiting multiple trips from various schools in sequence, and after its journey, it will return to the same depot from which it starts.

This study aims to minimize both the total number of buses and the total deadhead duration. The total deadhead duration is the total time buses run without students on board. Take the bus route in Figure 5, for example. This route serves two bus trips, namely Trip *A* and Trip *B*. Specifically, Trip *A* has five stops, and Trip *B* has four stops. And Trip *B* follows Trip *A*. The total deadhead time of this route consists of:

(1) The total time traveling from the depot to the first stop of the first trip (Trip *A*);

(2) The total time traveling from the last stop of Trip *A* to the first stop of Trip *B*;

(3) The total time traveling from the last stop of the last trip (Trip *B*) to the same depot from which the bust starts.

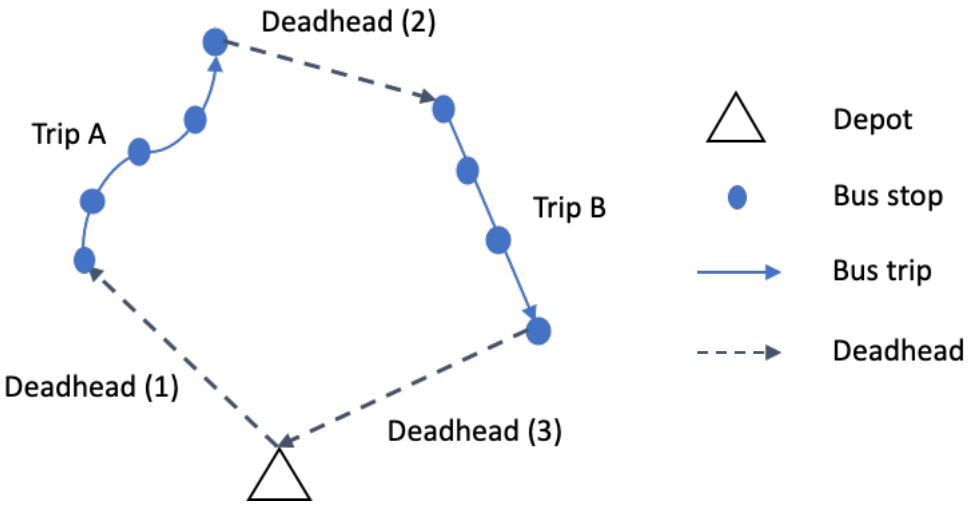


Figure 5. Illustration of the total deadhead duration of a route

For simplification, we call the sum of deadhead (1) and deadhead (3) the deadhead between trips and depots and call deadhead (2) the deadhead between trips.

We also incorporate the school bell time optimization into the school bus scheduling problem. Even small changes in the school bell times can make more trips compatible. Each bus route can become longer by serving more trips. Therefore, the overall number of buses can be reduced. This study aims to solve the multi-depot multi-school bus scheduling problem with bell time optimization (MDSBSPTW). The goal is to optimize the bus schedules to serve all the trips at minimum cost and find the best school bell time for each school within a given time window simultaneously.

3.2 Model Assumptions

Depots, schools, trips, and buses are the major components of this problem. The following assumptions are made regarding each of these components.

1. Depots
    - The depot locations are known;
    - The capacity of each depot is known.
2. Schools
    - Each school has a hard school bell time window;
    - All the trips belonging to the same school depart at the school bell time.
3. Bus trips
    - The trip information is known, including the sequence of visiting stops on each trip and the location of the first stop and the last stop on each trip;
    - Each trip has its fixed service time duration, and that is known.

By combining the location information of depots and trips (i.e., first stop and last stop), the deadhead between trips and depots and the deadhead between any pair of trips can

be calculated from Google Map Distance API. Considering the stochasticity of travel times, the option "travel time with traffic" is chosen.

4. Buses

- The fleet type is homogenous;
- The bus capacity is larger than the maximum load of the trips;
- All the buses arrive at schools on time;
- Every bus should start and end at the same depot.

5. Other assumptions

- Idle time is not considered in this study;
- Students have no special needs (homogeneous population).

3.3 Mathematical Formulation

3.3.1 Notation

Table 2 is a summary of the notations used in this model.

Table 2. Summary of the notations for the MIP model formulation

Variable	Description
x_{ij}^{s}	Binary variable equals 1 if trip i is served followed by trip j on the same route starting from depot s
y_k	Integer variable, the school k's bell time (or dismissal time)
Parameter	**Description**
SCH	Set of schools
T	Set of bus trips
S	Set of depots
SD	Set of start depot trip pairs
ED	Set of end depot trip pairs
E	Set of possible compatible trip pairs when $i, j \in T$

E'	Set of all the possible compatible trip pairs ($SD \cup ED \cup E$)
A_k	Set of discrete school bell timestamps for school k
tt_i	The travel time of the trip i
dd_{ij}	The deadhead time from trip (or depot) i to trip (or depot) j
EBT_k	The earliest allowable bell time for school k
LBT_k	The latest allowable bell time for school k
λ	The minimum time interval
$O\{i\}$	The school to which trip i belongs
M_b	The large coefficient for prioritizing the "total number of buses" term
f_c	The operation cost per bus per day
R_c	The traveling cost per minute
M	A very large positive number (big-M)
$Mcap_s$	The maximum capacity of each depot s

The school bell time window is denoted by (EBT_k, LBT_k), where EBT_k is the earliest allowable bell time and LBT_k is the latest allowable bell time. Each school must start (or dismiss) at some discrete timestamps in the given time window based on the minimum time interval λ. The minimum time interval λ can be set to any positive value, such as one minute, five minutes, or 10 minutes. Once the minimum time interval λ is chosen, the continuous time window (EBT_k, LBT_k) is discretized into a set of discrete timestamps as defined as A_k. And the elements in A_k can be written as:

$$EBT_k + n * \lambda \tag{3.1}$$

where $n \in \left[0, \frac{LBT_k - EBT_k}{\lambda}\right]$, and integer.

For example, if the given bell time window of school k is from 15:00 to 15:20, and the minimum time interval λ is set to be one minute, then we will have 21 one-minute discrete timestamps (e.g., 15:00, 15:01, and 15:02) that are just as well as continuous time. Those discrete timestamps are included in A_k and School k can only dismiss at

one of those timestamps. But if the minimum time interval λ is set to be five minutes, only five elements are in A_k, that is, 15:00, 15:05, 15:10, 15:15, and 15:20. School k can only dismiss at one of those five timestamps. In this study, the minimum time interval λ is set to be one minute.

The set of all the possible compatible trip pairs $E'(SD \cup ED \cup E)$ includes the trip pair (i, j), which satisfies one of the following conditions:

- (s, j) and $\forall \mathrm{s} \in \mathrm{S}, j \in T$. This is the start depot trip pair and stored in set SD.
- (i, s) and $\forall i \in T, \mathrm{s} \in \mathrm{S}$. This is the end depot trip pair and stored in set ED.
- Trip pair (i, j) $\forall i, j \in T$ is compatible if 1) i, j are not the same trip; and 2) the departure time of trip i plus the travel time of trip i and the deadhead time from trip i to trip j is less than or equal to the departure time of trip j. These compatible trip pairs are stored in set E.

  Mathematically, it can be written as:

$$bell\ time_{O\{i\}} + tt_i + dd_{ij} \leq bell\ time_{O\{j\}}, \forall i, j \in T \qquad (3.2)$$

Instead of enumerating all possible trip pairs, only compatible trip pairs in $E'(SD \cup ED \cup E)$ are used in the model. This can reduce the total number of variables.

3.3.2 Model Formulation

Based on the model assumptions, the mixed-integer programming (MIP) formulation for solving the multi-depot multi-school bus scheduling problem with school bell time optimization (MDSBSPTW) is presented below.

Objective

$$\textit{Minimize} \qquad M_b \sum_{s\in S}\sum_{j\in T} x_{sj}^{s} + \sum_{s\in S}\sum_{(i,j)\in E'} dd_{ij}x_{ij}^{s} \tag{3.3}$$

Or

$$\textit{Minimize} \qquad f_c \sum_{s\in S}\sum_{j\in T} x_{sj}^{s} + R_c \sum_{s\in S}\sum_{(i,j)\in E'} dd_{ij}x_{ij}^{s} \tag{3.4}$$

Subject to:

$$\sum_{i:(i,j)\in E'}\sum_{s\in S} x_{ij}^{s} = 1, \forall j \in T \tag{3.5}$$

$$x_{sj}^{s'} = 0, \forall s \in S; j \in T; s' \in S \setminus \{s\} \tag{3.6}$$

$$\sum_{i\in S}\sum_{j\in S}\sum_{s\in S} x_{ij}^{s} = 0 \tag{3.7}$$

$$\sum_{j:(i,j)\in E'} x_{ij}^{s} = \sum_{h:(h,i)\in E'} x_{hi}^{s}, \forall i \in T, s \in S \tag{3.8}$$

$$\sum_{j\in T} x_{sj}^{s} = \sum_{i\in T} x_{is}^{s}, \forall s \in S \tag{3.9}$$

$$y_{O_{\{i\}}} + tt_i + dd_{ij} - M \times \left(1 - x_{ij}^{s}\right) \leq y_{O_{\{j\}}}, \forall (i,j) \in E, s \in S \tag{3.10}$$

$$\sum_{j\in T} x_{sj}^{s} \leq Mcap_s, \forall s \in S \tag{3.11}$$

$$x_{ij}^{s} \in \{0,1\}, \forall (i,j) \in E', s \in S \tag{3.12}$$

$$A_k = EBT_k + n\lambda, n \in \left[0, \frac{LBT_k - EBT_k}{\lambda}\right] \; and \; integer, k \in SCH \tag{3.13}$$

$$y_k \in A_k \; and \; integer, k \in SCH \tag{3.14}$$

As for the objective function, we have two forms. Both are used to minimize the total number of buses and the total deadhead duration. Here the total deadhead duration includes the deadhead time between trip pairs and the deadhead time between trips and

depots. The first objective function (3.3) is suited for prioritizing minimizing the total number of buses by giving a very large coefficient M_b to the "total number of buses" term. This means that a bus scheduling plan with a shorter total deadhead time is worse than the other plans with a higher total deadhead time but fewer buses. Because the annual fixed cost for each bus is between $50,000 and $100,000 in the state of Maryland (Shafahi et al., 2018), we choose M_b as $ 80,000 in this study.

While the second one (3.4) is a cost function that minimizes the total bus operation cost per day. Suppose the total school days of a certain year is d_{sch}, then the operation cost of each bus per day f_c is $\$ (M_b/d_{sch})$. The traveling cost per minute R_c is set to be $1 per minute. More generally, f_c and R_c can also be considered as the weights of the two terms in the objective function, respectively. Therefore, they can be set to any value (even zero) for different research purposes (i.e., prioritizing either term).

Constraints (3.5) ensure that each trip can only be visited once. Constraints (3.6) show the consistency of the depot index. Constraints (3.7) eliminate the traffic between depots, which means that we cannot dispatch vehicles between depots. Constraints (3.8) and (3.9) are the conservation of flow constraints for the trips and depots, respectively.

Constraints (3.10), (3.13), and (3.14) relate to bell time optimization. To be specific, the school k's bell time window is (EBT_k, LBT_k), where EBT_k is the earliest allowable bell time, and LBT_k is the latest permissible bell time. Based on the minimum time interval λ, which is set to be one min in this study, the continuous time window

(EBT_k, LBT_k) is discretized into a set of discrete timestamps as defined as A_k. And Constraints (3.13) show the elements in A_k. Similar to Wang's work (Wang, 2019), Constraints (3.10) are the trip compatibility constraints. They ensure that for every possible trip pair (i, j), they are served on the same bus only if the finish time of Trip i (i.e., its departure time $y_{O_{\{i\}}}$, plus its travel time tt_i) plus the deadhead time from Trip i to Trip j (dd_{ij}) is less than or equal to the departure time of Trip j ($y_{O_{\{j\}}}$). Each school is required to dismiss at only one of the discrete timestamps in A_k. Constraints (3.11) are depot capacity constraints. The maximum capacity of each depot cannot be exceeded. Constraints (3.12)-(3.14) are domain constraints.

The proposed MIP model can solve the SDSBSPTW if the depot set S only has one element. It can also solve the multi-depot problems (or single-depot problems) without the bell time optimization if the bell time for each school is fixed at a particular timestamp. For solving the problems without bell time optimization, we need to make some changes in the formulation. Specifically, we don't need the decision variable y_k and the constraints (3.13) and (3.14). For Constraints (3.10), the school bell times $y_{O_{\{i\}}}$ and $y_{O_{\{j\}}}$ should change to the given bell times of the schools to which Trip i and Trip j belong, respectively. Therefore, for the problems without bell time optimization, the goal is only to find the best bus schedule that minimizes the total number of buses and total deadhead duration under the given school bell time plan.

3.4 Summary

This chapter proposed a mixed-integer programming formulation for solving the MDSBSPTWs. First, the problem was clearly stated. Then, the model assumptions were introduced. Finally, the mathematical model was presented with a detailed explanation of the objective function and the constraints. The objective function is to minimize the total number of buses and total deadhead duration and has two forms for different research purposes. One is used for prioritizing the total number of buses, and the other is a cost function that minimizes the total bus operation cost per day. The constraints mainly include the trip assignment constraints, conservation of flow constraints for both depots and trips, trip compatibility constraints, depot maximum capacity constraints, and the constraints for determining the best bell time for each school. The proposed MIP model can also solve single-depot problems with school bell time optimization and the problems without school bell time optimization.

Chapter 4: Two-Phase Heuristic Method

This chapter presents a two-phase heuristic method, that is, the first-route second-assignment method, for solving the multi-depot multi-school bus scheduling problem with school bell time optimization (MDSBSPTW). By adding a virtual depot and ignoring all the depot information, the MDSBSPTW is first converted into a single-depot multi-school bus scheduling problem with bell time optimization (SDSBSPTW) formulated as a mixed-integer programming model in the first-route phase. The goal is to use the minimum number of bus routes to serve all the trips. Then, those bus routes are the input for the second-assignment phase that decides the best depot for each bus. It is formulated as an integer programming model. We then introduce a Simulated Annealing-based Greedy Algorithm method (SA-GDA) to solve large size SDSBSPTWs more efficiently in the first-route phase. The SA-GDA method combined with the assignment model proposed in the second-assignment phase is the improved two-phase heuristic method. This chapter includes introductions to the proposed two-phase heuristic method, model assumptions, the mathematical model formulations for both phases, and detailed explanations of the SA-GDA framework.

4.1 Problem Description

The problem is the same as described in Section 3.1 in Chapter 3, which focuses on developing an efficient bus plan for a multi-depot multi-school system and optimizing the school bell times. The ultimate goal is to serve all the trips with a minimum number of buses and deadhead duration and find the best bell time for each school.

4.2 Two-Phase Heuristic Method

We first present a two-phase heuristic method, the first-route second-assignment approach for solving the MDSBSPTW. The goal is to solve large-size MDSBSPTWs more efficiently than the MIP model proposed in Chapter 3. The general idea is to construct a minimum number of bus routes to serve all the trips in the first-route phase. And then, in the second-assignment phase, we decide on the best starting depot for each bus. The key is to add a virtual depot in the first phase. The following subsections show each phase's input, constraints, objective, and output.

4.2.1 First-route Phase

By adding a virtual depot, the first-route phase is a single-depot multi-school bus scheduling problem with bell time optimization (SDSBSPTW).

(1) **Input**: A virtual depot, bus trips, deadhead time between any pair of bus trips, schools, and the given school bell time window for each school.

(2) **Constraints**:

1) Every trip should be served exactly once;
2) Each school bus is required to start its route from the virtual depot and return to the virtual depot after serving several bus trips in sequence;
3) The bell time of each school is constrained within a given time window.

(3) **Objective:**

1) Minimize the total number of scheduled buses and the total deadhead time between trips;
2) Find the best school bell time for each school.

(4) **Output:** School bus scheduling plan in which all the buses start from the virtual depot and return to the virtual depot, and the school bell time for each school.

4.2.2 Second-assignment Phase

(1) **Input**: School bus scheduling plan from the first phase, the deadhead duration matrix between trips and depots.

(2) **Constraints**:

1) Every bus should be assigned to only one depot;

2) The maximum capacity of each depot cannot be exceeded.

(3) **Objective**: Minimize the total deadheads between trips and depots.

(4) **Output**: The starting depot for each bus.

By combining the results from both phases, we can obtain the complete school bus scheduling plan and the best bell time for each school. Specifically, the total number of buses can be found from the outputs of the first-route phase. The total deadhead duration is the sum of the deadhead between trips (from the first-route phase) and the deadhead between trips and depots (from the second-assignment phase).

An example is shown in Figure 6. There are eight trips and three depots in the original multi-depot problem. After adding a virtual depot and doing the first-route phase optimization, results show that three bus routes are needed to serve those eight trips. The first route (blue dot lines with arrow) serves three bus trips. The second route (orange dot lines with arrows) serves four trips, and the final route marked in green only serves one trip. Those three bus routes should start and end at the virtual depot in

the first phase. The second-assignment phase decides the starting depot for each bus without violating the depot capacities. The solid lines with an arrow show the final assignment. Those three buses are assigned to three different depots.

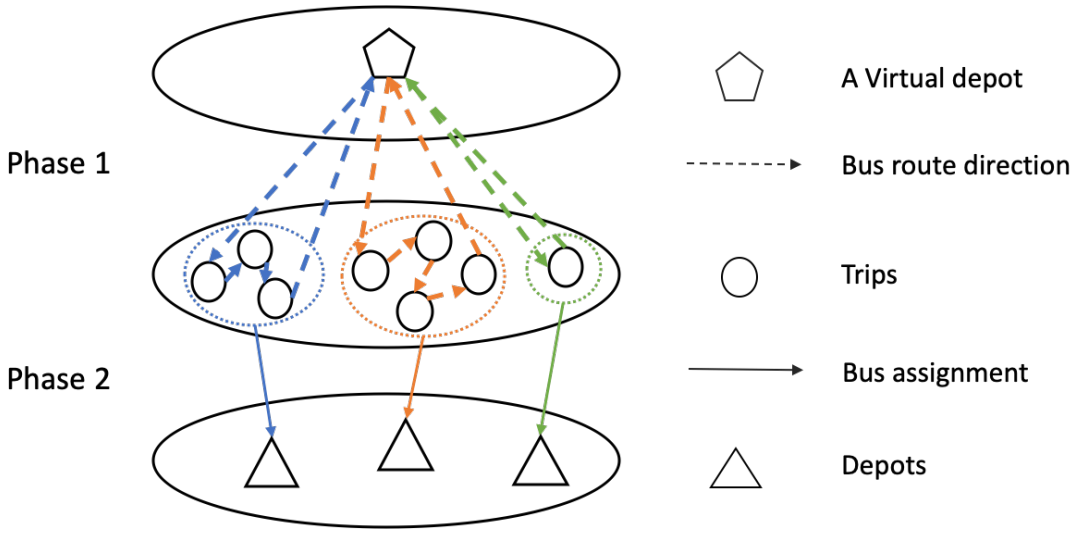


Figure 6. Example of the two-phase heuristic method for MDSBSPTW

When solving the MDSBSPTW with the proposed two-phase heuristic method, a Mixed-integer Programming (MIP) model is formulated for the first-route phase, which is an SDSBSPTW. It can also solve the problem without the bell time optimization (i.e., SDSBSP). And then, an Integer Programming (IP) model is proposed for the bus assignment problem in the second phase. The assumptions and formulations for both phases are provided in the following subsections.

4.3 Model Assumptions

Depots, schools, trips, and buses are the major components of this problem. The model assumptions are the same in Section 3.2 in Chapter 3. A brief summary is provided

here. As for each depot, the location and the capacity are known. As for each school, its bell time is constrained within a given time window. We set the minimum time interval to be one minute, and then the bell time must be in discrete time slots of one minute. Besides, each school may have multiple trips, and all the trips belonging to the same school depart at the school bell time.

As for each trip, its travel time is fixed and known. And the trip information is known as well, including the sequence of visiting the stops on each trip and the location of the first stop and last stop on each trip. Based on the location of depots and trips, the deadhead duration between trips and depots and the deadhead duration between trips are queried from Google API. Specifically, the "travel time with traffic" in Google API is chosen and used. As for buses, each bus has the same capacity (a homogeneous fleet) and must return to the same depot from which it starts.

4.4 Mathematical Formulation

We first introduce the notations used in both phases (Table 3). Then, the MIP model formulation for the first-route phase and the IP model formulation for the second-assignment phase are provided.

4.4.1 Notation

Table 3 summarizes the notations used in this two-phase heuristic method. For the first-route phase, we have two sets of decision variables, including x_{ij} for determining the trip connection on each bus route and y_k for determining the best bell time for each

school. While for the second-assignment phase, we only have one set of decision variables called z_{bs} for assigning each bus b to the best depot s.

Table 3. Summary of the notations for the two-phase model formulation

Variable	Description
x_{ij}	Binary variable equals 1 if trip i is served followed by trip j
y_k	Integer variable, the school k's bell time (or dismissal time)
z_{bs}	Binary variable equals 1 if bus b is assigned to depot s
Parameter	**Description**
SCH	Set of schools
T	Set of bus trips
S	Set of depots
B	Set of possible buses
S'	The virtual depot
SD	Set of start depot trip pairs
ED	Set of end depot trip pairs
E	Set of possible compatible trip pairs when $i, j \in T$
E'	Set of all the possible compatible trip pairs ($SD \cup ED \cup E$)
A_k	Set of discrete school bell timestamps for school k
tt_i	The travel time of the trip i
dd_{ij}	The deadhead time from trip i (or depot) to trip j (or depot)
EBT_k	The earliest allowable bell time for school k
LBT_k	The latest allowable bell time for school k
λ	The minimum time interval
$O\{i\}$	The school to which trip i belongs
M_b	The large coefficient for prioritizing the "total number of buses" term
f_c	The operation cost per bus per day
R_c	The traveling cost per minute
M	A large positive value (big-M)
c_{bs}	The sum of the deadhead from the last stop of the last trip on bus route b to depot s and the deadhead from depot s to the first stop of the first trip on bus route b
$Mcap_s$	The maximum capacity of each depot s

4.4.2 Model Formulation for First-route Phase

Objective

$$\textit{Minimize} \quad M_b \sum_{j \in T} x_{S'j} + \sum_{(i,j) \in E} dd_{ij} x_{ij} \tag{4.1}$$

Or

$$\textit{Minimize} \quad f_c \sum_{j \in T} x_{S'j} + R_c \sum_{(i,j) \in E} dd_{ij} x_{ij} \tag{4.2}$$

Subject to:

$$\sum_{i \in T \cup S'} x_{ij} = 1, \forall j \in T \tag{4.3}$$

$$\sum_{i \in T \cup S'} x_{ji} = 1, \forall j \in T \tag{4.4}$$

$$\sum_{i \in T} x_{iS'} = \sum_{j \in T} x_{S'j} \tag{4.5}$$

$$y_{O_{\{i\}}} + tt_i + dd_{ij} - M \times \left(1 - x_{ij}\right) \leq y_{O_{\{j\}}}, \forall (i,j) \in E \tag{4.6}$$

$$A_k = EBT_k + n\lambda, n \in \left[0, \frac{LBT_k - EBT_k}{\lambda}\right] \textit{ and integer}, \forall k \in SCH \tag{4.7}$$

$$y_k \in A_k \textit{ and integer}, \forall k \in SCH \tag{4.8}$$

$$x_{ij} \in \{0,1\}, \forall i,j \in T \cup S' \tag{4.9}$$

For multi-depot problems, we first introduce a virtual depot, and the objective is to minimize the total number of buses and the total deadhead duration only between trips. Similar to the MIP model presented in Chapter 3, we have two forms for the objective function here as well. The first one (4.1) is suitable for prioritizing minimizing the total

number of buses by using the very large coefficient M_b. The second one (4.2) is a cost function by adding the coefficient f_c and R_c for the "total number of buses" term and the "total deadhead duration between trips" term, respectively. It is used to minimize the total operation cost, including the bus cost and the deadhead travel per day.

But if the problem is a single-depot problem, the virtual depot is set to be the only actual depot. Besides, instead of only minimizing the deadhead duration between trips, we change the second term of the objective function (Eq.4.1 or Eq.4.2) to $\sum_{(i,j)\in E'} dd_{ij}x_{ij}$ so that the total deadhead duration, including the deadhead between trips and the deadhead duration between trips and depots, is minimized. Therefore, we can directly get the final solution to the single-depot problems in the first-route route without going into the second-assignment phase.

Constraints (4.3) ensure that each trip can only have one preceding trip. Constraint (4.4) ensure that each trip can only have one succeeding trip. Constraints (4.5) show the conservation of flow; that is, the total number of buses departing from the virtual depot should be equal to those arriving at the depot. Based on the minimum time interval λ, which is set to be one minute in this study, the continuous time window (EBT_k, LBT_k) is discretized into a set of discrete timestamps stored in A_k. And Constraints (4.7) show the elements in A_k. Constraints (4.6) are the trip compatibility constraints. They state that for every possible trip pair (i, j), They can be served on the same bus only if the finish time of trip i (i.e., its departure time $y_{O_{\{i\}}}$, plus its travel time tt_i) plus the deadhead time from trip i to trip j (dd_{ij}) is less than or equal to the departure time of

the trip j ($y_{O_{\{j\}}}$). And each school k can only dismiss at one of the discrete timestamps in A_k. Constraints (4.8)-(4.9) are domain constraints.

The proposed MIP model can also solve the problems without the bell time optimization if the bell time for each school is known and fixed. The goal is to find the best bus schedule that minimizes the total number of buses and the deadhead duration between trips under the given school bell times. Therefore, we don't need the decision variable y_k and the constraints (4.7) and (4.8). For Constraints (4.6), the school bell times $y_{O_{\{i\}}}$ and $y_{O_{\{j\}}}$ should change to the given bell times of the schools to which Trip i and Trip j belong, respectively.

4.4.3 Model Formulation for Second-assignment Phase

Objective

$$\textit{Minimize} \quad \sum_{b\in B}\sum_{s\in S} c_{bs}z_{bs} \tag{4.10}$$

Subject to:

$$\sum_{s\in S} z_{bs} = 1, \forall b \in B \tag{4.11}$$

$$\sum_{b\in B} z_{bs} \leq Mcap_s, \forall s \in S \tag{4.12}$$

$$z_{bs} \in \{0,1\}, \forall b \in B, \forall s \in S \tag{4.13}$$

The inputs for the second-assignment phase are the bus routes generated in the first-route phase. Each bus route can only be assigned to one single bus. And the goal of this phase is to determine the best depot for each bus without violating depot capacities.

The objective function of the second-assignment phase (4.10) minimizes the total deadhead time between trips and depots. For each bus route b, it is the sum of the deadhead duration from the first stop of the first trip on the bus route b to the depot s and the deadhead duration from the depot s to the last stop of the last trip on the bus route b. And Eq. 4.10 minimizes the deadhead duration for all the buses. Constraints (4.11) ensure that each bus can only be assigned to one depot. Constraints (4.12) are depot capacity constraints. Different buses can be assigned to the same depot, but the total number of buses assigned to the depot can't exceed the depot's maximum capacity. Constraints (4.13) are the domain constraints.

After finishing both phases, we can have the complete solution (i.e., the bus schedule and the best bell time for each school). For multi-depot problems, the total number of buses can be found from the first-route phase. The total deadhead duration is the sum of the deadhead duration between trips (from the first phase) and the deadhead duration between trips and depots (from the second phase). For single-depot problems, we can directly get the complete solution after finishing the first-route phase.

4.5 Improved Two-Phase Heuristic Method

The first-route phase is an SDSBSPTW. Due to its NP-hard nature, it may not be efficiently solved by the exact method when the problem size is relatively large or has a large bell time window. Since the final goal of the first-route phase is to find the best combination of school bell times such that the total number of buses and the deadhead duration between trips are minimized, some local search strategies can be applied to speed up the solution searching process. Therefore, we propose a hybrid heuristic method, namely, the Simulated Annealing-based Greedy Algorithm (SA-GDA) method, to replace the MIP model in the first-route phase for efficiently solving complicated SDSBSPTWs. Since the SA-GDA is designed for the single-depot problems, we still introduce a virtual depot for the multi-depot problems so that they can be converted into a single-depot problem in the first-route phase. We keep the assignment model in the second-assignment unchanged for determining the best depot for each bus route. Therefore, the SA-GDA method combined with the assignment model produces the improved two-phase heuristic method.

The SA-GDA method can solve the multi-depot bus scheduling problems with or without bell time optimization and the single-depot bus scheduling problems with or without bell time optimization. For problems with bell time optimization, the simulated annealing algorithm tries out different school bell time plans. Under each fixed bell time plan, the proposed greedy algorithm is used to find the best bus schedule. For multi-depot problems with bell time optimization (i.e., MDSBSPTWs), the best bus schedule is the one that minimizes the total number of buses and the deadhead duration

between trips (Eq.4.1 or Eq.4.2). But for single-depot problems (i.e., SDSBSPTWs), there is only one depot in the system, and there is no need for doing the bus-depot assignment later. Therefore, the virtual depot is set to be the actual depot, and the total deadhead duration, including the deadhead duration between trips and the deadhead duration between trips and depots, is calculated. So, for SDSBSPTWs, the results from the SA-GDA method are the final solution. But for the MDSBSPTWs, the buses obtained from the SA-GDA method should be assigned to different depots to minimize the deadhead duration between trips and depots in the second phase. Then, the complete solution to the MDSBSPTWs is obtained. For both MDSBSPTWs and SDSBPTWs, after embedding the greedy algorithm into the simulated annealing algorithm framework, the overall SA-GDA method compares different bus schedules and returns the best one and its corresponding school bell time plan.

For problems without the bell time optimization (i.e., MDSBSP or SDSBSP), the simulated annealing algorithm is not used as school bell times are given and fixed. Only the greedy algorithm is used in the first-route phase. For MDSBSPs, the greedy algorithm finds the best bus schedule that minimizes the total number of buses and the deadhead duration between trips under the given school bell times. We then pass the bus routes from the greedy algorithm to the assignment model in the second phase to get the deadhead duration between trips and depots to obtain the final solution. But for SDSBSPs, since there is only one depot, the virtual depot is set to be the only depot. Without doing the extra depot assignment, we can directly use the greedy algorithm to calculate the total deadhead duration, including the deadhead duration between trips

and the deadhead duration between trips and depots. Therefore, only using the greedy algorithm in the first-route phase can obtain the final solution that minimizes the total number of buses and the total deadhead duration for SDSBSPs.

4.5.1 Greedy Algorithm

The proposed Greedy Algorithm (GDA) is used to come up with the best bus schedule that minimizes the total number of buses and the deadhead duration between trips (Eq. 4.1 or Eq. 4.2) for multi-depot problems. For single-depot problems, the goal is to minimize the total number of buses and the total deadhead duration. The flow chart of the proposed GDA method is shown in Figure 7.

If the total number of unique bell times in the school bus system is N, we then divide the trips into N different groups based on the bell time so that the trips in the same group g_n $(n = 1, \cdots, N)$ have the same bell time. Those trip groups are saved in $sortedTG$ in ascending order of the bell time. Since the mixed load is not allowed, the trips in the same group can't be served on the same bus. In other words, N is the maximum length of a bus route. Figure 8 shows an example of an afternoon school bus scheduling problem. We have four schools but only have three unique dismissal times. Therefore, the total number of trips that each bus can serve cannot exceed three. For a random trip i, its bell time, which is DT_i, is in the $Rank(DT_i)th$ position among all the bell times. Because the trip groups are ordered, the group to which trip i belongs is $g_{Rank(DT_i)}$. Besides, $usedtrip$ and $Finalroute$ are used to store the used trips and the built routes, respectively. The $Finalroute$ is returned after the GDA is finished.

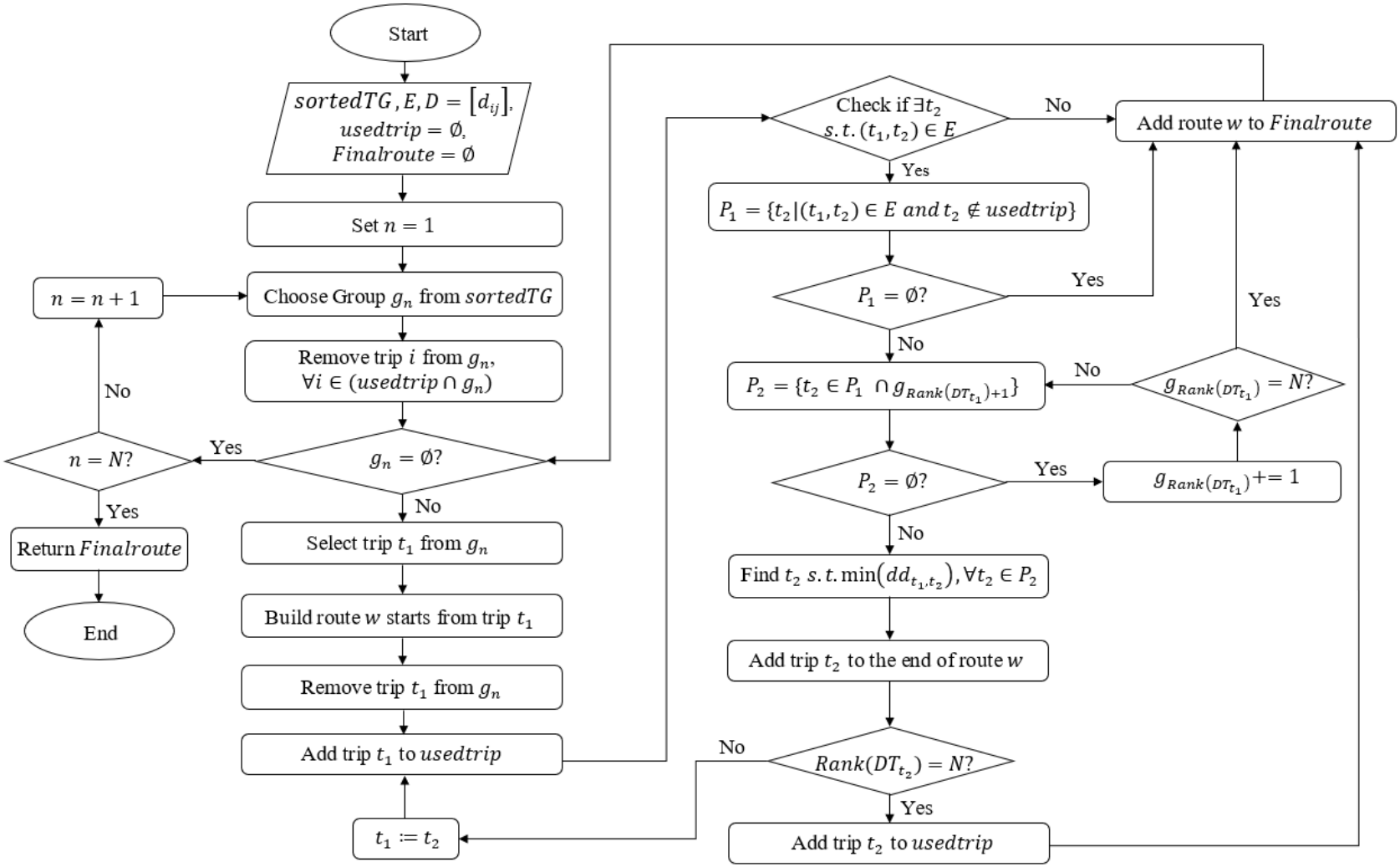
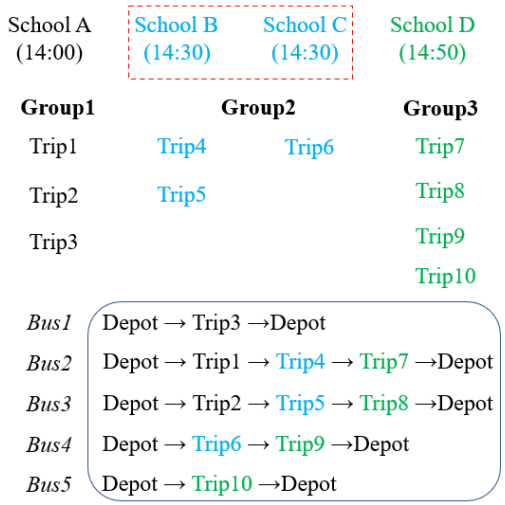


Figure 7. Flow chart of the proposed GDA method

To minimize the total number of buses, we try to connect as many trips from different trip groups as possible on a single bus. And the earlier bell time the first trip has, the more trips the bus is more likely to serve. Therefore, after the bus w leaving the depot, the first trip t_1 it serves is first randomly chosen from the group g_1 in which trips have the earliest bell time. Then, we check if any unused trips are compatible with trip t_1. If not, it is a single-trip bus (e.g., Bus 1 in Figure 8). We return it and start a new bus route. As for the new route, if there are some unused trips in g_1, the new bus route still first serves the trip from g_1 (e.g., Bus 2 and Bus 3). If all the trips in g_1 have been used and there are still trips unvisited, we start to build the bus route whose first trip is from the next trip group g_2 and so on. For example, the first trip on Bus 4 is from the group g_2, while the first trip on Bus 5 is from the group g_3.



Trip	TT	Trip Compatibility
Trip1	10	Trip4(5), Trip6(7), Trip7(10)
Trip2	15	Trip5(7), Trip6(7.5), Trip8(10)
Trip3	55	-
Trip4	9	Trip7(3), Trip8(7), Trip9(10)
Trip5	15	Trip8(2)
Trip6	12	Trip9(5), Trip8(8)

TT: Travel Time (in min); Trip Compatibility: first element shows the compatible trip, the number in the bracket shows the deadhead duration (in min).

Figure 8. Example of the proposed GDA method

If there are trips compatible with the first trip t_1, we continue checking if any of those trips are in the next group g_2. If yes, we choose the one (trip t_2) which minimizes the deadhead duration $dd_{t1,t2}$. Otherwise, we move to the next group g_3, do the same

check, and so on. By doing so, we connect more trips on a single bus and minimize the total deadhead duration. Once trip t_2 is determined, we seek the succeeding trip of the trip t_2 based on the same process. Once no more trips can be added to the bus route w, we return it and start a new bus route. The set $usedtrip$ and all the trip groups keep updating to ensure that each trip can only be visited once. The whole process is repeated until all the trips are visited. And the set $Finalroute$, which includes all the constructed bus routes, is returned.

The size of the set $Finalroute$ is the final total number of buses. Then, the total deadhead duration between trips is calculated for the multi-depot problems. While the total deadhead duration, including the deadhead between trips and the deadhead between trips and depots, is calculated for the single-depot problems. We can combine the total number of buses and the total deadhead duration between trips (or the total deadhead duration) into one value Z based on Equation (4.1) or Equation (4.2) for solution comparisons. The solution with a smaller Z is better.

4.5.2 Simulated Annealing Algorithm

The Simulated Annealing (SA) Algorithm is an iterative improvement algorithm that imitates the annealing process in metallurgy (Kirkpatrick et al., 1983). Its strength is that it can jump out of the local minima by accepting solutions that are worse than the current solution with some probability. The original SA has been famous for its good performance in solving some combinatorial problems such as Vehicle Routing Problem (VRP), Travelling Salesman Problems (TSP), etc.

The temperature T is the key parameter in the SA algorithm. The initial temperature T_0 is usually set to a very high value and is cooled down very slowly until reaching the frozen temperature T_{min} according to a specific cooling schedule. At the initial temperature T_0 an initial state is randomly selected, and the resulting initial solution is calculated. The state and the corresponding solution, take the TSP, for example, is a permutation of the cities to be visited and the corresponding total traveling cost. The neighboring states of the current state are the set of permutations produced based on operations like swapping or reversing.

But the SA here is used to try out different bell time plans. Each state here refers to a bell time vector $DT_0 = \{a_k | k \in SCH\}$, where a_k is a randomly selected bell time for school k from the given bell time window (EBT_k, LBT_k). The solution to each state (i.e., each bell time plan) is the bus schedule that minimizes the total number of buses and the total deadhead duration between trips (or the total deadhead duration for single-depot problems) calculated based on the bell time vector DT_0 using the proposed GDA method. Given the current state, we can create multiple new bell time vectors. Each new bell time vector DT is called a neighboring state of the current state. For creating a new bell time vector DT, each school randomly chooses a new bell time from the neighborhood of its current bell time. The neighborhood for each school is set to be within ± 5 min of its current bell time in this study. If the resulting DT is infeasible (i.e., some schools' new bell times violate the given bell time window), another random bell time vector will be generated until feasibility is satisfied. Under the new feasible bell time vector, the best bus schedule is also calculated using the GDA method.

Figure 9 shows an example of an afternoon school bus scheduling problem with dismissal time optimization. There are three schools, and their original dismissal time (in minutes) are 800, 830, and 900, respectively. The earliest dismissal time EBT_k and the latest dismissal time LBT_k for each school are also given, which are ±30 min of the original dismissal time. If we generate a neighborhood dismissal time vector as 803, 825, and 905, it is an acceptable neighborhood because (1) they all fall in the corresponding dismissal time window (EBT_k, LBT_k), and (2) each school has its dismissal time changed within ±5 min of the current dismissal time. However, if the neighborhood is generated as 806, 827, and 903, though they all fall in the given time window, it is not acceptable as the change that school A made is beyond 5 min. A new dismissal time vector must be generated until all requirements are satisfied.

Original Dismissal time

School A	School B	School C
800	830	900

$(EBT_{k'}\, LBT_k)$

School A	School B	School C
(730,830)	(800,900)	(830,930)

One acceptable neighborhood

School A	School B	School C
803	825	905

One unacceptable neighborhood

School A	School B	School C
806	827	903

Figure 9. Examples of the neighboring bell time vectors of a given bell time vector

A predetermined number of iterations are performed at each temperature of the SA algorithm to determine the current best solution. At each iteration, the algorithm randomly chooses one neighboring state (i.e., a new bell time plan) of the current state (i.e., the current bell time plan) to generate a new solution (i.e., the bus schedule under

the new bell time plan) based on the proposed GDA method. If the new solution is better than the current best solution, it replaces the current best solution. Otherwise, it is accepted with some probability. The temperature will decrease, and the same process will repeat at each temperature until the frozen temperature is reached. Then, the best school bus schedule and the associated school bell time plan are returned.

4.5.3 Overall SA-GDA method

Figure 10 shows the pseudo-code of the proposed SA-GDA method. It is built based on the framework of the SA algorithm. The GDA method is embedded into it to find the best bus schedule under each bell time plan created by the SA method.

1	Initialize $T_0, T_{min}, \alpha, K, T = T_0, it = 0, it_{max}$
2	Construct an initial bell time DT_0
3	Calculate Z_0 based on the initial bell time DT_0 using the GDA method
4	**while** $T > T_{min}$
5	**for** $it = 0$ **to** it_{max}
6	Generate a neighbor bell time DT
7	Calculate Z based on the bell time DT using the GDA method
8	**if** $Z < Z_0$
9	$Z_0 = Z$
10	B$T_0 = BT$
11	**else**
12	Calculate probability p, $p = \exp\left(\frac{Z_0 - Z}{T}\right)$
13	**if** $p > Random[0, 1]$
14	$Z_0 = Z$
15	$DT_0 = DT$
16	$T = \alpha T$

Figure 10. Pseudocode of SA-GDA method

We first initialize the initial temperature T_0, the frozen temperature T_{min}, the constant K ($K = 1$ in this study. In nature, it is Boltzmann's constant), and the maximum iterations it_{max} performed at each temperature. Based on previous studies and preliminary experiments, T_0 is set such that a solution 20% worse than the initial solution has a 50% chance to be accepted and T_{min} is set such that a solution that is inferior by 1% relative to the current solution is accepted with a probability of 0.1% (Wei et al., 2018). The geometric cooling schedule ($t = t * \alpha$) is chosen to decrease the temperature gradually, and the cooling rate α is set to be 0.95.

The SA algorithm starts with a random initial bell time vector $DT_0 = \{a_k | k \in SCH\}$, where a_k is a randomly selected bell time for school k from the given bell time window (EBT_k, LBT_k). The initial solution Z_0, which is the best initial bus schedule, is obtained by the proposed GDA method. Starting at the initial temperature T_0, we conduct it_{max} attempts and then decrease the temperature according to the geometric cooling schedule. At each attempt, we randomly generate a new bell time vector DT in a neighborhood of the current bell time vector. The neighborhood refers to being within ±5 min of the current bell time for each school. If the resulting DT is infeasible (i.e., the resulting bell time of some schools violates the given bell time window), another random bell time vector DT will be generated until feasibility is satisfied. Then, we compute the new solution Z based on DT. If $Z < Z_0$ the new solution is unconditionally accepted. Otherwise, the worse solution is accepted with the probability $p = \exp((Z_0 - Z)/T)$. The algorithm is terminated once the frozen temperature T_{min} is reached. And the best school bell time plan and its corresponding school bus schedule are returned.

For the single-depot problems, the result from the SA-GDA method is the final solution that minimizes the total number of buses and the total deadhead duration. But for the multi-depot problems, we still need to move on to the second-assignment phase to do the bus-depot assignment. The buses from the SA-GDA method are the input for the assignment model in the second-assignment phase. The output is the depot-bus assignment that minimizes the total deadhead duration between trips and depots. Combining the results from both phases gives us the final solution.

4.6 Summary

This chapter proposed a two-phase heuristic method, the first-route second-assignment heuristic, to solve the MDSBSPTWs. The first-route phase is an SDSBSPTW and is formulated as a mixed-integer programming model. The second-assignment phase is a bus assignment problem that assigns the buses to the depots and is formulated as an integer programming model. The developed mathematical models were presented with a detailed explanation of the objective function and constraints.

Then, we proposed a Simulated Annealing-based Greedy Algorithm method (SA-GDA) to solve large size SDSBSPTWs efficiently in the first phase. The greedy algorithm, simulated annealing algorithm, and SA-GDA method were explained in detail. Combined with the second-assignment phase, we have the improved two-phase heuristic method. Both the two-phase heuristic method and the improved two-phase heuristic method can also solve SDSBSPTWs or the problems without the bell time optimization with some modifications.

Chapter 5: Tabu Search-based Ant Colony Optimization

This Chapter presents a solution for the MDSBSPTWs without dividing the problem into different phases. It is the Tabu Search-based Ant Colony Optimization (TS-ACO) method. The TS-ACO method is under the Tabu Search framework, in which the Tabu Search (TS) method is used to examine different school bell time plans. Under each school bell time plan, the ACO method is used to find the best bus schedule that provides the minimum number of buses and total deadhead duration.

5.1 Ant Colony Optimization

Ant Colony Optimization (ACO) was first introduced by Dorigo and colleagues (1996). It is a population-based metaheuristic that efficiently uses probabilistic techniques to find approximate solutions to complex optimization problems. ACO mimics the foraging behavior of real ants for seeking the shortest path between their colony and the food. Ants start from the nest and initially explore the environment randomly. For ants that have found the food, they will deposit the chemicals called pheromone on the ground for guiding other ants in finding food. And ants are more likely to follow the paths with higher pheromone levels. It turns out that the ant that has discovered a shorter or more efficient path between the food source and nest can commute between the food source and nest more frequently than another that uses a longer route. This makes the shorter path has a higher pheromone level which is more attractive to the other ants. And as more ants use a path, the pheromone level of that path grows stronger.

Therefore, the solution (i.e., the path from the nest to the food source) can be gradually improved, and the shortest path can be identified eventually.

For solving the MDSBSPs using ACO, we assume that there are several ants in the nest, and each ant represents a solution. A solution includes the bus routes that visit all the trips without violating the depot capacity constraints and the trip compatibility constraints. Because all the trips are visited, it is also called a complete tour of the ant. For building such a tour, we first transform the school bus scheduling problem into a Time-dependent Directed Scheduling Graph where each node is a trip, and the directed edges are the compatible trip pairs (Haghani and Banihashemi, 2002). This guarantees trip compatibility. In the constructed graph, the nodes (i.e., the bus trips) are the food sources, and each ant should visit all of them in its tour. As there is more than one depot in the MDSBSP, we consider the multiple depots as the nest's different entrances (or exits), and ants can randomly choose a depot to start their journey.

Since each node in the graph (i.e., each bus trip) should be visited exactly once without violating the trip compatibilities, the ant may need to build several routes to visit all the trips. The ant can randomly choose a depot to start for each route without violating the depot capacity constraints. Besides, it should start and end at the same depot. Determining the trips visited on each route is based on trip compatibility and the probability related to the pheromone level and visibility level. This will be introduced in detail in the following subsections. A complete tour for a two-depot seven-trip problem is shown in Figure 11. We assume the depot capacity for each depot is two.

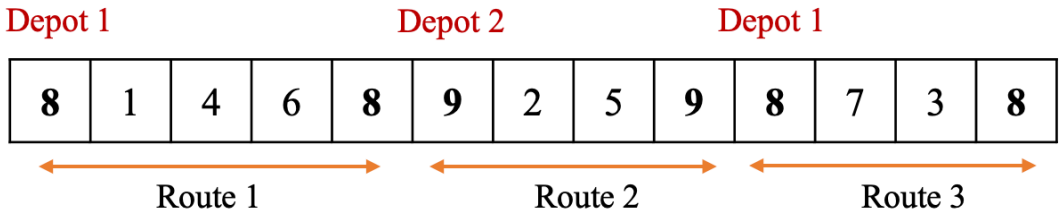


Figure 11. An example of the ant's complete tour

If we have n trips and m depots in a particular school bus system, we use numbers 1 to n to represent trips and $n + 1$ to $n + m$ to represent depots. In the example shown in Figure 11, the ant first chooses to start from Depot 1 (i.e., the first element of the first route, $8 - 7$) and serves Trip 1, Trip 4, and Trip 6 in sequence without violating trip compatibility constraints. Since every trip can only be visited once, the succeeding trip j should be labeled as visited when the ant moves from Trip i to Trip j and should never be revisited. Once no more trips can be added to the route, the ant must return to the same depot where it starts, namely Depot 1. And the capacity of Depot 1 should decrease by one. Since each depot still has some capacity, the ant can randomly choose a depot to continue its journey. It starts from Depot 2 for its second route, covering another two trips. The capacity of Depot 2 should be updated to one. The final route starts and ends at Depot 1 and includes the remaining two trips. Overall, the ant needs three routes to visit all the trips. And the depot capacities are 0 and 1 at the end, respectively. Because we build the routes based on trip compatibility constraints, ensure that each trip can be only visited once, and keep updating depot capacity, the feasibility of the complete tour is guaranteed. After having the complete tour, the corresponding total deadhead duration is also calculated and saved.

It should be noticed that Figure 11 only illustrates one complete tour created by one ant. Since we assume multiple ants are in the nest, each ant will build such a tour. Their solutions will be compared, and the best one will be returned. The detailed process of the proposed ACO method is described in the following subsections.

5.1.1 Initialization

For using the proposed ACO method to solve the MDSBSPs, we should first initialize some parameters shown in Table 4.

Table 4. Parameters of the ACO

Parameter	Description	Value
$Iter_{max}$	Maximum number of iterations	100
A_n	Number of ants	20~30
α	The magnitude of the pheromone intensity	10
β	The magnitude of visibility	2
ρ	Evaporation rate of the pheromone	0.1

We perform the ACO for at most $Iter_{max}$ Iterations to gradually improve the solution. The algorithm may stop earlier if there is no significant improvement after a predetermined maximum number of iterations. Specifically, if the improvement is less than 1% after five consecutive iterations, the whole procedure of the proposed ACO will stop. A total number of A_n ants are required to build their own complete tour at each iteration. The tour construction process can be broken down into a series of node-to-node movements for each ant. Given the current node i, the next node j is chosen based on some probability related to the pheromone intensities and the visibility intensities. The parameters α and β are the magnitudes for the pheromone intensities and the visibility intensities in the probability function, respectively.

We also consider pheromone evaporation in the proposed ACO. Based on the pheromone evaporation rate ρ ($0 < \rho < 1$), the pheromone intensities of each arc will evaporate gradually based on some rules described in subsection 5.1.4. It can help to reduce the probability of early convergence to a locally optimal solution. After a series of tests on the value of those parameters, we chose the final values that gave the best results and listed them in Table 4.

5.1.2 Solution Construction

A total number of A_n ants are used to create ant solutions at each iteration. Each ant is required to build a complete tour (i.e., visit all the trips) without violating the depot capacity constraints and the trip compatibility constraints. For each ant k, Figure 12 shows the solution construction process.

For each ant k, the tour construction process can be broken down into a series of node-to-node movements. It first starts from a randomly chosen depot s. Then, it determines the subsequent trips to visit. Generally, the visiting sequence of the trips on each route is determined based on the probability function given in Eq. 5.1, where p_{ij}^k (or p_{sj}^k) is the probability of the ant k determining the next trip j at the current trip i (or depot s). Since all the bus routes are required to start and end at the same depot, we assume that the ant will directly return to its starting depot if no more trips can be added to the existing route. Therefore, we don't calculate the pheromone intensity and the visibility level between the last trip i of a route and the depot s, and the corresponding p_{is}^k.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha}\eta_{il}^{\beta}}, & if\ j \in N_i^k \\ 0, & otherwise \end{cases} \tag{5.1}$$

where τ_{ij} and η_{ij} are the pheromone intensity and visibility level of edge ij, respectively. If the ant is currently at the depot s, then τ_{sj} and η_{sj} are the pheromone intensity and visibility level between the depot s and the trip j. N_i^k is the feasible set including ant k's all the unvisited trips l that are compatible with the current trip i.

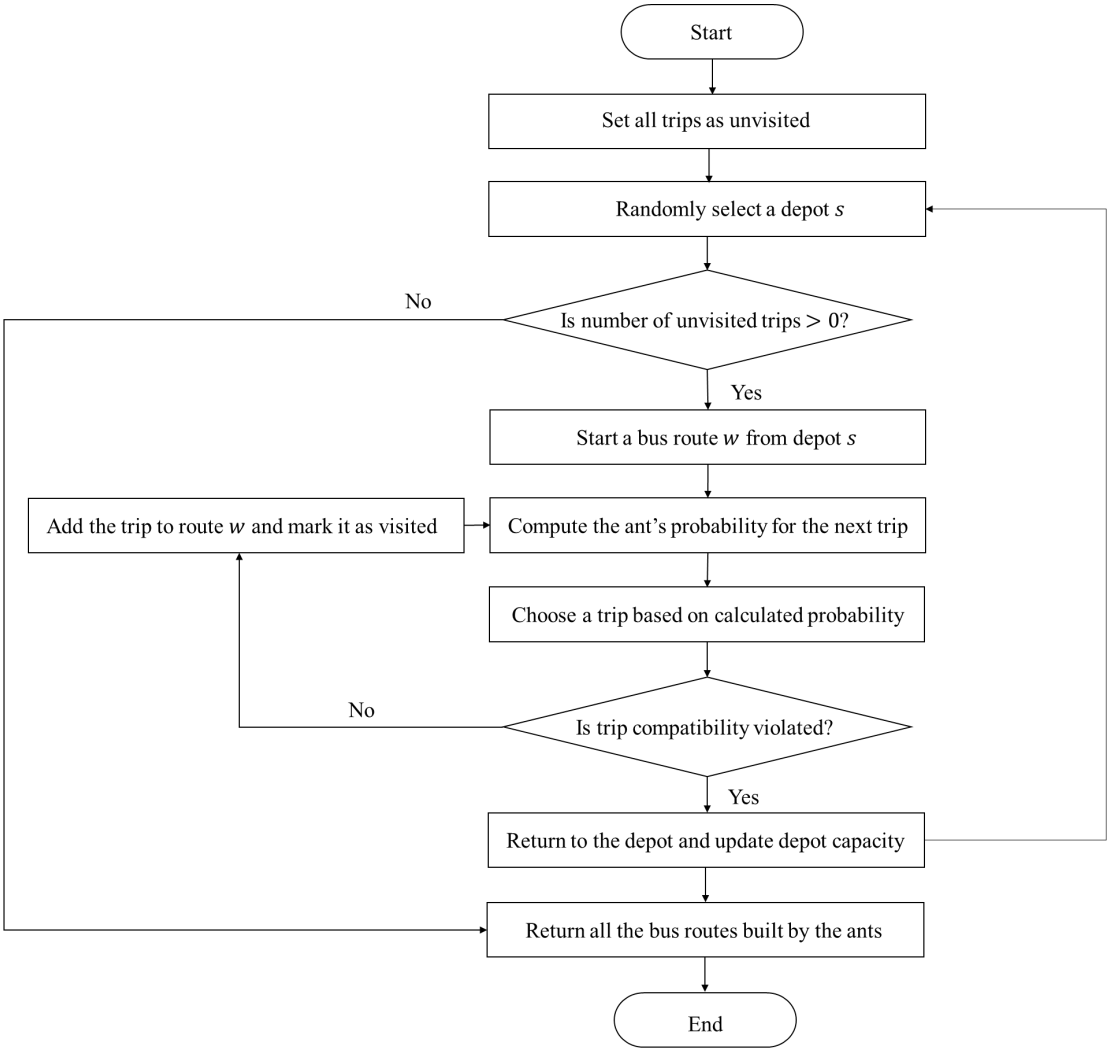


Figure 12. The solution generation process of each ant

We need to initialize all the edges' pheromone intensities and visibility levels before the algorithm starts. The initial visibility levels are initialized differently for depot-trip pairs and trip-trip pairs. The initial visibility value η_{ij} between any two trips i and j is simpler, which is $\eta_{ij} = \frac{1}{dd_{ij}}$ where dd_{ij} is deadhead duration between trip i and trip j. It is used to represent travel desirability. The shorter the deadhead duration dd_{ij} is, the more the ant is willing to choose the edge from trips i to trip j.

As for the initial visibility value η_{si} between the depot s and the trip i, it is calculated as $\eta_{si} = \frac{1}{dd_{si}}(latestBT - bell\ time\ of\ the\ school\ to\ which\ trip\ i\ belongs)$, where dd_{si} is the deadhead duration between the depot s and the trip i, and $latestBT$ is the latest bell time of the given school bus system. One way to reduce the total number of routes is to make individual routes longer, allowing the ant to serve more trips in a single route. To achieve this, the bell time of the first trip i on the route should be as early as possible. The additional term in the η_{si} as compared to η_{ij}, namely $(latestBT - bell\ time\ of\ the\ school\ that\ trip\ i\ belongs)$, shows the sequence of trip i's bell time in the whole system. If the bell time of trip i is among the earliest ones, the value of $(latestBT - bell\ time\ of\ the\ school\ to\ which\ trip\ i\ belongs)$ will be large, which will give us a high visibility value η_{si}. This makes trip i more likely to be chosen as the first trip on a bus route. As a result, the corresponding bus route can connect more trips and help potentially reduce the total number of buses.

The initial values of the pheromone intensities can be set to be some constants (Jabir et al., 2017). Or they can be determined based on some heuristic methods (e.g., the nearest neighbor heuristic) to accelerate the convergence of the ACO (Gambardella et al., 1999). We generate initial pheromone intensities for the proposed ACO using the Greedy Algorithm (GDA) proposed in Chapter 4. But the GDA is designed for solving the single-depot multi-school bus scheduling problems. It can only determine the connections among trips (i.e., the depot for each bus route is unknown). Therefore, we assign each bus to its nearest depot after obtaining the incomplete bus routes from the GDA method. Once the bus routes are fully determined, we calculate the total deadhead duration TD, including the total dead duration between trips and the total deadhead duration between trips and depots. And then the initial pheromone intensity τ_{ij} of edge ij is calculated based on the formula $\tau_{ij} = \frac{1}{TD}$.

However, TD is much larger than the deadhead between each pair of nodes dd_{ij} (or dd_{si}), so the initial pheromone value τ_{ij} (or τ_{si}) will be much smaller than the initial visibility value η_{ij} (or η_{si}). Especially for the edges between depots and trips, after considering the difference between the bell time of the trip i and the latest bell time in the system, the visibility value η_{si} will be much larger than τ_{si}. This will make the visibility value dominate the ant's edge selection and make pheromone intensities useless. Therefore, we add a coefficient to the original formula for calculating the initial pheromone intensity τ_{ij} (including τ_{si}), which is $\tau_{ij} = \frac{1}{TD} * 100$. By doing so, the initial pheromone value and the initial visibility value can have the same magnitude so that both terms can be effective for guiding the ants to find the shorter path.

Based on the initial pheromone intensities, initial visibility levels, and the corresponding magnitude α and β, we can calculate the initial probability matrices between depot to trip (p_{sj}^k) and trip to trip (p_{ij}^k). Then, ant k can begin its tour construction. It will start from a randomly selected depot s and then determine the first trip i based on the probability p_{si}^k. Suppose a trip i has an earlier bell time, a higher pheromone intensity between itself and the depot s, and a shorter deadhead duration between itself and depot s. In that case, it will be of higher probability to be chosen as the first trip of the ant's first route due to a high probability p_{si}^k. After determining the first trip i, the ant k starts its first route by moving from depot s to trip i. The depot s's capacity is reduced by one. On the first trip i, The ant searches for the next possible trip j based on the probability p_{ij}^k. This procedure continues until no more compatible trips can be added to the current route. Then, the ant returns to the starting depot s.

After returning to the nest, the ant starts its second route from one of those depots that still have capacities. The depot capacity is updated, and the above process is repeated for finishing the second route. For building a complete tour that visits all the trips, the whole solution construction process of the ant k is continued until all the trips are visited exactly once. When all the trips are visited, the algorithm computes the total number of bus routes and the corresponding total deadhead duration.

For each iteration, a total number of A_n ants build their own complete tour based on the above procedures. The one with the minimum objective (i.e., the minimum number

of buses and the total deadhead duration) in the current iteration is compared with the best solution so far. If there is any improvement, the best solution is updated and stored. Then, we start the next iteration. The procedure is repeated till the algorithm's stopping rules are reached. Two stopping criteria are used in this algorithm: (1) the maximum number of iterations $Iter_{max}$ is reached, and (2) the maximum number of iterations with no significant improvement is reached. For (2), if the improvement is less than 1% after five consecutive iterations, the procedure will stop.

5.1.3 Local Search Procedures

After obtaining the current best solution from the ACO of each iteration, we sequentially apply three local search schemes to improve the solution. They include trip-shift, bus assignment, and trip-swap procedures. First, without considering the current bus-depot assignment, the trip-shift operation is conducted to mainly reduce the total number of buses. Second, the bus assignment scheme determines the best depot location for each bus route. At last, the trip swap operator mainly reduces the deadhead duration within the bus routes belonging to each depot. The details of those local search procedures are shown below.

(1) Trip-shift procedure

The trip shift operator is mainly used to reduce the number of bus routes. For using it, we need to first ignore the current depot assignment plan from the ACO and only focus on the trip connections on the bus routes. To be specific, we randomly choose two bus routes (r_1 and r_2) every time. A total number of u trips (in this study, u is set to 1) is removed from the route r_1 and is inserted into the route r_2 without violating the trip compatibilities. There are three main possible outcomes:

- Case I: One new route r_1' or r_2', regardless of the changes in total deadhead duration (the total deadhead duration can be increased or decreased).

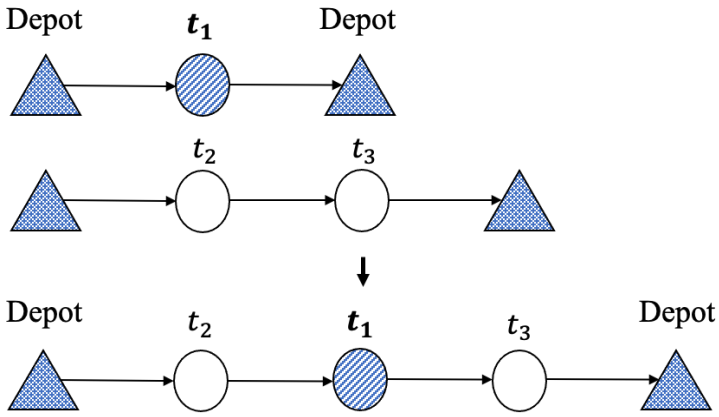


Figure 13. Output case I after the trip-shift operation

An example of Case I is shown in Figure 13. We first randomly choose two routes. Those two routes serve one trip and two trips, respectively. During the trip-shift operation, Trip t_1 from the first route is chosen. Luckily, it is compatible with Trip t_2 and Trip t_3 on the second route. Therefore, it is removed from the first route and inserted into the second route. This eliminates the first route while making the second route longer. The new route is saved, and the original two routes are deleted. We don't care about the changes in the deadhead duration here because the main goal of the trip-shift operation is to reduce the total number of buses.

- Case II: Two new routes r_1' and r_2', and the total deadhead duration of those two new routes is less than that of the original two routes.

An example of Case II is shown in Figure 14. Two routes are randomly chosen at first. The deadhead duration between trips for each of those two routes is 20 minutes and 25

minutes, respectively. Trip t_5 from the second route is chosen during the trip-shift operation. Since Trip t_5 is compatible with Trip t_2 on the first route, we remove it from the second route and insert it into the first route. After the trip-shift operation, the total number of buses is still two. The deadhead duration between trips for the two new routes is 27 minutes and 15 minutes, respectively. And the total deadhead duration between trips of those two new routes is 42 minutes which is smaller than the original two routes (i.e., 45 minutes). Therefore, we accept the two new routes and save them while deleting the original two routes.

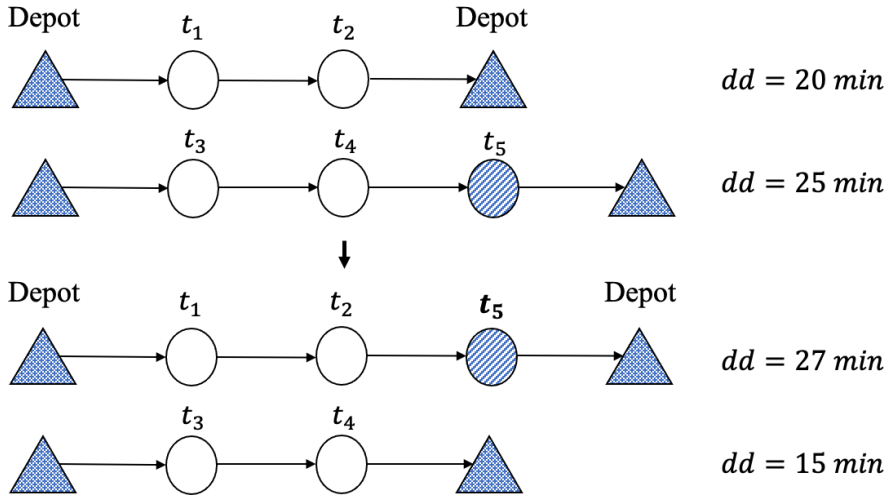


Figure 14. Output case II after the trip-shift operation

- Case III: Two new routes r_1' and r_2', and the total deadhead duration of those two new routes is larger than that of the original two routes.

Based on the same example of Case II, if the deadhead duration between trips for the two new routes is 27 minutes and 20 minutes, respectively (Figure 15). Then, the total deadhead duration between trips of those two new routes is 47 minutes which is larger than the original two routes (i.e., 45 minutes). In this study, if the total deadhead

duration is slightly larger than the original deadhead duration (the difference is set to be 5~10 min in this study), we still accept the two new routes. Otherwise, we should reject the two new routes and keep the original two routes unchanged.

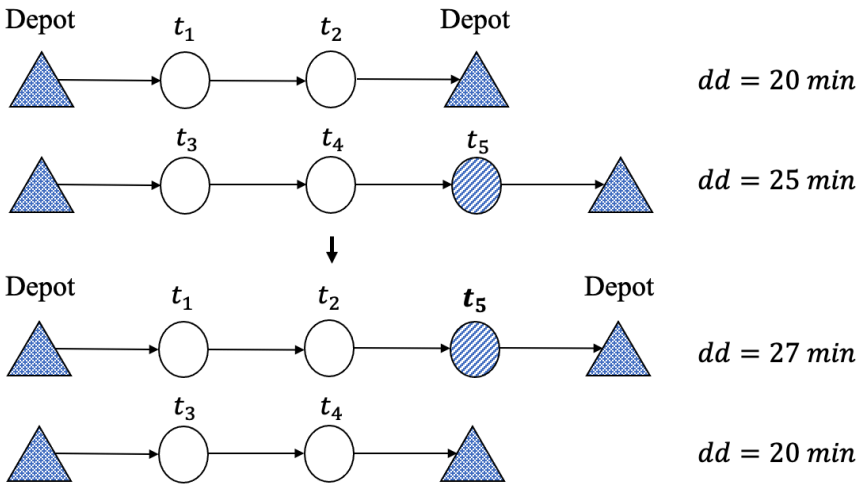


Figure 15. Output case III after the trip-shift operation

As a result, since we are interested in searching for better solutions, if the total number of routes is reduced (i.e., Case I), we accept the new route (r_1' or r_2') regardless of the deadhead duration of the new route. For Case II, though the total number of routes is unchanged, the total deadhead duration is reduced. We also accept the new routes. However, for case III, the new solution after the trip shift operation is worse than the original one. We will accept it only if the total deadhead duration is slightly larger than the original deadhead duration (the difference is set to be 5~10 min). We accept this "bad" move for potentially reducing the buses later. An example is shown in Figure 16. Otherwise, the resulting inferior solution is discarded.

As shown in Figure 16, three routes were generated after the ACO method. The deadhead duration between trips for those routes is 15 minutes, 10 minutes, and 9

minutes, respectively. First, the first and the second routes are chosen for the trip-shift operation. Trip t_2 from the first route is chosen. Since it is compatible with Trip t_4 on the second route, Trip t_2 is removed from the first route and is inserted into the second route. There are still three routes after the first trip-shift operation. However, the total deadhead duration between trips of the three new routes is 37 minutes, slightly larger than the original three routes (i.e., 34 minutes). We still accept the three new routes since the difference is only three minutes (less than 5 minutes).

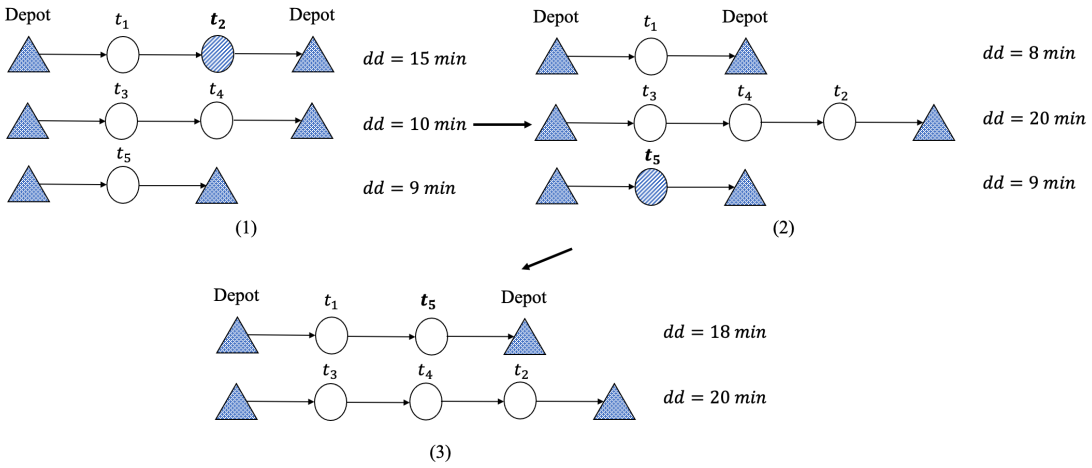


Figure 16. Another example for output case III after the trip-shift operation

Then, we start the second trip-shift operation in which the first and the third routes are chosen. Those two routes are both single-trip bus routes and are likely to be combined into one bus route. It turns out that Trip t_5 can be inserted into the first route, hence eliminating the third route. We can save one bus after conducting the trip-shift operation twice. Though the solution after the first trip-shift operation is slightly worse than the original one, it helps reduce the total number of buses in the later trip-shift operations. This example shows the benefit of accepting slightly worse solutions.

The trip-shift operation is conducted for multiple iterations until no significant improvement is achieved or the predetermined maximum number of iterations is reached. After that, we will move forward to the next bus assignment step.

(2) Bus assignment procedure

After the trip-shift procedure, we will determine the best depot from which each bus route should start. It is based on the same integer programming formulation we proposed in section 4.4.3 in Chapter 4. The goal is to determine the best bus-depot assignment, which minimizes the total deadhead duration between trips and depots for all the bus routes. For each bus route, the deadhead duration between trips and depots is the sum of the deadhead duration from the depot to the first trip on the route and the deadhead duration from the last trip on the route to the depot. The constraints guarantee that each bus can only be assigned to one depot without violating the depot capacities. If the original problem is a single-depot problem, we skip this step.

(3) Trip-swap procedure

After determining the starting depot for each bus route, the trip-swap operation is used to reduce the total deadhead duration. The trip-swap operator used in this study is $Swap(1,1)$. For the bus routes that start from the same depot, we randomly choose two bus routes (r_1 and r_2). Take the example of Depot 1 in Figure 17, for example. We randomly select one trip (i.e., t_2) from route r_1 and one trip (i.e., t_5) from route r_2. If those two trips have the same bell times, trip t_2 is swapped with trip t_5, that is, trip t_2 is removed from the route r_1 and inserted into the route r_2 at the position of trip t_5, while trip t_5 is removed from the route r_2 and inserted into the route r_1 at the position

of trip t_2. But if trip t_2 and trip t_5 have different bell times, the trips can still be swapped if the trip compatibilities are maintained for both bus routes after the swapping. Otherwise, a new trip-swap operation should be conducted until both new routes are feasible. The total deadhead duration of the new two routes is 28 minutes which is less than that of the original two routes (i.e., 40 minutes). So, we accept those two new routes. Otherwise, we keep the original routes.

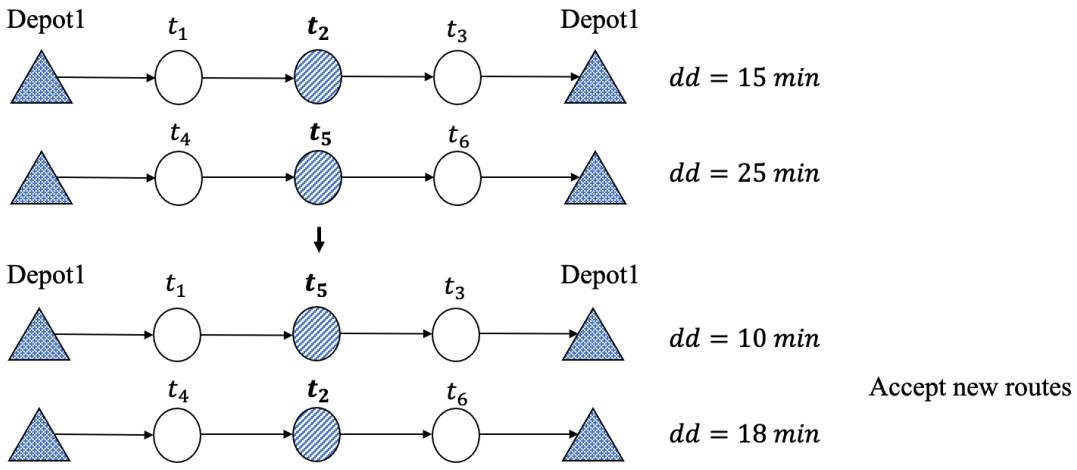


Figure 17. Example of the trip-swap operation

We conduct the trip-swap operation on its bus routes for multiple iterations for each depot until no significant improvement is achieved or the predetermined maximum number of iterations is reached. After the trip-swap operation, the best route plan will be returned and will be used to update pheromones.

5.1.4 Pheromone Update

The global pheromone evaporation is conducted for all edges to avoid a too rapid convergence of the proposed ACO. Besides, we will also increase the pheromone intensities for those edges included in the best solution after the local search

improvements. Given the best solution, the pheromone intensity on edge (i, j) at the current iteration t and the pheromone evaporation rate ρ, the pheromone intensity on edge (i, j) for the next iteration $t + 1$ is carried out using Eq. (5.2).

$$\tau_{ij}^{t+1} = \begin{cases} (1-\rho) * \tau_{ij}^{t} + \dfrac{1}{TD_{best}}, & if\ arc\ ij \in best\ solution\ so\ far \\ (1-\rho) * \tau_{ij}^{t}, & otherwise \end{cases} \quad (5.2)$$

where TD_{best} is the total deadhead duration of the best solution we found at the current iteration t, ρ is the evaporation rate, and τ_{ij}^{t} and τ_{ij}^{t+1} are the pheromone intensity on edge (i, j) at the current iteration t and the next iteration $t + 1$.

The pheromone evaporates on all edges based on the evaporation rate, and $(1 - \rho) * \tau_{ij}^{t}$ is the remaining amount of pheromone after evaporation. For those edges included in the best solution, the pheromone intensities are increased by an additional amount $\frac{1}{TD_{best}}$. Those edges have higher pheromone levels and will be more attractive to the ants in the next iteration, which helps accelerate the whole searching process.

5.1.5 Overall ACO Method

The overall ACO method is summarized in Figure 18.

Pseudocode of the overall ACO algorithm
Begin Initialize • Collected input data (bus trips, current school bell time) • Algorithm parameters ($A_n, Iter_{max}, \alpha, \beta, \rho$) • Pheromone trail matrix, τ

- Visibility matrix, η

While (Stopping criteria on iterations)

For ants, from 1 to A_n

- Build a complete tour

  While the number of unvisited trips > 0 **do**

  Randomly select a depot // *check depot capacity*

  Probability-based customer selection // *related to pheromone trails and visibility levels*

  Build bus routes // *check trip compatibility*

  End while

- Compute the objective function value, $S(ACO)$
- Compare and store the best solution, $S^*(ACO)$

End For

// applied local search

While (termination conditions)

- Do the trip shift // *reduce the total number of buses*
- Do the bus assignment // *find the best depot assignment; skip this step if it is a single-depot problem*
- Do the trip swap(1,1) within each depot // *reduce the total deadhead duration*
- Compare and store the best solution $S^*(ACO\_LS)$

End while

- Pheromone trials update using the best solution (including pheromone evaporation)

End while

End

Figure 18. Pseudocode of the proposed ACO algorithm

The overall ACO includes initialization, solution construction by the ants, three local search procedures for improving the solution, and the pheromone update procedure. The output is the best schedule that minimizes the total number of buses and the total deadhead duration under a fixed bell time plan. If the original problem is without the

bell time optimization, the proposed ACO can provide the final solution to the problem. Otherwise, we need to use the tabu search method to find the best bell time plan.

5.2 Tabu Search Method

The ACO method is used to find the best bus schedule, which minimizes the total number of buses and the total deadhead duration under each fixed bell time plan. When it comes to bell time optimization, the ACO should be embedded into some local search method to find the best combination of bell times. The simulated annealing algorithm presented in Chapter 4 could be one option. It accepts non-improving moves with some probability to escape from the local minima, but it may revisit a solution, fall back to a previous local optimum, or even cycle, which is a waste of time and resources. Therefore, we use the Tabu Search (TS) method to avoid revisiting a solution or cycling. The tabu search method is an iterative, memory-based neighborhood-search method (Glover et al., 1993). Like the simulated annealing algorithm, the tabu search method iteratively searches for a better solution. It also utilizes the tabu list data structure to forbid or penalize certain moves that would return to a recently visited solution, making it more efficient than other local search procedures. The tabu search method used in this study is described below in detail.

5.2.1 Initial Solution

The proposed tabu search method is used to find the best combination of school bell times. It starts from a random initial bell time vector $DT_0 = \{a_k | k \in SCH\}$, where a_k is a randomly selected bell time for school k from the given bell time window

(EBT_k, LBT_k). Then, the initial solution Z_0 is calculated based on DT_0 using ACO. It is the best bus schedule that minimizes the total number of buses and the total deadhead duration given the initial bell time DT_0 using ACO. Besides, the Z_0 and DT_0 are set to be the current best solution Z^* and DT^*.

5.2.2 Neighborhoods

The neighborhood of the current bell time is defined as a new bell time in which only a certain number of schools are allowed to change their bell times within a specific range. We construct n new bell times based on the current bell time. Specifically, for building each new bell time, we randomly choose m schools and only allow those schools to change their bell times in a neighborhood of their current bell times. The neighborhood refers to being within ±5 min of the current bell time for each school. Each bell time change is called a move. Suppose the resulting bell time is infeasible (i.e., the resulting bell time of some schools violates the given bell time window); in that case, another random bell time will be generated until feasibility is satisfied. We calculate the corresponding Z using ACO for each new feasible bell time DT.

5.2.3 Tabu List

The tabu list TL is used to avoid re-visiting of recent neighborhoods. It records the latest moves and is updated dynamically as the search proceeds. If a move is already in the tabu list, it is not allowed until it reaches a termination point. Specifically, the tabu list TL is initialized as an empty list with a fixed length before TS starts. As the search proceeds, at each iteration, we create n new bell times based on the current bell time. For each new bell time DT, we calculate its corresponding Z using ACO. We then

choose the best $\tilde{Z}$ among them and compare it with the current best solution Z^*. If $\tilde{Z} < Z^*$, the new solution $\tilde{Z}$ and its corresponding bell time plan $\widetilde{DT}$ are kept as the best solution. For the bell time $\widetilde{DT}$, we know which schools have their bell time changed. Those schools are added to the list for preventing cycling. As new moves enter the list, the list's size might exceed the predetermined length. If that happens, according to the First-In-First-Out rule, we delete the schools added at the earliest time.

5.2.4 Aspiration Criterion

The aspiration criterion is employed to override the tabu status. The schools in the tabu list change their bell times until they reach an expiration point. Only if the tabu list exceeds the predetermined size can the elements added at the earliest be deleted. In this study, we use the most intuitive aspiration rule to relax the tabu restrictions of some elements in the tabu list. If the aspiration criterion is met, those elements can be removed from the tabu list and used to provide better solutions. Specifically, if some schools in the tabu list happen to contribute to a better solution than the current best-known solution, we revoke the tabu status of those schools. Therefore, those otherwise-excluded schools can be used to produce a better solution.

5.2.5 Stopping Rules

The whole search process of the TS method is terminated once the maximum number of iterations is reached or no significant improvement is achieved after a certain number of consecutive iterations. Then, the best school bell time plan DT^* and its corresponding school bus schedule Z^* are returned.

5.3 Overall TS-ACO Method

We combined the tabu search method and the proposed ACO algorithm for solving the MDSBSPTWs. The overall proposed TS-ACO method is shown in Figure 19.

The tabu search method is used to examine different school bell time plans. Under each bell time plan, the ACO is used to find the best bus schedule that minimizes the total number of buses and the total deadhead duration. The proposed TS-ACO method can be used to solve SDSBSPTWs as well. If it is a single-depot problem, we simply skip the bus assignment procedure in the ACO while keeping other steps unchanged.

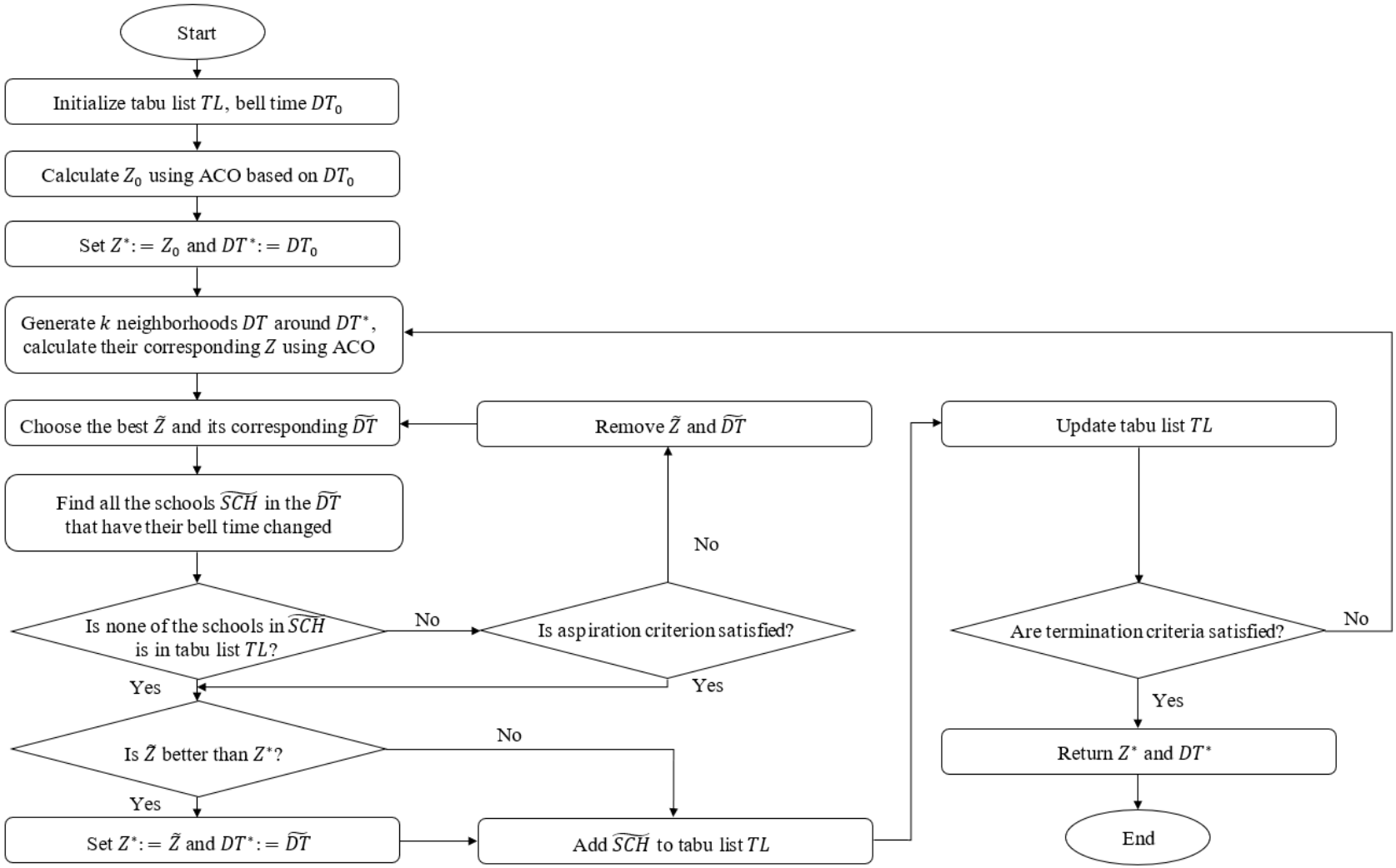


Figure 19. Flow chart of the proposed TS-ACO method

5.4 Summary

This chapter introduced the Tabu Search-based Ant Colony Optimization (TS-ACO) method for solving the MDSBSPTW (or SDSBSPTW) without dividing the problem into different phases. The TS method is used to examine different bell time plans. The ACO is embedded into it to find the best bus schedule that minimizes the total number of buses and the total deadhead duration under each fixed bell time plan. First, the ACO was described, including the initialization, solution construction by the ants, three local search procedures for improving the solution, and the pheromone update procedure. Then, the TS method was presented, including the initialization, neighborhood construction, tabu list settings, and aspiration criteria.

Chapter 6: Test Problems

In this chapter, five groups of test problems constituting a total of fourteen test problems with different characteristics are used to examine the performance of the proposed four methods, namely the mixed-integer programming model, the two-phase heuristic method, the improved two-phase heuristic method, and the TS-ACO method. All the test problems consider the school bus scheduling problem in the PM, that is, to find the best dismissal time plan and the optimal bus schedule to transport students from school back home. They all are derived from real-world data collected from a public school system in Maryland. The results of solving these test problems using the exact and the heuristic approaches are presented and analyzed.

6.1 Data Description

The real-world data was collected from a public school system in Maryland. There are, in total, 55 schools and 678 trips to serve all schools. Fifty-five schools consist of 31 elementary schools (ESs), 13 middle schools (MSs), and 11 high schools (HSs). The school district wants to find the optimal bus schedules to transport students from schools to their designated bus stops and optimize school dismissal time. The current dismissal times for all schools are provided. And they are 15:30, 15:00, and 14:15 for all ES, MS, and HS, respectively. For each bus trip, its fixed travel duration and the locations of its first stop and last stop (i.e., longitudes and latitudes) are given. All the trips belonging to the same school depart at the same time. All school buses are parked at 28 different depots, as shown in Figure 20.

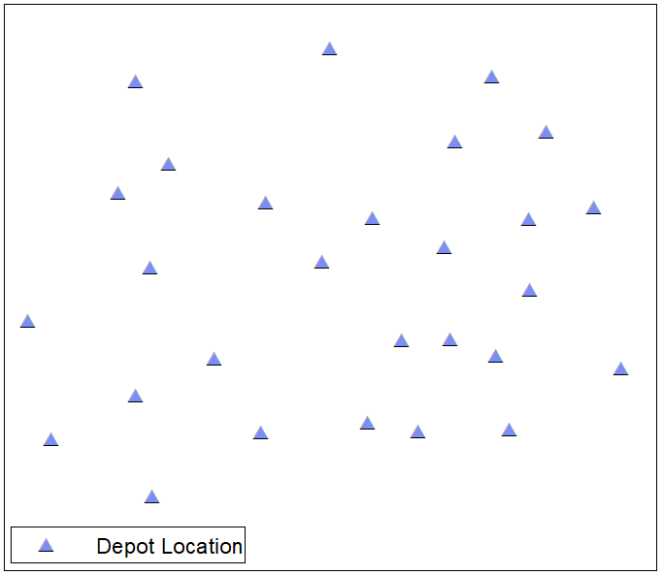


Figure 20. The layout of the depot locations

We call this a PM problem, and the goal is to find the optimal bus routes (from school to bus stops) and optimize school dismissal time. While the objective of an AM problem is to find the optimal bus routes (from bus stops to schools) and optimize school bell time. If school bell times are provided, we can use one of the proposed methods to solve the AM problem and get the corresponding bus schedule. As for the bell time optimization, if we assume that the school duration is fixed, the AM bell time can be easily obtained by subtracting the school duration from the calculated PM dismissal time. If the problem is an AM problem and the bell time is optimized, the PM dismissal time can be obtained by adding the school duration to the AM bell time.

Five test problem groups, including fourteen test problems, were derived from the collected real-world data. Table 5 shows the detailed configurations of the school, trip, depot, and school dismissal time window for each test problem. Each test group contains trips from all types of schools. Inside each test group, three cases are

considered: (1) the problem without the dismissal time optimization; (2) the problem with a small dismissal time window; (3) the problem with a larger dismissal time window. Take Case #2; for example, for each school, its dismissal time after optimization should be no earlier than 30 minutes before its original dismissal time and no later than 30 minutes after its original dismissal time. The school dismissal times after optimization are allowed to be before the current school dismissal times. If that happens, the starting time of those schools will be changed accordingly (i.e., starting earlier than before). Overall, across different test groups, different school dismissal time windows and number of depots are set to examine the models' performance. The last group considers the entire collected real-world data.

Table 5. Configurations of test problems

		No. of Schools	No. of Trips	No. of Depots	Depot Capacity*	School Dismissal Time Windows**
Group1	**Case #1**	13	94	1	-	-
	Case #2	13	94	1	-	$[ODT_i - 30\ min, ODT_i + 30\ min]$
	Case #3	13	94	1	-	$[ODT_i - 40\ min, ODT_i + 40\ min]$
Group2	**Case #4**	13	55	3	15	-
	Case #5	13	55	3	10	$[ODT_i - 30\ min, ODT_i + 30\ min]$
	Case #6	13	55	3	10	$[ODT_i - 40\ min, ODT_i + 40\ min]$
Group3	**Case #7**	21	129	7	15	-
	Case #8	21	129	7	10	$[ODT_i - 30\ min, ODT_i + 30\ min]$
	Case #9	21	129	7	10	$[ODT_i - 40\ min, ODT_i + 40\ min]$
Group4	**Case #10**	45	306	8	25	-
	Case #11	45	306	8	25	$[ODT_i - 30\ min, ODT_i + 30\ min]$
	Case #12	45	306	8	25	$[ODT_i - 40\ min, ODT_i + 40\ min]$
Group5	**Case #13**	55	678	28	25	-
	Case #14	55	678	28	20	$[ODT_i - 30\ min, ODT_i + 30\ min]$

Note: "*": "-" in depot capacity: unconstrained depot capacity; "**": "-" in school dismissal time windows: without dismissal time optimization; **ODT**: original dismissal time of School $i, i \in SCH$.

All the test problems consider the school bus scheduling problem in the PM, that is, to find the optimal bus schedules to transport students from schools to their designated bus stops. Therefore, the school bell time optimization is actually school dismissal time optimization. For those cases with dismissal time optimization, we try to find the best dismissal time for each school within the given time window.

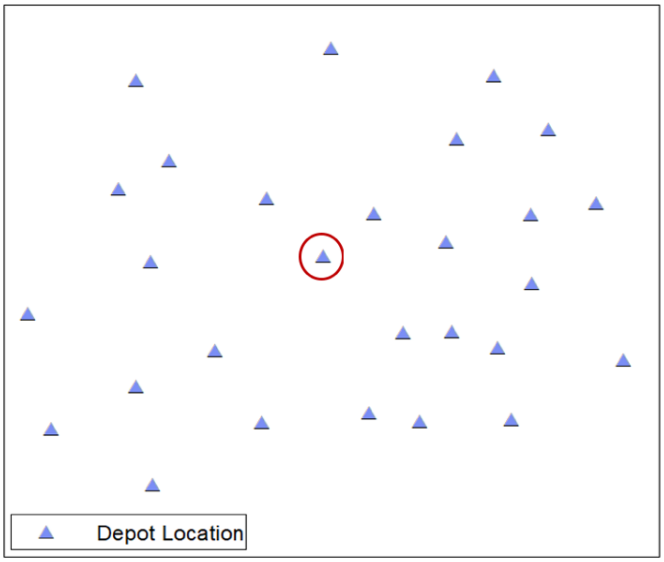


Figure 21. Depot used in Case #1 to Case #3

Case #1 to Case #3 can be grouped together. They are the special cases of the MDSBSP by reducing the total number of depots to one. For these three cases, the input data comes from the small region in the middle part of the whole area, which includes a total of 94 trips that belong to 13 different schools (i.e., seven ESs, three MSs, and three HSs) and a depot highlighted with the red circle as shown in Figure 21. As for the school dismissal time windows, Case #1 is the single-depot multi-school bus scheduling problem without school dismissal time optimization. Case #2 and Case #3

are the single-depot multi-school bus scheduling problem with school dismissal time optimization. The time window in Case #2 is smaller than that in Case #3.

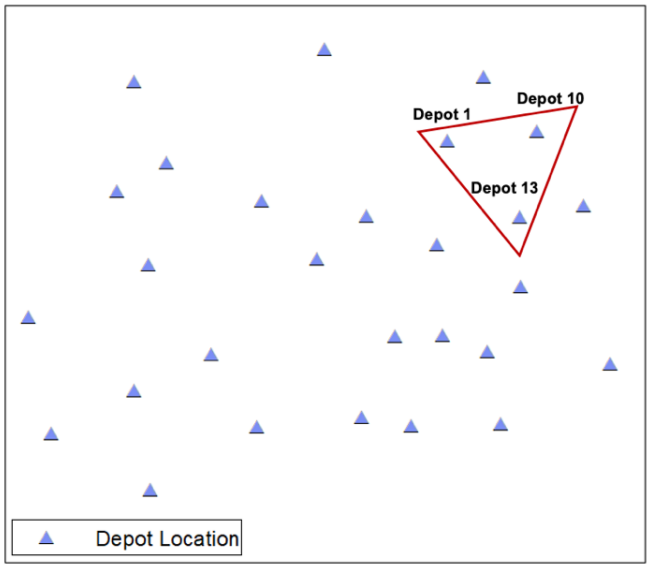


Figure 22. Depots used in Case #4 to Case #6

Case #4 to Case #6 can be grouped together as well. The trips are derived from the region in the northeast part of the whole area, as shown in Figure 22. A total of 55 trips that belong to 13 different schools (i.e., seven ESs, three MSs, and three HSs) are used as the input data for these three cases. Three depots, namely, Depot 1, Depot 10, and Depot 13, are used and are highlighted with the red triangle in Figure 22. As for the dismissal time optimization, Case #4 is the MDSBSP, but Case #5 and Case #6 are the MDSBSPTW. The time window in Case #5 is smaller than that in Case #6.

Case #7 to Case #9 is a larger size MDSBSP derived from the region in the southwest of the whole area, as shown in Figure 23. It has more trips, schools, and depots. To be specific, a total of 129 trips which belong to 21 different schools (i.e., nine ESs, six

MSs, and six HSs) and a total of seven depots (i.e., Depot 3, Depot 6, Depot 7, Depot 12, Depot 14, Depot 17, and Depot 21) are used in this test group. As for the dismissal time optimization, Case #7 is the MDSBSP, but Case #8 and Case #9 are the MDSBSPTW. The time window in Case #8 is smaller than that in Case #9.

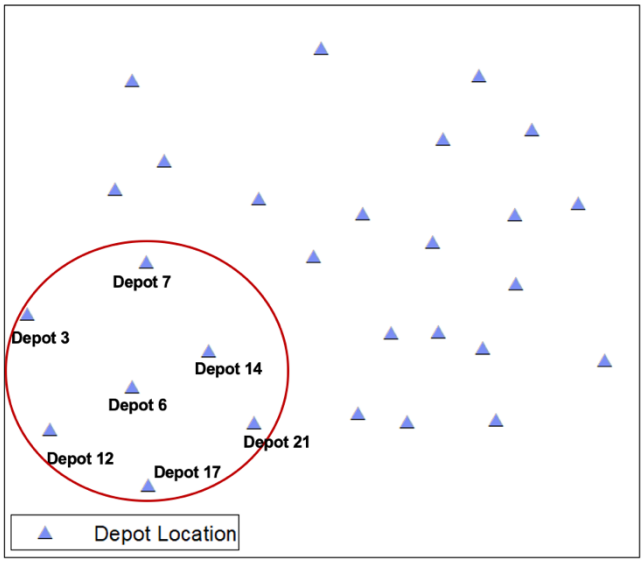


Figure 23. Depots used in Case #7 to Case #9

Case #10 to Case #12 is also a large size MDSBSP. It is derived from the region in the northeast of the whole area, as shown in Figure 24. It has a total number of 306 trips which belongs to 45 different schools (i.e., 25 ESs, 11 MSs, and 9 HSs) and a total of eight depots (i.e., Depot 1, Depot 2, Depot 10, Depot 13, Depot 20, Depot 23, Depot 24, and Depot 27) are used in this test group. As for the dismissal time optimization, Case #10 is the MDSBSP, but Case 11 and Case #12 are the MDSBSPTW. The time window in Case #11 is smaller than that in Case 12.

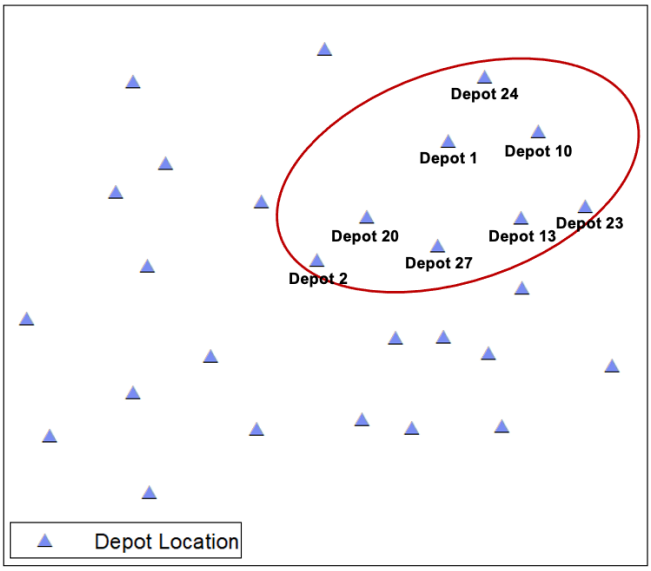


Figure 24. Depots used in Case #10 to Case #12

Case #13 and Case #14 use the entire collected data, including 678 trips from 55 schools (i.e., 31 ESs, 13 MSs, and 11 HSs) and 28 depots shown in Figure 20. Case #13 is an MDSBSP, while Case #14 is an MDSBSPTW. After optimization, each school's dismissal time should be no earlier than 30 minutes before its original dismissal time and no later than 30 minutes after its original dismissal time.

For each test problem, the MIP model, the two-phase heuristic method, and the second-assignment phase of the improved heuristic method are solved by the Gurobi solver in Python. The model running time limit is set to be 10 hours (36,000 seconds) for Case #1 to Case #12. And it is set to be 50 hours (180,000 seconds) for the largest-size test problems (i.e., Case #13 and Case #14). We run each model 10~20 times for each test problem for all proposed heuristic methods. The best solution and the average model running time are reported for each test problem. The code is written in Python 3.8 on a computer with Intel® Core™ i5-10600K processor, 4.10GHz with 8GB RAM.

6.2 Results of the MIP Model

The results of all the test problems based on the MIP model presented in Chapter 3 are shown in Table 6. Under the name of each test problem shows the configuration of the test problem: $number\ of\ schools - number\ of\ trips - number\ of\ depots - with\ or\ without\ dismissal\ time\ optimization.$ NDO means the test problem is without the bell time optimization. $30\ min$ or $40\ min$ means the test problem considers the dismissal time optimization, and the dismissal time after optimization for each school should be no earlier than 30 min (or 40 min) before its original dismissal time and no later than 30 min (or 40 min) after its original dismissal time.

Cases #1 to # 3 are the special cases of the MDSBSPs, that is, the SDSBSPs. By looking at the model running time, Case #1 used much less time to reach optimality than Case #2 and Case #3. Case #4 to #12 are all MDSBSPs belonging to three different test groups. Generally, all the test problems without dismissal time optimization can be solved to optimality regarding the size of depots, trips, and schools. However, after adding the dismissal time window constraints, small problems with a small school dismissal time window (e.g., Case #4) can still reach optimality. As the problem size (i.e., number of trips and schools) and dismissal time window get larger, it is harder to solve the problems to optimality after reaching the model running time limit. Instead, the MIP model provides unreasonable solutions with huge optimal gaps (e.g., Case #9 and #11) or even fails to find a feasible solution (i.e., Case #12).

Table 6. Results of all the test problems based on the MIP model

	RT	Optimal Gap	TOB	TD
Case #1 $\mathbf{13-94-1-}\boldsymbol{NDO}$	0.87	0.00%	60	1,458
Case #2 $\mathbf{13-94-1-30}\boldsymbol{min}$	11,069.35	0.01%	31	1,174
Case #3 $\mathbf{13-94-1-40}\boldsymbol{min}$	36,000	10.34%	29	1,094
Case #4 $\mathbf{13-55-3-}\boldsymbol{NDO}$	0.46	0.00%	36	778
Case #5 $\mathbf{13-55-3-30}\boldsymbol{min}$	36,000	9.15%	19	615
Case #6 $\mathbf{13-55-3-40}\boldsymbol{min}$	36,000	42.09%	19	576
Case #7 $\mathbf{21-129-7-}\boldsymbol{NDO}$	21.86	0.00%	81	1,374
Case #8 $\mathbf{21-129-7-30}\boldsymbol{min}$	36,000	57.13%	49	1,501
Case #9 $\mathbf{21-129-7-40}\boldsymbol{min}$	36,000	98.12%	54	2,269
Case #10 $\mathbf{45-306-8-}\boldsymbol{NDO}$	603.15	0.00%	179	5,495
Case #11 $\mathbf{45-306-8-30}\boldsymbol{min}$	36,000	87.026%	156	6,917
Case #12 $\mathbf{45-306-8-40}\boldsymbol{min}$	-	-	-	-
Case #13 $\mathbf{55-678-28-}\boldsymbol{NDO}$	48,704	0.00%	412	6,585
Case #14 $\mathbf{55-678-28-30}\boldsymbol{min}$	-	-	-	-

Note **RT**: model running time (sec); **TOB**: total number of buses; **TD**: total deadhead time between trips and between trips and depots (min).

Cases #13 and #14 are the largest test problems created based on the entire collected data, including 678 trips from 55 schools and 28 depots. Case #13, an MDSBSP, can still be solved optimally. But the model running time (i.e., 48,704 sec) is much longer than that of all the other smaller MDSBSPs. However, for Case #14, an MDSBSPTW, Gurobi fails to find a feasible solution after reaching the model running time limit.

In summary, all the test problems without the dismissal time optimization (Case #1, Case #4, Case #7, Case #10, and Case #13) can be solved to optimality based on the proposed MIP model. The larger the problem is, the more computational time is needed to reach optimality. Only one test problem (Case #2), an SDSBSPTW, was solved to optimality when considering the test problems with the dismissal time optimization. For the others, the optimal gap increases as the problem size (i.e., more trips or more schools) and the dismissal time window increase. The MIP may end up with some unreasonable solution with a huge optimal gap (e.g., Case #9 and Case #11) or even fail to find a feasible solution (i.e., Case #12 and Case #14) after reaching the model running time limit. Results indicate that Gurobi is not the best option for solving relatively large size problems with dismissal time window constraints.

6.3 Results of the Two-phase Heuristic Method

We used the two-phase heuristic method for solving all the test problems. The results for each phase of each test problem are shown in Table 7. Notice that Cases #1 to #3 are the single-depot problems, so their final solutions (i.e., the total number of buses and the total deadhead duration) can be directly derived from the first-route phase without going further into the second-assignment phase. For these three cases, the virtual depot in the first phase is set to be the actual depot. The total deadhead duration, which consists of the deadhead duration between trips and deadhead duration between trips and depots, is then calculated. For all other multi-depot test problems, the results for each phase are listed in detail, including the model running time, the optimal gap from the Gurobi solver, the total number of buses (from the first-route phase), the total

deadhead duration between trips (from the first-route phase), and the total deadhead duration between trips and depots (from the second-assignment phase).

Table 7. Results of each phase of all the test problems based on the two-phase heuristic method

Case #1	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TD**
	0.79	0.00%	60	1,458
Case #2	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TD**
	36,000	2.14%	31	1,174
Case #3	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TD**
	36,000	25.09%	29	1,090
Case #4	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	0.24	0.00%	36	110
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.01	0.00%	682	
Case #5	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	15,740.12	0.01%	19	307
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	316	
Case #6	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	36,000	26.26%	19	261
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.01	0.00%	312	
Case #7	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	2.72	0.00%	81	251
	Second-assignment Phase			

	RT	**Optimal Gap**	**TDTD**	
	0.03	0.00%	1226	
Case #8	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	36,000	26.81%	41	658
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.03	0.00%	566	
Case #9	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	36,000	56.39%	41	785
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	686	
Case #10	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	42.88	0.00%	179	581
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	5,119	
Case #11	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	36,000	51.75%	94	1,269
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	1,989	
Case #12	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	36,000	70.25%	90	1,268
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	1,871	
Case #13	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	709.82	0.00%	412	1,235
	Second-assignment Phase			

	RT	**Optimal Gap**	**TDTD**	
	0.12	0.00%	5,993	
Case #14	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	180,000	55.79%	255	2,869
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.07	0.00%	4,292	

Note **RT**: model running time (sec); **TOB**: total number of buses; **TD**: total deadhead time between trips and between trips and depots (min); **TDT**: total deadhead time between trips (min); **TDTD**: total deadhead time between trips and depots (min).

According to Table 7, the second-assignment phase of all the test problems can always be solved to optimality in a very short time (less than one second). For the first-route phase, all the test problems are first converted into a single-depot problem (i.e., SDSBSP or SDSBSPTW) when using the two-phase heuristic method. For all SDSBSPs, the two-phase heuristic method can solve them to optimality just like the MIP model but use much less computational time. For all SDSBSPTWs, it took Gurobi a long time to solve them. Gurobi provided solutions with large optimal gaps for some of them after reaching the model running time limit. But overall, the optimal gap is much smaller than that of the MIP model. For example, for Case #12 and Case #14 that are unsolvable using the MIP model, the two-phase heuristic method can still provide a feasible solution to those problems. However, for Case #3, the optimal gap from the two-phase heuristic method is larger than that from the MIP model. Gurobi failed to efficiently solve it because it has a relatively large number of trips and a large school dismissal time window though it only has one depot.

The complete solution for each test problem is presented in Table 8. The model running time is the total model running time for both phases. The total number of buses is derived from the first-route phase. And the total deadhead duration is the sum of the deadhead between trips (results from the first-route phase) and the deadhead between trips and depots (results from the second-assignment phase).

Table 8. Complete results of all the test problems based on the two-phase heuristic method

	RT	**TOB**	**TD**
Case #1 $\mathbf{13-94-1-}\boldsymbol{NDO}$	0.79	60	1,458
Case #2 $\mathbf{13-94-1-30}\boldsymbol{min}$	36,000	31	1,174
Case #3 $\mathbf{13-94-1-40}\boldsymbol{min}$	36,000	29	1,090
Case #4 $\mathbf{13-55-3-}\boldsymbol{NDO}$	0.25	36	792
Case #5 $\mathbf{13-55-3-30}\boldsymbol{min}$	15,740.14	19	623
Case #6 $\mathbf{13-55-3-40}\boldsymbol{min}$	36,000.01	19	573
Case #7 $\mathbf{21-129-7-}\boldsymbol{NDO}$	2.75	81	1,477
Case #8 $\mathbf{21-129-7-30}\boldsymbol{min}$	36,000.03	41	1,224
Case #9 $\mathbf{21-129-7-40}\boldsymbol{min}$	36,000.02	41	1,471
Case #10 $\mathbf{45-306-8-}\boldsymbol{NDO}$	42.90	179	5,700
Case #11 $\mathbf{45-306-8-30}\boldsymbol{min}$	36,000.02	94	3,258
Case #12 $\mathbf{45-306-8-40}\boldsymbol{min}$	36,000.02	90	3,139
Case #13 $\mathbf{55-678-28-}\boldsymbol{NDO}$	709.94	412	7,228
Case #14 $\mathbf{55-678-28-30}\boldsymbol{min}$	180,000.07	255	7,161

Note **RT**: model running time (sec); **TOB**: total number of buses; **TD**: total deadhead time between trips and between trips and depots (min).

Most test problems with the dismissal time optimization have reached the given model running time limit. But based on the current results, we can still find out that the total number of buses was significantly reduced after the school dismissal time optimization. The larger the school dismissal time window is, the more buses can be reduced, but a longer model running time is needed.

6.4 Results of the Improved Two-phase Heuristic Method

We also list the detailed results for each phase of each test problem based on the improved two-phase heuristic method in Table 9.

Table 9. Results of each phase of all the test problems based on the improved two-phase heuristic

Case #1	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TD**
	0.13	-	60	1,479
Case #2	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TD**
	491	-	32	1,215
Case #3	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TD**
	535.15	-	29	1,218
Case #4	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	0.05	-	36	115
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.01	0.00%	688	
Case #5	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	157.95	-	19	413
	Second-assignment Phase			

	RT	**Optimal Gap**	**TDTD**	
	0.01	0.00%	320	
Case #6	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	222.40	-	19	310
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.01	0.00%	319	
Case #7	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	0.23	-	83	233
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	1,249	
Case #8	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	1,368	-	41	998
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	631	
Case #9	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	1,610.22	-	38	1,088
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.03	0.00%	720	
Case #10	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	1.63	-	179	652
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.03	0.00%	5,114	
Case #11	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	3,913.01	-	92	2,282
	Second-assignment Phase			

	RT	**Optimal Gap**	**TDTD**	
	0.02	0.00%	1,964	
Case #12	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	6,617.25	-	84	2,328
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.01	0.00%	1,826	
Case #13	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	7.42	-	413	1,305
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.11	0.00%	5,922	
Case #14	***First-route Phase***			
	RT	**Optimal Gap**	**TOB**	**TDT**
	23,297.44	-	226	5,366
	Second-assignment Phase			
	RT	**Optimal Gap**	**TDTD**	
	0.07	0.00%	4,240	

Note **RT**: model running time (sec); **TOB**: total number of buses; **TD**: total deadhead time between trips and between trips and depots (min); **TDT**: total deadhead time between trips (min); **TDTD**: total deadhead time between trips and depots (min).

For the improved two-phase heuristic method, the first-route phase formulated as a single-depot problem is solved by the proposed SA-GDA algorithm without using Gurobi solver, so the optimal gap is not reported. According to Table 9, the proposed SA-GDA algorithm can optimally solve the test problems without dismissal time optimization except for Case #7, Case #10, and Case #13. But the optimal gap in terms of the total number of buses is only 2%, 1%, and 0.24% for those three cases, respectively. Also, the model running time of those three cases is much less than that of the previous two methods. Take Case #13, for example; it is the largest MDSBSP

created based on the entire collected data. The total number of buses and the total deadhead duration from the improved two-phase heuristic method are 0.24% and 0.9% larger than the optimal solution. But the model running time of the improved two-phase heuristic method is only 7.42 seconds, while that of the other two methods are 48,704 seconds and 709.04 seconds, respectively. Overall, the improved two-phase heuristic performs better than the other two methods.

As for the test problems with dismissal time optimization, the improved two-phase heuristic performs much better than the other two in all the test problems. It can even achieve fewer buses much quicker when the test problem involves more trips, depots, schools, and larger dismissal time windows (e.g., Case #6, #9, #11, #12, and #14). For example, Case #14 is the MDSBSPTW based on the entire dataset. The MIP model failed to solve it within the given running time limit. The two-phase heuristic method reached a solution with a 55.79% optimal gap after 50 hours. However, the improved heuristic method only used around 12% of the running time of the two-phase heuristic method but achieved a much better solution with 29 fewer buses.

When it comes to the second-assignment phase, all problems can be solved optimally in less than one second. The complete solution for each test problem based on the improved two-phase heuristic method is presented in Table 10. The model running time is the total model running time from both phases. The total number of buses is from the first-route phase. And the total deadhead duration is the sum of the deadhead between

trips (from the first-route phase) and the deadhead between trips and depots (results from the second-assignment phase).

Table 10. Complete results of all the test problems based on improved two-phase heuristic

	RT	TOB	TD
Case #1 $\mathbf{13-94-1-}\boldsymbol{NDO}$	0.13	60	1,479
Case #2 $\mathbf{13-94-1-30}\boldsymbol{min}$	491	32	1,215
Case #3 $\mathbf{13-94-1-40}\boldsymbol{min}$	535.15	29	1,218
Case #4 $\mathbf{13-55-3-}\boldsymbol{NDO}$	0.06	36	803
Case #5 $\mathbf{13-55-3-30}\boldsymbol{min}$	157.96	19	733
Case #6 $\mathbf{13-55-3-40}\boldsymbol{min}$	222.41	19	629
Case #7 $\mathbf{21-129-7-}\boldsymbol{NDO}$	0.25	83	1,482
Case #8 $\mathbf{21-129-7-30}\boldsymbol{min}$	1,368.02	41	1,629
Case #9 $\mathbf{21-129-7-40}\boldsymbol{min}$	1,610.25	38	1,808
Case #10 $\mathbf{45-306-8-}\boldsymbol{NDO}$	1.66	179	5,766
Case #11 $\mathbf{45-306-8-30}\boldsymbol{min}$	3,913.03	92	4,246
Case #12 $\mathbf{45-306-8-40}\boldsymbol{min}$	6,617.26	84	4,154
Case #13 $\mathbf{55-678-28-}\boldsymbol{NDO}$	7.42	413	7,227
Case #14 $\mathbf{55-678-28-30}\boldsymbol{min}$	23,297.51	226	9,606

Note **RT**: model running time (sec); **TOB**: total number of buses; **TD**: total deadhead time between trips and between trips and depots (min).

According to Table 10, the improved two-phase heuristic method can efficiently provide better results (i.e., fewer buses) for almost all the test problems than the

previous two methods. Besides, like previous findings, having a larger school dismissal time window could reduce more buses but result in a longer model running time.

6.5 Results of the TS-ACO Method

The results of the proposed TS-ACO method are presented in Table 11.

Table 11. Results of all the test problems based on the TS-ACO method

	RT	**TOB**	**TD**
Case #1 $\mathbf{13-94-1-}\boldsymbol{NDO}$	1.20	60	1,459
Case #2 $\mathbf{13-94-1-30}\boldsymbol{min}$	1,512	32	1,198
Case #3 $\mathbf{13-94-1-40}\boldsymbol{min}$	1,401	30	1,153
Case #4 $\mathbf{13-55-3-}\boldsymbol{NDO}$	0.65	36	779
Case #5 $\mathbf{13-55-3-30}\boldsymbol{min}$	284	20	631
Case #6 $\mathbf{13-55-3-40}\boldsymbol{min}$	328	19	594
Case #7 $\mathbf{21-129-7-}\boldsymbol{NDO}$	3.82	81	1,399
Case #8 $\mathbf{21-129-7-30}\boldsymbol{min}$	3,395	43	1,453
Case #9 $\mathbf{21-129-7-40}\boldsymbol{min}$	6,000	38	1,701
Case #10 $\mathbf{45-306-8-}\boldsymbol{NDO}$	30.15	179	5,677
Case #11 $\mathbf{45-306-8-30}\boldsymbol{min}$	21,437.16	94	4,050
Case #12 $\mathbf{45-306-8-40}\boldsymbol{min}$	22,966.52	85	4,096
Case #13 $\mathbf{55-678-28-}\boldsymbol{NDO}$	326.45	412	6,991
Case #14 $\mathbf{55-678-28-30}\boldsymbol{min}$	120,628	235	9,539

Note **RT**: model running time (sec); **TOB**: total number of buses; **TD**: total deadhead time between trips and between trips and depots (min).

For the test problems without dismissal time optimization, the running time of the ACO model is around the same as the MIP model and the two-phase heuristic method when the problem size is relatively small (i.e., Case #1 and Case #4). When the problem size gets larger, the running time of the ACO model is less than the other two methods. However, the running time of the ACO model is longer than that of the improved two-phase heuristic method in all cases. As for the solution, the ACO model, like the MIP model and the two-phase heuristic method, can reach the optimal number of buses for all test problems without dismissal time, outperforming the improved two-phase heuristic method. Besides, the ACO model can obtain a shorter deadhead duration than the two-phase heuristic method and the improved two-phase heuristic method but a slightly longer deadhead duration than that of the MIP model.

For the test problems with dismissal time optimization, the TS-ACO method consistently outperforms the MIP model in terms of the model running time and the solution quality. Though for the SDSBSPTWs (Case #2 and Case #3), the total number of buses from TS-ACO has one more bus than that from the MIP model, considering the computational time, the TS-ACO method is still better than the MIP model. Compared to the two-phase heuristic method, the TS-ACO performs better as the problem size gets larger or the dismissal time window is larger (e.g., Case #9, #11, #12, and #14). However, the improved two-phase heuristic method can get even better results for the large-size MDSBSPTWs than the TS-ACO method in a much shorter time (e.g., Case #12 and #14). Take Case #14, for example; it is the largest MDSBSPTW based on the entire collected data. The MIP model failed to find a

solution after reaching the running time limit. The TS-ACO used less time than the two-phase heuristic method and provided a better solution with 20 fewer buses than the two-phase heuristic method. The improved two-phase heuristic method only used around 19.3% of the running time of the TS-ACO method but achieved a solution that has nine fewer buses than the TS-ACO method. Therefore, the improved two-phase heuristic method is the most powerful among all proposed methods.

6.6 Overall Results and Comparison

Table 12 summarizes the results of all the test problems based on the four methods proposed in this study regarding the model running time, optimal gap, the total number of buses, and total deadhead duration. As for the model running time for the two-phase heuristic method and the improved two-phase heuristic method, first, the total model running time is listed. Then the running times of both phases are listed in the bracket in which the first element is the model running time of the first-route phase, and the second element is the model running time of the second-assignment phase. For the single-depot problems (Case #1 to Case #3), only the first-route phase is used, so the model running time column only lists the model running time of the first phase. It works the same for the total deadhead duration column. The total deadhead duration is first shown. Then, inside the next bracket, the first element is the deadhead duration between trips obtained from the first-route phase. The second element is the deadhead duration between trips and depots from the second-assignment phase.

Table 12. Summary of the results of all the test problems based on all the proposed methods

	Method	RT	OG	TOB	TD
Case #1 $\mathbf{13-94-1-}\boldsymbol{NDO}$	MIP model	0.87	0.00%	60	1,458
	Two-phase	0.79	0.00%	60	1,458
	Improved two-phase	0.13	-	60	1,479
	TS-ACO	1.20	-	60	1,459
Case #2 $\mathbf{13-94-1-30}\boldsymbol{min}$	MIP model	11,069.35	0.01%	31	1,174
	Two-phase	36,000	2.14%	31	1,174
	Improved two-phase	491	-	32	1,215
	TS-ACO	1,512	-	32	1,198
Case #3 $\mathbf{13-94-1-40}\boldsymbol{min}$	MIP model	36,000	10.34%	29	1,094
	Two-phase	36,000	25.09%	29	1,090
	Improved two-phase	535.15	-	29	1,218
	TS-ACO	1,401	-	30	1,153
Case #4	MIP model	0.46	0.00%	36	778

$\mathbf{13-55-3-}\boldsymbol{NDO}$	Two-phase	0.25 (0.24+0.01)	0.00%	36	792 (110+682)
	Improved two-phase	0.06 (0.05+0.01)	-	36	803 (115+688)
	TS-ACO	0.65	-	36	779
Case #5 $\mathbf{13-55-3-30}\boldsymbol{min}$	MIP model	36,000	9.15%	19	615
	Two-phase	15,740.14 (15,740.12+0.02)	0.01%	19	623 (307+316)
	Improved two-phase	157.96 (157.95+0.010)	-	19	733 (413+320)
	TS-ACO	284	-	20	631
Case #6 $\mathbf{13-55-3-40}\boldsymbol{min}$	MIP model	36,000	42.09%	19	576
	Two-phase	36,000.01 (36,000+0.01)	26.26%	19	573 (261+312)
	Improved two-phase	222.41 (222.40+0.01)	-	19	629 (310+319)
	TS-ACO	328	-	19	594
Case #7 $\mathbf{21-129-7-}\boldsymbol{NDO}$	MIP model	21.86	0.00%	81	1,374
	Two-phase	2.75 (2.72+0.03)	0.00%	81	1,477 (251+1,226)
	Improved two-phase	0.25 (0.23+0.02)	-	83	1,482 (233+1,249)
	TS-ACO	3.82	-	81	1,399

Case #8 $\mathbf{21-129-7-30}\boldsymbol{min}$	MIP model	36,000	57.13%	49	1,501
	Two-phase	36,000.03 (36,000+0.03)	26.81%	41	1,224 (658+566)
	Improved two-phase	1,368.02 (1,368+0.02)	-	41	1,629 (998+631)
	TS-ACO	3,395	-	43	1,453
Case #9 $\mathbf{21-129-7-40}\boldsymbol{min}$	MIP model	36,000	98.12%	54	2,269
	Two-phase	36,000.02 (36,000+0.02)	56.39%	41	1,471 (785+686)
	Improved two-phase	1,610.25 (1,610.22+0.03)	-	38	1,808 (1,088+720)
	TS-ACO	6,000	-	38	1,701
Case #10 $\mathbf{45-306-8-}\boldsymbol{NDO}$	MIP model	603.15	0.00%	179	5,495
	Two-phase	42.90 (42.88+0.02)	0.00%	179	5,700 (581+5,119)
	Improved two-phase	1.66 (1.63+0.03)	-	179	5,766 (652+5,114)
	TS-ACO	30.15	-	179	5,677
Case #11 $\mathbf{45-306-8-30}\boldsymbol{min}$	MIP model	36,000	87.026%	156	6,917
	Two-phase	36,000.02 (36,000+0.02)	51.75%	94	3,258 (1,269+1,989)
	Improved two-phase	3,913.03 (3,913.01+0.02)	-	92	4,246 (2,282+1,964)

	TS-ACO	21,437.16	-	94	4,050
Case #12 $\mathbf{45-306-8-40}\boldsymbol{min}$	MIP model	-	-	-	-
	Two-phase	36,000.02 (36,000+0.02)	70.25%	90	3,139 (1,268+1,871)
	Improved two-phase	6,617.26 (6,617.25+0.01)	-	84	4,154 (2,328+1,826)
	TS-ACO	22,966.52	-	85	4,096
Case #13 $\mathbf{55-678-28-}\boldsymbol{NDO}$	MIP model	48,704	0.00%	412	6,585
	Two-phase	709.94 (709.82+0.12)	0.00%	412	7,228 (1,235+5,993)
	Improved two-phase	7.42 (7.31+0.11)	-	413	7,227 (1,305+5,922)
	TS-ACO	326.45	-	412	6,991
Case #14 $\mathbf{55-678-28-30}\boldsymbol{min}$	MIP model	-	-	-	-
	Two-phase	180,000.07 (180,000+0.07)	55.79%	255	7,161 (2,869+4,292)
	Improved two-phase	23,297.51 (23,297.44+0.07)	-	226	9,606 (5,366+4,240)
	TS-ACO	120,628	-	235	9,539

Note: **RT**: model running time (sec); **TOB**: total number of buses; **TD.**: total deadhead time between trips and between trips and depots (min)

The optimal gap is obtained directly from the Gurobi solver. Table 12 shows the optimal gap for the MIP model and the first-route phase of the two-phase heuristic method, which is also a MIP model. The first-route phase of the improved heuristic method and the TS-ACO is not solved with Gurobi, so the optimal gap is not presented. Besides, we found that the second-assignment phase of the two-phase heuristic method and the improved two-phase heuristic method can always reach optimality in a very short time. So, the optimal gaps of that phase for both methods are omitted.

For the test problems without the dismissal time optimization, the MIP model, the two-phase heuristic method, and the TS-ACO method can always achieve optimality. The improved two-phase heuristic method can reach optimality in most cases, but sometimes it may reach a solution that is only slightly worse than the optimal solution (i.e., Case #7 and Case #13) using much less time than the other three methods.

For the test problems with the dismissal time optimization, the MIP model performs well when the problem size (e.g., a smaller depot size, trip size, or school size) and the dismissal time window are small. When the problem size and the dismissal time window get larger, a solution with a relatively large optimal gap will be returned after reaching the running time limit. The same thing happens for the first-route phase of the two-phase heuristic method, which is an SDSBSPTW. It seems that Gurobi cannot efficiently solve the large-size test problems with the dismissal time window. But overall, the two-phase heuristic method performs better than the MIP model in terms of the model running time and optimal gap.

The TS-ACO method works better as the problem size gets larger or has a larger dismissal time window than the MIP model and the two-phase heuristic method in terms of the model running time and the solution quality. The improved two-phase heuristic method performs the best among all the proposed methods in all test problems. It can even achieve fewer buses much quicker when the problem involves more trips, depots, schools, and a larger dismissal time window. Generally, for both SDSBSPTW and MDSBSPTW, no matter what method to use, having a larger school dismissal time window could reduce more buses but make the problem more complicated and thus result in a longer model running time.

Figure 25 shows the actual bus savings and the percentage savings in the number of buses from all the proposed methods for all test problems with dismissal time optimization. For each test problem, the exact bus saving is calculated as the total number of buses without dismissal time optimization minus the total number of buses after the dismissal time optimization. The percentage of saving in the number of buses is the exact bus saving divided by the total number of buses without dismissal time optimization. In Figure 25, the $(-30,30)$ or $(-40,40)$ right after the test problem's name indicate the size of the dismissal time window for each school i in that test problem. For example, the dismissal time window for each school i in Case #2 is $[ODT_i - 30\ min, ODT_i + 30\ min]$. The problem size is also described in the bottom as $number\ of\ schools - number\ of\ trips - number\ of\ depots$. Because the MIP model failed to solve Case #12 and Case #14, we didn't plot the bus savings from the MIP model for those two test problems in Figure 25.

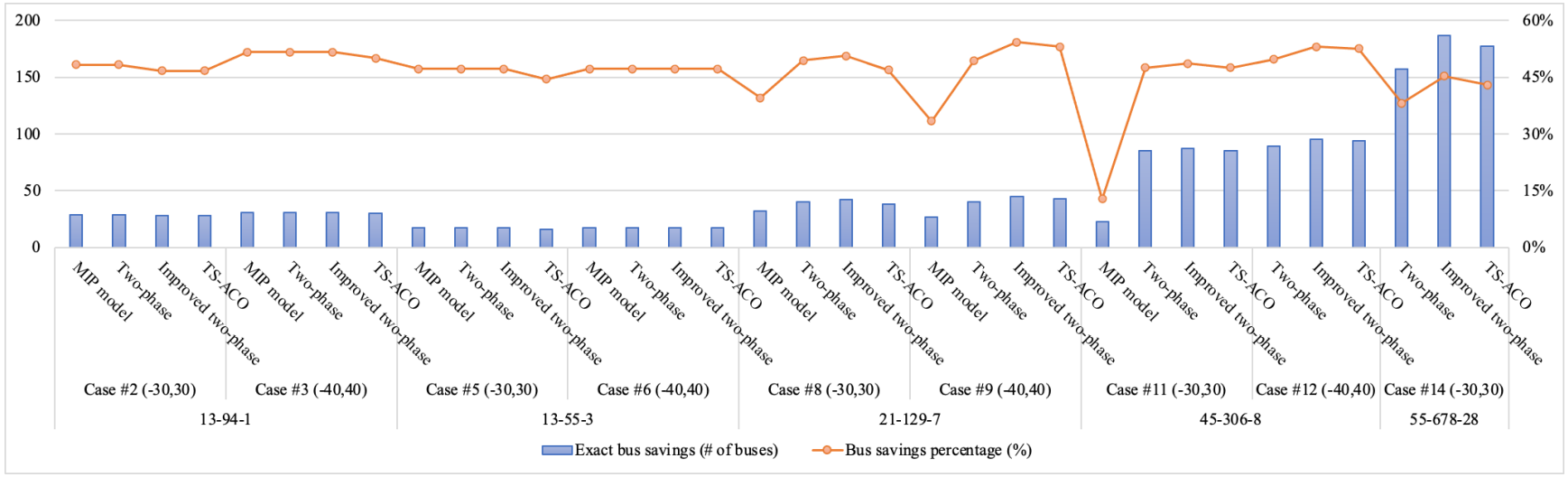
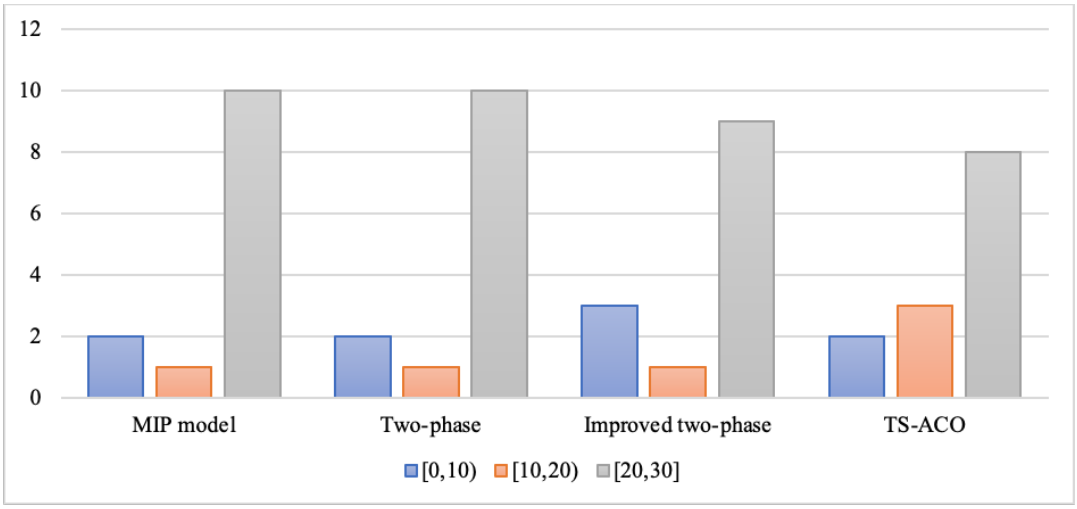


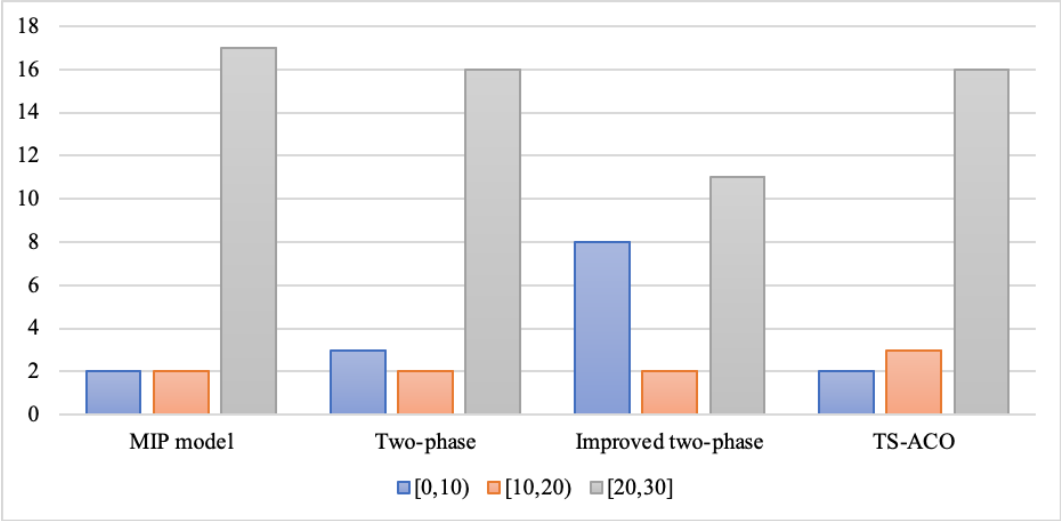
Figure 25. Savings in the number of buses after the dismissal time optimization

According to Figure 25, the average bus saving percentage is around 45%, and the highest is 54.22% (Case #9 based on the improved two-phase heuristic method). In comparison, the lowest bus saving percentage is 12.85% (Case #11 based on the MIP model). Under the same problem size setting, the exact bus saving and the bus saving percentage are larger when the dismissal time window is larger. It indicates that the larger dismissal time window results in more bus savings. When the problem size is small (Case #2 to Case #6), the models' performances regarding the bus savings are around the same for all proposed methods. But as the problem size gets larger (Case #8 to Case #14), the MIP model tends to be less powerful and can only reduce a limited number of buses or even fail to solve the problem. The improved two-phase heuristic method performs the best, followed by the TS-ACO method.

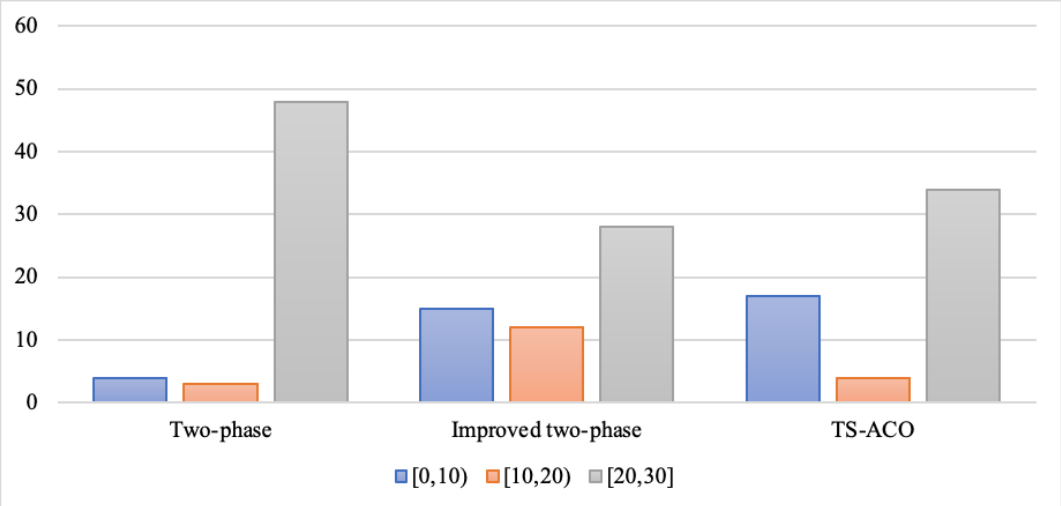
For both SDSBPTWs and MDSBSPTWs, the dismissal times for most schools are changed after the school dismissal optimization. And the dismissal times of those schools after the optimization are more likely to reach (or near) the boundary values of the dismissal time window no matter which method is used. Figure 26 shows how the school dismissal time changed after the optimization for one SDSBSPTW (i.e., Case #2), one MDSBSPTW (i.e., Case #8), and another larger size MDSBSPTW that uses all the collected data (i.e., Case #14). For those three test problems, the dismissal time window settings are the same; that is, for each school i, the dismissal time window is $[ODT_i - 30\ min, ODT_i + 30\ min]$.



(a) Case #2



(b) Case #8



(c) Case #14

Figure 26. How dismissal time changed after the optimization

We can get each school's dismissal time after the optimization based on each proposed method. So for each of those three test problems based on each proposed method, we calculated the absolute difference between the school dismissal time before and after the school dismissal optimization for each school i, which is $abs(ODT_i - newDT_i)$, where ODT_i and $newDT_i$ are the original dismissal time and new dismissal time of school i after the optimization, respectively. The absolute difference is classified into three categories: $[0,10)$, $[10,20)$, $[20,30]$. Suppose the absolute difference between the school dismissal time before and after the optimization for a particular school falls in the last category $[20,30]$. In that case, it indicates that that school has experienced a significant change in its dismissal time, and its new dismissal time is close to the boundary values of its dismissal time window.

As shown in Figure 26, no matter what method is used, most schools have their dismissal time changed to reach (or get near) the boundary values of the dismissal time window (falling in the last category). To better illustrate this, we also present the differences between the school dismissal times before and after the optimization for each school (i.e., the original dismissal time minus the dismissal time after the optimization) for Case #2 based on each proposed method in Figure 27. The figure shows that only one school's dismissal time (i.e., Sch-3) was unchanged after the optimization in Case #2 based on the improved heuristic method. For those schools whose dismissal times are changed, if we only look at the results from the MIP model, we can find that seven schools have their dismissal time adjusted to be $ODT - 30\ min$ or $ODT + 30\ min$. Two schools (i.e., Sch-9 and Sch-10) have their new dismissal

times very close to the boundary values of the dismissal time window. The results from the other methods show the same pattern.

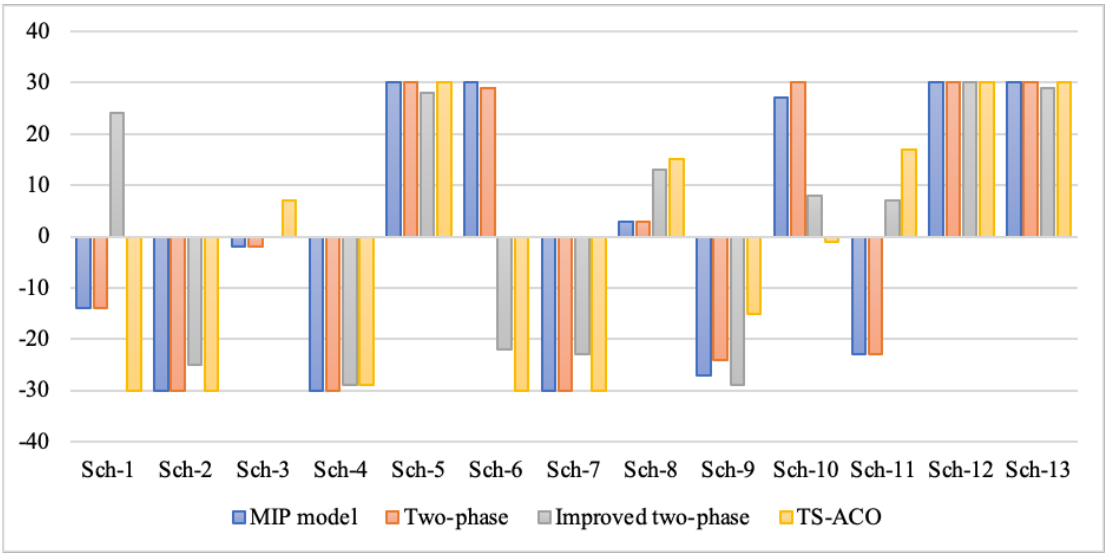
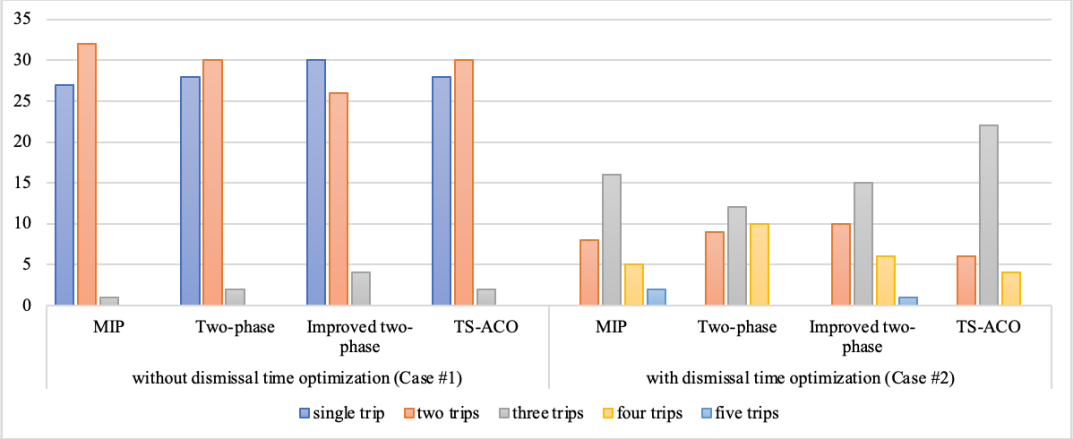
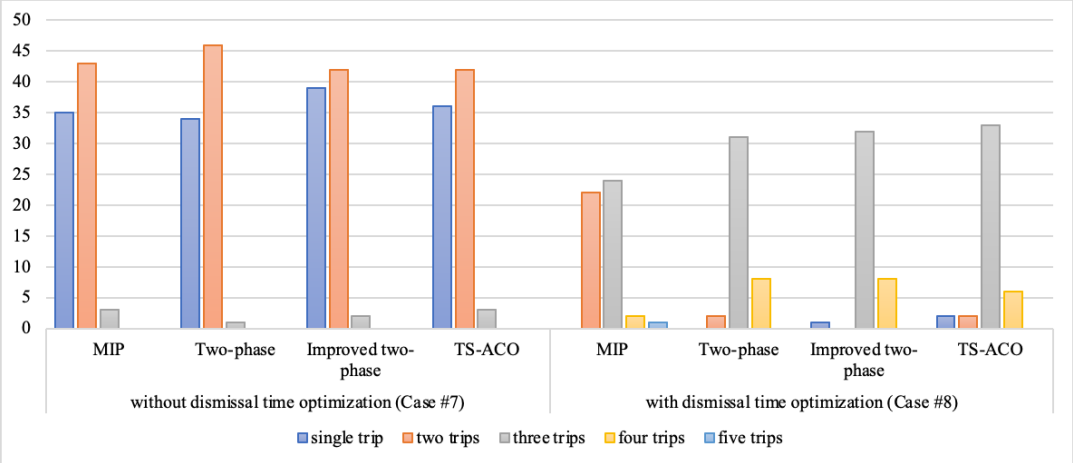


Figure 27. Differences between the school dismissal times before and after the optimization for Case #2 based on all proposed methods

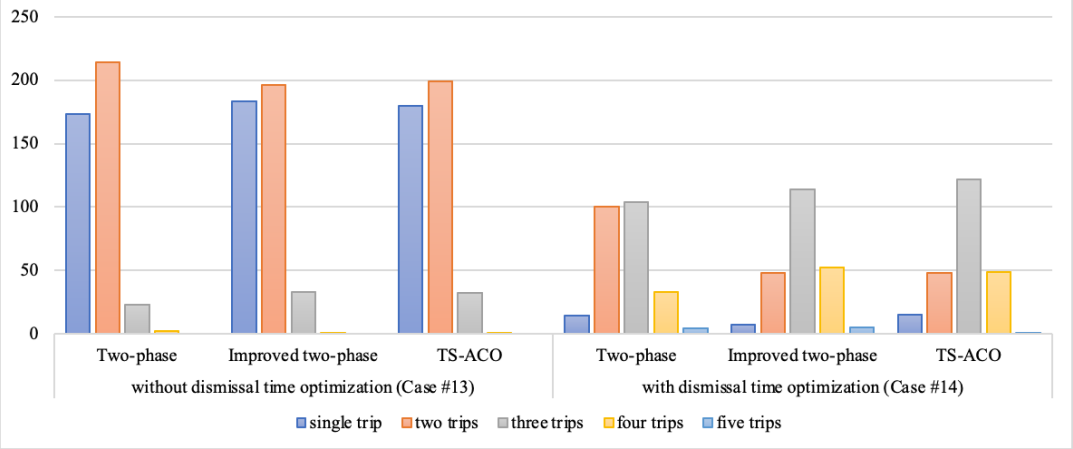
Since the original school dismissal times are quite close, changing the dismissal time to the boundary values of the school dismissal time window after the optimization can make more trips compatible. Thus, more trips can be linked together on a single bus route. Figure 28 shows the number of trips services per route with and without the optimization for one single-depot problem (Case #1 and Case #2), one multi-depot problem (Case #7 and Case #8), and the largest multi-depot problem based on all the collected data (Case #13 and Case #14). We can find that almost all the single-trip bus routes are eliminated, and bus routes tend to become longer after the school dismissal time optimization, which helps reduce the total number of buses.



(a) Case #1 and Case #2



(b) Case #7 and Case #8



(c) Case #13 and Case #14

Figure 28. Number of trips services per route with or without dismissal time optimization

According to Figure 28, the test problems without the school dismissal time optimization tend to have many single-trip bus routes. This is because the original

school dismissal times are quite close, making it impossible for buses to go to different schools after finishing the trips of the current school. After the dismissal time optimization, single-trip bus routes can be significantly reduced or even eliminated. Also, bus routes after the dismissal time optimization tend to become longer. Therefore, more trips become compatible as each school can have its own dismissal time after the optimization. As more trips can be served on one bus, the bus route can become longer, and thus the total number of buses can be reduced.

6.7 Sensitivity Analysis

6.7.1 Coefficients of the Objective Function

The objective function of the proposed MIP model in Chapter 3 and the objective function of the MIP model of the first-route phase of the two-phase heuristic method in Chapter 4 can also be a cost function based on the coefficient f_c and R_c. Therefore, we chose a couple of test problems and changed their objective function into the cost function to see how it impacts the solution. The test problems that we chose include Case #1 and Case #2, Case #10 and Case #11, and Case #13 and Case #14. Case #1 is an SDSBSP. Case #10 and Case #13 are the MDSBSPs. Case #2 is an SDSBSPTW. Case #11 and Case #14 are the MDSBSPTWs. For Case #2, Case #11, and Case #14, the dismissal time window for each school i is $[ODT_i - 30\ min, ODT_i + 30\ min]$. The results from all proposed methods for those test problems are shown in Table 13.

Table 13. Results of some test problems using cost function as the objective function

	Method	RT	OG	TOB	TD
Case #1 $\mathbf{13-94-1-}\boldsymbol{NDO}$	MIP model	0.93	0.00%	60	1,458
	Two-phase	0.75	0.00%	60	1,458
	Improved two-phase	0.17	-	60	1,479
	TS-ACO	1.77	-	60	1,458.4
Case #2 $\mathbf{13-94-1-30}\boldsymbol{min}$	MIP model	7,907.36	0.007%	31	1,174
	Two-phase	23,566.47	0.006%	31	1,174
	Improved two-phase	498.69	-	32	1,210.93
	TS-ACO	1,068.03	-	32	1,186.93
Case #10 $\mathbf{45-306-8-}\boldsymbol{NDO}$	MIP model	635.76	0.00%	179	5,495
	Two-phase	43.37 (43.35+0.02)	0.00%	179	5,699.57 (580+5119.57)
	Improved two-phase	1.62 (1.60+0.02)	-	179	5,766 (652+5,114)
	TS-ACO	82.44	-	179	5,709.73
Case #11 $\mathbf{45-306-8-30}\boldsymbol{min}$	MIP model	36,000	86.45%	169	7,877
	Two-phase	36,000.02 (36,000+0.02)	48.8774%	95	3,251.27 (1,237+2,014.27)

	Improved two-phase	3,815.01 (3,815+0.01)	-	93	4,237 (2,214+2,023)
	TS-ACO	22,940.64	-	94	4,016.92
Case #13 $\mathbf{55-678-28-}\boldsymbol{NDO}$	MIP model	53,450	0.0031%	412	6,,591
	Two-phase	1,284.12 (1,284+0.12)	0.00%	412	7228 (1,235+5,993)
	Improved two-phase	8.66 (8.54+0.12)	-	413	7,227 (1,305+5,922)
	TS-ACO	958.83	-	412	6,947.933
Case #14 $\mathbf{55-678-28-}\boldsymbol{30min}$	MIP model	-	-	-	-
	Two-phase	184,466.07 (184,466+0.07)	50.04%	245	6,,509.92 (2637+3,872.92)
	Improved two-phase	22,373.81 (22,373.75+0.06)	-	226	9,708 (5,377+4,331)
	TS-ACO	124,090	-	236	8,819.367

Note: **RT**: model running time (sec); **TOB**: total number of buses; **TD.**: total deadhead time between trips and between trips and depots (min)

For the test problems without the dismissal time optimization, the results in Table 13 are almost the same (or just slightly different in the total deadhead duration) compared to the results shown in Table 12. But for the test problems with the dismissal time optimization, we may have different results based on different forms of the objective function. For Case #2, which is an SDSBSPTW, the MIP model and the two-phase heuristic method reached the same optimal solution regardless of the form of the objective function. However, when using the improved heuristic method and the TS-ACO method, the number of buses is the same, but the deadhead duration is slightly smaller when the objective function is a cost function.

Case #11 is an MDSBSPTW. When solving it using the MIP model, the two-phase heuristic method, or the improved two-phase heuristic method, obtained a larger total number of buses when the objective function is a cost function. But the TS-ACO reached the same total number of buses. Overall, the total number of buses is larger when the objective function is a cost function. It is an expected outcome because the coefficient of the "total number of buses" term is much larger in the previous objective function (i.e., M_b) than that in the cost function form of the objective function (i.e., f_c), hence resulting in more reduction in the total number of buses.

However, the large weight (M_b) may not always work well, especially when solving very large-size MDSBSPTWs problems using the MIP model. Take Case #14, for example; it is the largest MDSBSPTW that uses all the collected data. However, the total number of buses from the MIP model in which the objective function is the cost

function is ten fewer buses than that from the MIP model that prioritizes the "total number of buses" term. The results from the two-phase heuristic method and the improved two-phase method are almost the same regardless of the forms of the objective function for Case #14. When using the TS-ACO method, the model with the cost function as the objective function has one more bus than the model that prioritizes the "total number of buses" term for Case #14.

6.7.2 Dismissal Time Window

By comparing the solutions to the test problem with and without dismissal time optimization, we can find that the total number of buses can be significantly reduced after incorporating the dismissal time optimization into the school bus scheduling problem. Besides, the improved two-phase heuristic method performs the best among all the proposed methods. Therefore, we conduct a sensitivity analysis on the solution of the improved two-phase heuristic method under different sizes of the dismissal time window based on the entire collected data (e.g., 55 schools and 678 trips). Four different sizes of dismissal time windows are proposed. That is, for each school i, we consider $[ODT_i - 10\ min, ODT_i + 10\ min]$, $[ODT_i - 20\ min, ODT_i + 20\ min]$, $[ODT_i - 30\ min, ODT_i + 30\ min]$, and $[ODT_i - 40\ min, ODT_i + 40\ min]$. When the dismissal time window for each school i is $[ODT_i - 30\ min, ODT_i + 30\ min]$, it is Case #14. After running the improved two-phase heuristic method under each dismissal time window scenario, we calculate the actual bus savings and the percentage of savings in the number of buses for each scenario. For each test problem, the exact bus saving is calculated as the total number of buses without dismissal time optimization minus the total number of buses after the dismissal time optimization. The percentage

of saving in the number of buses is the exact bus saving divided by the total number of buses without dismissal time optimization. The results are shown in Figure 29.

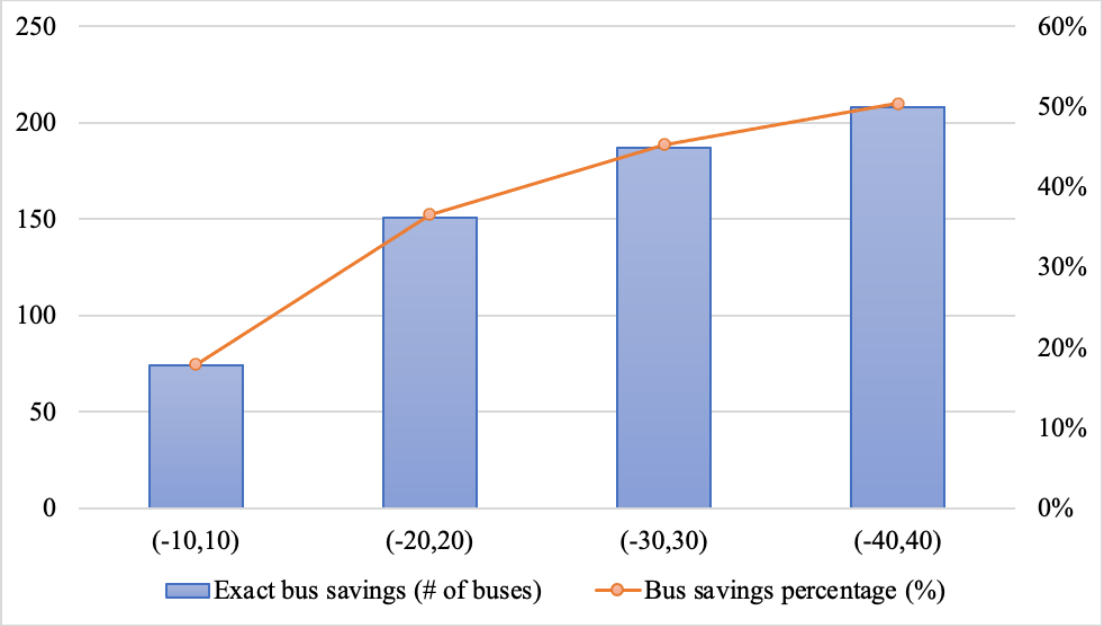


Figure 29. Savings in the number of buses over the different dismissal time windows

According to Figure 29, as the size of the dismissal time window increases, more buses can be saved. More trips become compatible after the dismissal time optimization so that more trips can be linked together on a single bus route. Bus routes tend to become longer, hence helping to reduce the total number of buses.

6.8 Summary

The performance of the proposed MIP model, the two-phase heuristic method, the improved two-phase heuristic method, and the TS-ACO method were tested on fourteen test problems in this chapter. All the test problems were derived from the real-world data collected from a public school district in Maryland. Those problems have

different school sizes, trip sizes, depot sizes, and two types of school dismissal time windows for testing the performance of the proposed methods. First, the collected data and how those test problems were generated were described in detail. Then, all the proposed methods were examined on all the test problems. The results from all four proposed methods were presented, compared, and explained. Finally, we did the sensitivity analysis on two forms of the objective functions and several different sizes of the dismissal time window. The results were compared and analyzed.

Chapter 7: Conclusions and Future Work

7.1 Summary and Conclusions

The school bus scheduling problem is an important real-world transportation problem that affects many families daily. The school districts' main issue is transporting all the students between homes and schools safely and promptly. In such problems, even a small improvement in the daily operation could lead to significant potential financial benefits. Therefore, this study aimed to solve the school bus scheduling problem. We first added some real-world settings such as multi-depot and multi-school configurations into the basic school bus scheduling problem for helping the school district make better decisions. We also incorporated the school bell time optimization into the school bus scheduling problem. Changing some schools' bell times can make the bus schedule more efficient as many buses can be reduced after the school bell time optimization. As a result, school districts can achieve significant savings.

Therefore, for helping school districts with their decision-making, this study aimed to solve the multi-depot multi-school bus scheduling problem with bell time optimization (MDSBSPTW). We proposed four different methods for solving the MDSBSPTWs. The goal is to provide the optimal bell time for each school and the corresponding best bus schedule for the school district. The four proposed methods include a Mixed-Integer Programming (MIP) model, a two-phase heuristic method, an improved two-phase heuristic method, and a TS-ACO method. The performances of all the proposed methods were tested on fourteen test problems with different characteristics. All the

test problems were derived from the real-world data collected from a public school district in Maryland. The results were compared and analyzed.

First, the MDSBSPTW was formulated as a MIP model. It is an integrated model that simultaneously optimizes the school bell times and the school bus schedule, considering the multi-depot and multi-school settings. It can solve single-depot problems if we only have one depot. It can also handle school scheduling problems solely without the school bell time optimization if fixing the bell time of each school to a specific timestamp. The objective function is to minimize the total number of buses and the total deadhead time, which is the time without students on board. We have two forms of the objective function. One adds a very large coefficient for the total number of buses term for prioritizing minimizing the total number of buses which is the major contributing factor to costs. Another one is a cost function that minimizes the total operation cost per day. Therefore, the proposed MIP model is a great tool that gives the school district flexibility for trying out different plans.

The proposed MIP model is an exact method for solving MDSBSPTWs. However, it may not be powerful enough for solving relatively large-size problems. To address that, we proposed our second method, which is the two-phase heuristic method. It solves the MDSBSPTWs through two phases sequentially. By introducing a virtual depot, we first ignore all the depot information and convert the MDSBSPTW into a single-depot multi-school bus scheduling problem with bell time optimization (SDSBSPTW) in the first-route phase. All buses should start and end at the virtual depot. The first-route

phase is formulated as a MIP model, and the purpose is to come up with the best way for connecting trips to construct bus routes. The objective function minimizes the total number of buses and the deadhead duration only between trips. If we only have one depot in the system, the virtual depot is the actual depot, and the objective function is changed to minimize the total deadhead duration, including the deadhead duration between trips and the deadhead duration between trips and depots. Like the previous MIP model, the objective function can be a cost function or prioritize either term using a large weight. The proposed MIP model for the first-route phase can also solve the problem without the bell time optimization. Then, the bus routes from the first phase are the input for the second-assignment phase. The goal is to assign each bus to the best depot that minimizes the total deadhead duration between trips and depots. It is an assignment problem formulated as an integer programming model.

The above two methods were fully implemented using the Gurobi solver in Python. They all perform well and provide optimal solutions for the test problems without the dismissal time optimization (including both SDSBSPTWs and MDSBSPTWs). However, for the test problems with the dismissal time optimization, the MIP model can only solve the ones with a smaller number of depots, trips, or schools or a smaller dismissal time window. As the problem size and the dismissal time window get larger, it will return a solution with a relatively large optimal gap or even can't find a feasible solution after reaching the running time limit. The same thing happens for the first-route phase of the two-phase heuristic method. But since the MIP model of the first-route phase is designed for solving the SDSBSPTW, it has fewer variables and

constraints than the previous MIP model for solving the MDSBSPTW. Thus, the two-phase heuristic method performs better than the MIP model regarding the model running time and optimal gap. The second-assignment phase can always be solved optimally in less than one second. Overall, Gurobi cannot efficiently solve the large-size multi-school bus scheduling problem with the dismissal time optimization regardless of the number of depots in the system.

We proposed a hybrid heuristic method that uses a local search to speed up the solution searching process to solve the first-route phase more efficiently. It is a Simulated Annealing-based Greedy Algorithm (SA-GDA) method and replaces the MIP model in the first-route phase. Keeping the second-assignment phase unchanged, we have our third method, the improved two-phase heuristic method. The SA-GDA method is proposed to solve the SDSBSPTWs (or SDSBSP) in the first-route phase. The simulated annealing algorithm is used to examine different school bell time plans. The proposed greedy algorithm is embedded into the framework of the simulated annealing algorithm to find the best bus schedule under each school bell time plan. Since we are only interested in the trip connections in the first-route phase, the best bus schedule under a given bell time plan is the one that minimizes the total number of buses and the deadhead duration between trips. The SA-GDA method compares different bus schedules and returns the best and its corresponding school bell time plan. Then, the buses are assigned to different depots to minimize the deadhead duration between trips and depots based on the assignment model proposed in the second-assignment phase.

The improved two-phase heuristic method can solve the test problems without the bell time optimization optimally (or near-optimally) using much less time than the above two methods. For the test problems with the bell time optimization, the improved two-phase heuristic method performs much better than the previous two methods on all the test problems in terms of the solution quality and the computational time. It can even achieve fewer buses much quicker, especially for complicated problems that involve more trips, depots, schools, and larger dismissal time windows.

Though the improved two-phase heuristic method is powerful, we still want to have an efficient model to solve the MDSBSPTW without dividing it into different phases. So here comes our last model, the Tabu Search-based Ant Colony Optimization (TS-ACO) method. The tabu search, just like the simulated annealing algorithm, is used to find the best combination of the school bell time plan. It iteratively searches for a better solution and uses a special data structure called a tabu list to forbid or penalize certain moves to avoid cycling or revisiting a solution to improve efficiency. Under each fixed school bell time plan, the ant colony optimization algorithm is used to find the best bus schedule with the minimum number of buses and total deadhead duration. The ant colony optimization used in this study includes the solution construction process of the ants. We also introduced three local search procedures to improve the best solution found by the ant. The proposed TS-ACO method can solve both SDSBPTWs and MDSBSPTWs. The proposed ACO is enough for finding the final solution if the problem is without the bell time optimization.

The TS-ACO method also works well on the test problems without the dismissal time optimization. It always performs better than the MIP model regarding model running time and the solution quality for the test problems with the bell time optimization. It is better than the two-phase heuristic method when the problem gets more complicated (e.g., having more depots, more trips, more schools, or a larger dismissal time window). However, the improved two-phase heuristic performs even better.

Regarding the performances of all the proposed methods, in summary, results show that all four methods perform well on the problems without the bell time optimization. When incorporating the bell time optimization into the bus scheduling problem, the improved two-phase heuristic method and the TS-ACO method outperform the MIP model and the two-phase heuristic method that are limited by the Gurobi solver. Among them, the improved two-phase heuristic method has the best performance in terms of the model running time and the solution quality. It can help the school district examine different schedules or bell time plans efficiently and quickly find the best one.

For the test problems with bell time optimization, most schools' dismissal times will change after the optimization. The dismissal times after the optimization tend to be either at the boundary values of the given dismissal time window or near those boundary values, no matter what method is used. Also, after the school dismissal times are changed, almost all the single-trip bus routes are eliminated. Most bus routes tend to become longer by serving more trips, resulting in significant savings regarding the number of buses. However, the dismissal time optimization makes the integrated

problem much more complicated. For all proposed methods, results show that the larger the dismissal time window is, the more buses can be reduced, but longer model running time and computational resources are needed.

7.2 Recommendations for Future Work

There are several interesting avenues for future work, and we list some recommended directions for future research:

1. The proposed tabu search method used the most intuitive aspiration criteria for overriding the tabu status of some schools corresponding to search steps that lead to improvements. In the future, we can enhance the aspiration criteria by simultaneously considering the tabu status of those schools and the change in the objective function after overriding those schools' tabu status to see if we can improve the performance of the proposed TS-ACO method.
2. The trips are fixed for each school in this study (i.e., the visiting sequence of bus stops on the trips is known and fixed). Since they are the critical inputs for our school bus scheduling problem, we can first optimize them and even allow mixed load (i.e., a trip can serve stops from more than one school) by first solving a school bus routing problem. Combined with the current study, we can have a comprehensive school bus routing and scheduling system.
3. We assume that the buses are homogeneous (have the same capacity), and all the students don't have special needs in this study. However, in reality, the bus fleet always consists of buses with different capacities for serving different types of students (e.g., special education students and general education students) in a

school district. Therefore, in the future, developing a model that considers the mixed fleet (buses have varying bus capacities) could make the model able to simulate situations closer to real-world cases. If we introduce multiple types of buses into the model, the fixed cost for each type of bus should be different.

4. We can add some level-of-service constraints to the existing model for improving the equity of the school bus system. For example, we can add the maximum vehicle time per route or maximum length per route constraints into the model. By doing so, the vehicle time (or route length) of bus routes can be more balanced.
5. We can consider the uncertainty of the travel time and the pickup and drop-off time. We assume that all the buses arrive at schools on time, and each trip has its fixed service time duration in this study. However, buses may not always arrive at schools or serve trips on time due to bad traffic conditions. Also, the service time of each trip may vary depending on the traffic condition. Therefore, the developed model could be further improved by accounting for the stochasticity of the travel time so that we can take the real-world traffic condition into account.

References

Alinaghian, M., and N. Shokouhi, 2018, "Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search." *Omega*, 76: 85-99.

Arnold, F., K. Sörensen, 2019, "Knowledge-guided local search for the vehicle routing problem." *Computers and Operations Research*, 105: 32-46.

Babaei, M., and M. Rajabi-Bahaabadi, 2019, "School bus routing and scheduling with stochastic time-dependent travel times considering on-time arrival reliability." *Computers and Industrial Engineering*, 138: 106125.

Baldacci, R., and A. Mingozzi, 2009, "A unified exact method for solving different classes of vehicle routing problems." *Mathematical Programming*, 120(2): 347.

Banerjee, D., and K. Smilowitz, 2019, "Incorporating equity into the school bus scheduling problem," *Transportation Research Part E: Logistics and Transportation Review*, 131: 228-246.

Barma, P. S., J. Dutta, and A. Mukherjee, 2019, "A 2-opt guided discrete antlion optimization algorithm for multi-depot vehicle routing problem." *Decision Making: Applications in Management and Engineering*, 2(2): 112-125.

Bezerra, S. N., S. R. de Souza, and M. J. F. Souza, 2018, "A GVNS algorithm for solving the multi-depot vehicle routing problem." *Electronic Notes in Discrete Mathematics*, 66: 167-174.

Bektaş, T., and S. Elmastaş, 2007, "Solving school bus routing problems through integer programming." *Journal of the Operational Research Society*, 58(12): 1599-1604.

Bertsimas, D., A. Delarue, and M. Sebastien, 2019, "Optimizing schools' start time and bus routes." *Proceedings of the National Academy of Sciences*, 116(13): 5943-5948.

Bettinelli, A., A. Ceselli, and G. Righini, 2011, "A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows." *Transportation Research Part C: Emerging Technologies*, 19(5): 723-740.

Brandão, J., 2020, "A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem." *European Journal of Operational Research*, 284(2): 559-571.

Caceres, H., R. Batta, and Q. He, 2017, "School bus routing with stochastic demand and duration constraints." *Transportation Science*, 51(4): 1349-1364.

Chen, X., Y. Kong, L. Dang, Y. Hou, and X. Ye, 2015, "Exact and metaheuristic approaches for a bi-objective school bus scheduling problem." *PloS One*, 10(7): e0132600.

Christiaens, J., G. Vanden Berghe, 2020, "Slack induction by string removals for vehicle routing problems." *Transportation Sciences*, 54(2): 417-433.

Clarke, G. and J. W. Wright, 1964, "Scheduling of vehicles from a central depot to a number of delivery points." *Operations Research*, 12(4): 568-581.

Contardo, C. and R. Martinelli, 2014, "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints." *Discrete Optimization*, 12: 129-146.

Cordeau, J. F., G. Laporte, and A. Mercier, 2001, "A unified tabu search heuristic for vehicle routing problems with time windows." *Journal of Operational Research Society*, 52: 928-936.

Cordeau, J. F., G. Laporte, and A. Mercier, 2004, “Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows.” *Journal of Operational Research Society*, 55: 542-546.

Cordeadu, J. F., and M. Mercier, 2012, “A parallel iterated tabu search heuristic for vehicle routing problems.” *Computers and Operations Research*, 39(9): 2033-2050.

Crevier, B., J. F. Cordeau, and G. Laporte, 2007, “The multi-depot vehicle routing problem with inter-depot routes.” *European Journal of Operational Research*, 176(2): 756-773.

Dantzig, G. B., and J. H. Ramser, 1959, “The truck dispatching problem.” *Management Science*, 6(1): 80-91.

de Brey, C., T. D. Snyder, A. Zhang, and S. A. Dillow, 2021, “Digest of Education Statistics 2019. NCES 2021-009.” *National Center for Education Statistics*, Table 236.90.

Demirel, T., and Ş. Yilmaz, 2012, “A new solution approach to multi-depot vehicle routing problem with any colony optimization.” *Journal of Multiple-Valued Logic and Soft Computing*, 18.

Dorigo, M., and V. Maniezzo, and A. Colorni, 1996, “Ant system: optimization by a colony of cooperating agents.” *IEEE Transaction on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1): 29-41.

Ellegood, W. A., S. Solomon, J. North, and J. F. Campbell, 2020 “School bus routing problem: Contemporary trends and research directions.” *Omega*, 95: 102056.

Escobar, J. W., R. Linfati, P. Toth, and M. G. Baldoquin, 2014, "A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem." *Journal of Heuristics*, 20(5): 483-509.

Fügenschuh, A., 2009, "Solving a school bus scheduling problem with integer programming." *European Journal of Operational Research*, 193(3): 867-884.

Fügenschuh, A., 2011, "A set partitioning reformulation of a school bus scheduling problem." *Journal of Scheduling*, 14(4): 307-38.

Gambardella, L. M., É. Taillard, and G. Agazzi, 1999, "MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows." *New Ideas in Optimization*.

Giosa, I. D., I. L. Tansini, and I. O. Viera, 2002, "New assignment algorithms for the multi-depot vehicle routing problem." *Journal of the Operational Research Society*, 53(9): 977-984.

Glover, F., and E. Taillard, 1993, "A user's guide to tabu search." *Annals of Operations Research*, 41(1): 1-28.

Haghani, A., and M. Banihashemi, 2002, "Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints." *Transportation Research Part A: Policy and Practice*, 36(4): 309-333.

Ho, W., G. T. Ho, P. Ji, and H. C. Lau, 2008, "A hybrid genetic algorithm for the multi-depot vehicle routing problem." *Engineering Applications of Artificial Intelligence*, 21(4): 548-557.

Jabir, E., V. V. Paincker, and R. Sridharan, 2017, "Design and development of a hybrid any colony-variable neighborhood search algorithm for a multi-depot green vehicle

routing problem." Transportation Research Part D: Transport and Environment, 57: 422-457.

Karakatič, S., and V. Podgorelec, 2015, "A survey of genetic algorithms for solving multi depot vehicle routing problem." *Applied Soft Computing*, 27: 519-532.

Kim, B. I., S. Kim, and J, Park, 2012, "A school bus scheduling problem." *European Journal of Operational Research*, 218(2): 577-585.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, 1983, "Optimization by simulated annealing." *Science*, 220(4598): 671-680.

Kytöjoki, J., T. Nuortio, O. Bräysy, M. Gendreau, 2007, "An efficient variable neighborhood search heuristic for very large scale vehicle routing problems." *Computer and Operations Research*, 34(9): 2743-2757.

Laporte, G., Y. Nobert, and D. Arpin, 1984, "Optimal solutions to capacitated multidepot vehicle routing problems." *Congressus Nemerantium*, 4: 283-292.

Laporte, G., Y. Nobert, and S. Taillefer, 1988, "Solving a family of multi-depot vehicle routing and location-routing problems." *Transportation Science*, 22(3): 161-172.

Laporte, G., S. Ropke, and T. Vidal, 2014, "Chapter 4: Heuristics for the vehicle routing problem." In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Society for Industrial and Applied Mathematics, 87-116.

Leachman, M., K. Masterson., and E. Figueroa, 2017, "A punishing decade for school funding." *Center on Budget and Policy Priorities*, 29.

Li, J., Y. Li, and P. M. Pardalos, 2016, "Multi-depot vehicle routing problem with time windows under shared depot resources." *Journal of Combinatorial Optimization*, 31(2): 515-532.

Löbel, A., 1998, "Vehicle scheduling in public transit and Lagrangean pricing." *Management Science*, 44(12-part-1), 1637-1649.

Luo, J., and M. R. Chen, 2014, "Multi-phase modified shuffled frog leaping algorithm with external optimization for the MDVRP and the MDVRPTW." *Computers and Industrial Engineering*, 72: 84-97.

Miranda, D.M., R. S. de Camargo, S. V. Conceição, M. F. Porto, and N. T Nunes, 2021, "A metaheuristic for the rural school bus routing problem with bell adjustment." *Expert Systems with Applications*, 180: 115086.

Montoya-Torres, J. R., J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla, 2015, "A literature review on the vehicle routing problem with multiple depots." *Computers and Industrial Engineering*, 79: 115-129.

Nagy, G. and S. Salhi, 2005, "Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries." *European Journal of Operational Research*, 162(1): 126-141.

Narasimha, K. V., E. Kivelevitch, B. Sharma, and M. Kumar, 2013, "An ant colony optimization technique for solving min-max multi-depot vehicle routing problem." *Swarm and Evolutionary Computation*, 13: 63-73.

Park, J., and B-I. Kim, 2010, "The school bus routing problem: A review." *European Journal of Operational Research*, 202: 311-319.

Pecin, D., A. Pessoa, M. Poggi, and E. Uchoa, 2017, "Improved branch-cut-and-price for capacitated vehicle routing." *Mathematical Programming Computation*, 9(1): 61-100.

Polacek, M., R. F. Hartl, K. Doerner, and M. Reimann, 2004, "A variable neighborhood search for the multi depot vehicle routing problem with time windows." *Journal of Heuristics*, 10(6): 613-627.

Polacek, M., S. Benkner, K. F. Doerner, and R. F. Hartl, 2008, "A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows." *Business Research*, 1(2): 207-218.

Queiroga, E., R. Sadykov, and E. Uchoa, 2021, "A POPMUSIC for the capacitated vehicle routing problem." *Computers and Operations Research*, 136: 105475.

Ramos, T. R. P., M. I. Gomes, and A. P. B. Póvoa, 2020, "Multi-depot vehicle routing problem: a comparative study of alternative formulations." *International Journal of Logistics Research and Application*, 23(2): 103-120.

Renaud, J., G. Laporte, and F. F. Boctor, 1996, "A tabu search heuristic for the multi-depot vehicle routing problem." *Computers and Operations Research*, 23(3): 229-235.

Sadati, M. E. H., B. Çatay, D. Aksen, 2021 "An efficient variable neighborhood search with tabu shaking for a class of multi-depot vehicle routing problems.", *Computers and Operations Research*, 133: 105269.

Salhi, S. and M. Sari, 1997, "A multi-level composite heuristic for the multi-depot vehicle fleet mix problem." *European Journal of Operational Research*, 103(1): 95-112.

Salhi, S., A. Imran, and N. A. Wassan, 2014, "The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation." *Computers and Operations Research*, 52: 315-325.

Savas, E. S., 1978, "On equity in providing public services." *Management Science*, 24(8): 800-808.

Shafahi, A., S. Aliari, and A. Haghani, 2018, "Balanced Scheduling of School Bus Trips using Perfect Matching Heuristic." *Transportation Research Record*, 2672(48): 1-11.

Subramanian, A., E. Uchoa, and L. S. Ochi, 2013, "A hybrid algorithm for a class of vehicle routing problems." *Computers and Operations Research*, 40(10), 2519-2531.

Tillman, F. A., 1969, "The multiple terminal delivery problem with probabilistic demands." *Transportation Science*, 3(3): 192-204.

Toth, P., and D. Vigo, 2002, "*The vehicle routing problem*." Society for Industrial and Applied Mathematics.

Tu, W., Z. Fang, Q. Li, S. L. Shaw, and B. Chen, 2014, "A bi-level Voronol diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem." *Transportation Research Part E: Logistics and Transportation Review*, 61: 84-97.

Vidal, T., T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, 2012, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems." *Operation Research*, 60(3): 611-624.

Vidal, T., T. G. Crainic, M. Gendreau, and C. Prins, 2013, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows." *Computers and Operations Research*, 40(1): 475-489.

Wang, X., B. Golden, E. Wasil, and R. Zhang, 2016, "The min-max split delivery multi-depot vehicle routing problem with minimum service time requirement." *Computer and Operations Research*, 71: 110-126.

Wang, Z., 2019, "Solving the integrated school bell time, and bus routing and scheduling optimization problem under the deterministic and stochastic conditions." UMD Dissertation.

Wang, Z., and A. Haghani, 2020, "Column generation based stochastic school bell time and bus scheduling optimization." *European Journal of Operational Research*, 286(3): 1087-1102.

Wang, Z., A. Shafahi, and A. Haghani, 2017, "SCDA: School compatibility decomposition algorithm for solving the multi-school bus routing and scheduling problem." *arXiv preprint arXiv*:1711.00532.

Wei, L., Z. Zhang, D. Zhang, and S. C. Leung, 2018, "A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints." *European Journal of Operational Research*, 265(3): 843-859.

Yan, S., F. Y. Hsiao, and Y. C. Chen, 2015, "Inter-school bus scheduling under stochastic travel times." *Networks and Spatial Economics*, 15(4): 1049-1074.

Yu, B., Z. Z. Yang, and J. X. Xie, 2011, "A parallel improved ant colony optimization for multi-depot vehicle routing problem." *Journal of the Operational Research Society*, 62(1): 183-188.

Yücenur, G. N., and N. Ç. Demirel, 2011, "A new geometric shape-based genetic clustering algorithm for the multi-depot vehicle routing problem." *Expert Systems with Applications*, 38(9): 11859-11865.

Zhang, S., W. Zhang, Y. Gajpal, S. S. Appadoo, 2019, "Ant colony algorithm for routing alternative fuel vehicles in multi-depot vehicle routing problem." In *Decision science in action: 251-260.* Springer, Singapore.

Zhou, L., R. Baldacci, D. Vigo, and X. Wang, 2018, “A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution.” *European Journal of Operational Research*, 265(2): 765-778.