Research Article

A Branch-and-Price-and-Cut Algorithm for the Integrated Scheduling and Rostering Problem of Bus Drivers

Dung-Ying Lin,1 Chieh-Ju Juan,2 and Ching-Chih Chang2

1Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan
2Department of Transportation and Communication Management Science, National Cheng Kung University, Tainan 70101, Taiwan

Correspondence should be addressed to Chieh-Ju Juan; r58061021@mail.ncku.edu.tw

Received 19 August 2019; Revised 6 December 2019; Accepted 26 December 2019; Published 25 February 2020

Abstract

In the transportation industry, crew management is typically decomposed into two phases: crew scheduling and crew rostering. Due to the complexity of scheduling and rostering, bus transportation is not an exception and many relevant studies do not consider both procedures simultaneously. However, such a decomposition can yield inferior schedules/rosters. To address this issue, this paper proposes an integrated scheduling and rostering model for bus drivers and devises a branch-and-price-and-cut (BPC) algorithm to solve the complex problem. The proposed solution framework is empirically applied to real-world instances with various problem sizes whose data is collected from H Bus Company located in southern Taiwan. To validate the effectiveness and evaluate the efficiency of the proposed solution framework, this paper compares the solution obtained from the BPC algorithm with that of a benchmark optimization package. The results show that the proposed BPC algorithm can solve problems with large real-world instances within a reasonable computational time. Moreover, in the numerical experiments, this paper finds that the scheduling and rostering results of the bus drivers are more sensitive to the rostering constraints. Also, the proposed integrated framework can yield a better solution than the solution from a conventional two-phase approach, which demonstrates the advantage of the integration in this paper. The proposed method provided can be employed to deal with the challenges in driver planning for bus companies.

1. Introduction

As the public bus transport system produces less pollution than private transport, improving the quality of bus services and encouraging their use have become significant issues with the growing importance for energy conservation and carbon emissions reduction. A bus company typically needs to address several issues regarding its daily operation: timetabling, vehicle scheduling, crew scheduling, and crew rostering [1]. Ma et al. [2] noted that crew scheduling and rostering are the importance activities in the bus operation process. The scheduling and rostering problems are closely related and are typically decomposed into two phases because of the significant complexity when considered simultaneously. In the bus driver scheduling problem, the aim is to find a set of feasible daily duties that covers all the trips [3]. After generating a duty set, the driver rostering problem assigns the duties to the drivers while incorporating various practical considerations, such as driver preferences and the labor laws, which must be taken into account [4]. These two problems are typically solved sequentially due to the significant problem complexity and computational resources required to determine a feasible schedule and roster. However, as can be expected, the decomposition can result in inferior solutions compared to the solutions obtained from an integrated framework. To address this discrepancy, the current research proposes an integrated crew scheduling and rostering formulation.

To optimally solve the integrated problem, one should theoretically consider all the potential feasible driver duties, identify the minimum number of duties so that a timetable can be fully covered, and form a feasible roster that requires the minimum number of drivers. Unfortunately, enumerating all the possible driver duties/columns is an impractical
approach because the number of driver duties/columns can grow exponentially as the problem size increases. Even if all the duties/columns are enumerated, solving the resulting problem is an extremely challenging task. To avoid these issues, this paper first relaxes the integer constraint of the decision variables and constructs a branch-and-bound solution scheme. This paper then designs a column generation algorithm that decomposes this complicated problem into a restricted master problem (RMP) and a series of pricing subproblems. The RMP corresponds to the rostering problem that orders the duties generated from the pricing problem in the optimal manner. Based on the solution from the RMP, the dual variables are passed to the pricing subproblem. In the pricing subproblem, this paper designs a time constrained shortest path network that generates the columns based on the given dual variables and feeds back to the RMP. Instead of enumerating all the potential duties/columns, the pricing problem generates the duties/columns on-the-fly and only finds the good ones for the RMP. The pricing subproblem also serves as the stopping criterion of the column generation solution procedure if the corresponding reduced cost indicates that no more column can be generated to improve the RMP. After finalizing the column generation procedure, this paper devises a series of cuts to fathom the branch-and-bound tree and reduce the feasible solution space so that the overall procedure can be accelerated. Therefore, the resulting algorithm is a branch-and-price-and-cut (BPC) algorithm.

The proposed algorithms are empirically applied to real-world cases of integrated bus driver scheduling and rostering cases. The empirical results show that the integrated framework produces a superior solution when compared to solving the scheduling and rostering subproblems sequentially. In addition, the proposed algorithm outperforms a commonly used commercial optimization package and is capable of determining the optimal solution of a practical size. Interesting observations and managerial insights are presented together with the empirical results. It is worth noting that from the preliminary experiments, the proposed BPC algorithm can solve randomly generated problem instances of even larger sizes. However, the lack of realistic data prevents this paper from going further. Therefore, the current paper only summarizes the results that are comparable to either real-world cases or benchmark optimization package. The presented results show the solution quality and computational efficiency of the proposed solution framework.

The remainder of this paper is structured as follows. Section 2 critically overviewed the related work in the existing literature, and Section 3 presents a mathematical model for the integrated bus driver scheduling and rostering problem. The branch-and-price-and-cut algorithm used for solving the problem is detailed in Section 4, and the empirical studies are summarized in Section 5. The final section offers the conclusions and suggestions for future research.

2. Literature Review

In this section, this paper critically reviews the studies on crew scheduling, rostering, and integrated problems in bus driver management. The proposed models and solution methods devised are overviewed. Note that there exists a significant amount of research on scheduling, rostering, and integrated problems in railway and airline transportation. Interested readers are referred to the review papers from Zeren and Özkol [5], Janacek et al. [6], Doi et al. [7], Nishi et al. [8], Salazar-González [9], and Souai and Teghem [10]. This section focuses primarily on bus driver management problem.

The crew scheduling problem aims to create a set of daily duties that can cover all the tasks in the published timetables while satisfying the labor laws and the contractual rules of companies [11]. In bus transportation, many studies have focused on the crew scheduling problem. Chen and Niu [12] developed an integer programming model for the crew scheduling problem of a circle bus line with the objective of minimizing the crew idle time while considering multiple duty types. Then, a tabu search algorithm is proposed to solve the model. Ma et al. [2] proposed a variable neighborhood search (VNS) algorithm to solve the bus driver scheduling problem. In the case study of two depots of the Beijing Public Transport Group, the results showed that the VNS-based algorithm is able to effectively reduce the total driver costs. De Leone et al. [1] introduced a new mathematical model to solve the bus driver scheduling problem in real-world scenarios. Because the exact model can only solve small or medium sized problems, a greedy randomized adaptive search procedure (GRASP) was developed to tackle the larger problem instances, and the numerical comparison shows the effectiveness of the proposed metaheuristic method.

Perumal et al. [13] presented a matheuristic that combines a metaheuristic and a mathematical programming technique to solve a driver scheduling problem with staff cars and compared their results with those obtained by the state-of-the-art mixed integer programming method. The comparison indicates that the metaheuristic provides higher quality solutions with less computational time. Toth and Kresz [14] proposed a methodology based on a cut and join (CAJ) approach to solve a driver scheduling problem. Their approach consists of two phases. The first phase produced the rough shifts, while the second phase produced the complete shifts containing the obligatory activities and filling the idle times. Their case study showed the efficiency and flexibility of the approach when compared with the present schedules of the company. A column generation approach was proposed by Lin and Hsu [15] to solve the bus scheduling problem. This study decomposed the problem into a master problem and a series of pricing subproblems. The master problem identified the optimal duties from a set of feasible duties, and the pricing subproblems created the necessary duties for the duties set to improve the results obtained from the master problem. These studies provided interesting insights into solving the crew scheduling problem for bus companies.

Xie and Suhl [16] classified the driver rostering problem into cyclic and noncyclic rosters. This paper focuses on the cyclic roster, in which every driver is fairly assigned each duty by turn. After obtaining a set of feasible duties in the first stage (crew scheduling problem), the second stage (crew rostering problem) assigns a sequence of duties to the drivers.
to form the roster so that the drivers can execute the duties in rotation. Each driver must be assigned one duty or day-off per day, and the resulting roster must follow the government regulations for issues such as the minimum rest time and maximum working hours [17]. Crew rostering problem has also attracted the attention of many researchers. Xie and Suhl [16] provided a multicommodity network flow formulation for the crew rostering problem of German bus companies. The network model can solve both the cyclic and noncyclic crew rostering problems by using commercial solvers. Xie et al. [18] presented a multiobjective optimization problem for the bus driver rostering problem, which was solved by metaheuristics, which included ant colony optimization, simulated annealing, and tabu search methods. Respício et al. [19] devoted their efforts to developing a new memetic evolutionary heuristic for a bus driver rostering problem. The heuristic combined a decoder, a strength Pareto fitness evaluation, and a local search procedure. The empirical results showed that the proposed approach can improve the solutions for the difficult problem; furthermore, a set of potentially efficient rosters can be created for the planner to make a decision.

Mesquita et al. [20] proposed two multicommodity flow models for the driver rostering problem with day-off patterns and developed a new decompose-and-fix heuristic to solve the models. The computational experiments were carried out with instances of various sizes, and the results showed the good performance of the heuristic. Nishi et al. [8] established a two-level decomposition approach for crew rostering problems with the objective of ensuring fair working conditions. An efficient branch-and-bound algorithm was developed to solve the master problem, and a valid cut approach was applied to reduce the feasible search space to tighten the duality gap.

In the crew management problem, the scheduling problem and the rostering problem are typically solved sequentially due to the complexity when solving both problems simultaneously [21]. The scheduling problem creates a set of daily duties, which is then inputted into the rostering problem to form a roster. It is not surprising that the two-stage process may potentially yield inferior solutions. Therefore, researchers have begun to integrate both problems to improve the quality of the solution. Ernst et al. [22] presented an integrated optimization model for both crew scheduling and crew rostering, which was able to generate either cyclic rosters or noncyclic rosters. Due to the computational complexity, the model can only be solved using the relaxed version. Freling et al. [23] provided insight into the implementation issues of crew scheduling and crew rostering and showed the difficult details in practical problems that can be solved by using a branch-and-price algorithm. For the bus industry, although the crew scheduling and rostering problems are considered within a solution framework, the two subproblems are solved sequentially by Yunes et al. [24]. These subproblems were formulated based on mathematical programming (MP) and constraint logic programming (CLP) principles and solved by a hybrid column generation algorithm [24]. Lin and Tsai [25] provided an integrated formulation and solution algorithm for crew management. However, their work focused on the crew management of railway transportation that has very different scheduling/rostering rules. As a result, the mathematical formulation, decomposed formulation, and solution techniques are not applicable to bus driver scheduling.

Furthermore, as the rapid development of data technology, a fair amount of research is devoted to developing novel schemes to improve the public transportation system based on big data. Graves et al. [26] presented a set of metrics which is the additional time customers spend waiting for and riding buses in excess of the schedule, and the proposed methodology leveraged an origin-destination ridership model and bus location data to calculate these values for each individual passenger. Compared to the past bus-level measurement, this study is measured at the passenger level which enables the design of service management techniques to benefit the most riders possible. Besides, in order to satisfy the urban traffic demand and alleviate traffic congestion, Yang et al. [27] proposed an urban taxi fleet size calculating model based on GPS tracking data to meet the urban transportation demand for taxis. The results showed that the method can effectively calculate the required fleet size to mitigate the urban traffic congestion problem. Zeng et al. [28] developed a highly optimized network-generation tool to estimate bus link loads and boarding and alighting locations based on the data collected from the vehicle location system and fare collection system. This proposed method can support the operations management and the monitoring of patterns and trends in ridership behavior for bus service scheduling and planning. Moreover, a refined public transportation system can alleviate energy consumption and carbon emission in an urban city. Some studies are dedicated to assessing energy conservation and emission reduction effect on urban transportation. Yang et al. [29] proposed the improved oscillating value method based on entropy weight to develop an evaluation system which was able to evaluate energy conservation and emission reduction effect on traffic and transportation. Stankovic et al. [30] proposed a method connecting monitoring stations and automatic traffic control system to manage air pollution in critical zones of the city by activating a signaling cycle to change the direction of traffic route. Tang et al. [31] developed a simulated model to analyse the key factors which affected the location of electric vehicle charging stations, and the objective is to minimize the total travel distance of cars. Xu [32] proposed a full velocity difference model considering driver’s reaction delay for various types of traffic jams.

After reviewing the abovementioned literature regarding the crew management problem, this paper summarizes the observations as follows: (1) The column generation technique is the mainstream method for the crew management problem. (2) There are limited studies on the integrated model due to the computational challenge of each scheduling and rostering problem.

The crew scheduling problem is typically formulated as a set covering or set partitioning problem [33]. The disadvantage of such models is that numerous feasible duties are created to find a set of duties to cover the given tasks. As the
problem size increases, the large number of duties corresponding to the numerous columns would make the problem difficult to solve and even make identifying the feasible solutions challenging within an acceptable computational time. Therefore, the column generation procedure is used to create only the necessary duties to effectively reduce the computational effort.

Even if considering the integration, the proposed manner can only deal with the relaxed version or solve the two-phase problem sequentially. Consequently, this paper focuses on formulating the crew scheduling and crew rostering problem as an integrated model and developing an algorithm that combines column generation and branch-and-bound techniques to deal with real-size instances.

3. Mathematical Formulation

In this section, a formulation of the integrated bus driver scheduling and rostering problem is presented. Section 3.1 offers the definitions and assumptions used throughout the paper, followed by the solution design in Section 3.2. Based on the problem statement and solution design, Section 3.3 describes the mathematical formulation.

3.1. Definitions and Assumptions. The definitions are first introduced so that the problem can be formally stated based on these definitions.

(i) Task. A published bus timetable is formed by trips, which can be further divided into small tasks that can be serviced by a bus driver. A task is characterized by its start/end times and start/end stops.

(ii) Duty. A duty is formed by combining one or more tasks and is serviced by a driver in a day. A feasible duty should not violate any scheduling rules.

(iii) Dummy Task. In this research, dummy tasks are introduced to establish a solution structure. There are two types of dummy tasks. (1) Task of set $T_{\text{group}}$: the dummy task that represents the end of a group. A group is the subset of tasks employed in the solution design, which will be detailed in Section 3.2. (2) Task of set $T_{\text{dum}}$: the dummy task that represents the end of a duty.

(iv) Scheduling. The driver scheduling is to create a set of duties with which all the tasks from a published bus timetable can be adequately covered. A feasible duty must obey the following rules:

1. All the tasks from the bus timetable should be covered.
2. Only when the end stop of one task is the same as the start stop of the next task can the two tasks be consecutively assigned to a duty. Otherwise, a connection task should be inserted to connect these two tasks.
3. Only when the end time of one task is earlier than the start time of the next task can these two tasks be serviced sequentially by a driver.
4. The working hours of a duty in a day cannot exceed the maximum daily working hours regulated by the labor law.
5. According to the labor laws in this case study, an adequate break time (i.e., 30 minutes) is needed after a driver works for 4 hours continuously. Furthermore, the working hours should be calculated in a rolling manner, which means that every continuous 4 working hours beginning from the start time of every task must contain an adequate break time. Note that a time slot between tasks that is less than 15 minutes is not considered as a break time in this case. The relevant research studies investigating the crew scheduling problem consider the parameter of minimum length of break time that can be referred to Haase et al. [34] and Shen and Chen [35].
6. A driver has to return to her/his home stop on a daily basis.

(v) Roster. A roster is a sequence of duties assigned to a set of bus drivers on a weekly/monthly/seasonal basis. This paper investigates a cyclic driver rostering problem, in which drivers grouped together are trained to have identical qualifications and use the same roster. For the noncyclic driver rostering problem, interested readers are referred to the review papers from Mesquita et al. [20], Peng et al. [36], and Xie et al. [18]. The following rules should be obeyed when constructing a roster in this case:

1. The weekly working hours cannot exceed the maximum weekly working hours regulated by the labor law.
2. The rest time between the current duty and the duty from the next day should be at least 11 hours.

Next, the assumptions are presented so that the following discussion can be concentrated on the model and algorithm design:

1. No delay or early departure. All the tasks are performed on time according to a published bus timetable.
2. No overtime or insufficient rest/break time. The rest/break time and maximum working hours regulated by the labor and contractual rules cannot be violated.
3. Identical drivers. The skills or qualifications of the drivers are the same; therefore, all the tasks can be performed by all the drivers.
4. Similar driver preferences. The current paper focuses on the cyclic driver rostering problem, in which a roster is generated for a group of drivers who have identical skills and similar preferences [16].

3.2. Solution Design. Based on the definitions and assumptions described in the earlier section, the solution design for the
mathematical formulation is presented. The duty illustrated in Figure 1 is used as an example to explain the details. For each duty, there are \( r \) groups, and each group contains \( j \) cells. Each cell can be assigned a task. As mentioned earlier, a time slot between tasks that is less than 15 minutes should not be considered as a break time. To obey the break time rule calculated in a rolling manner, the solution design constructs groups to ensure that the working hours in each group does not exceed 4 hours and that the break time between two consecutives groups is greater than 15 minutes. In other words, if the break time between two consecutives tasks is greater than 15 minutes, the previous task should be placed in the last cell of group \( i \), and the following task will be assigned in the first cell of group \( i + 1 \). Therefore, no break time will be assigned within a group, and a break time only exists between two consecutives groups. Furthermore, if the working hour of groups \( i \) and \( i + 1 \) exceeds 4 hours, an adequate break time will be given between these two groups.

For a feasible solution, this study considers \( w \) weeks and \( d \) days, as illustrated in Figure 2. Based on the labor laws, there should be a day-off assigned during the week. In other words, only 6 working days can be assigned within a week. Therefore, the index of day \( d \) is up to 6. An arbitrary large number, \( M \), is set as the largest number of weeks so that all the duties can be adequately assigned. Furthermore, this paper provides an easy method to estimate the value of \( M \).

Suppose that an experiment determines 12 duties in the initial solution procedure that is described in Section 4.1. The upper bound, \( M \), can be estimated based on this number. In a loose manner, 3 duties are assigned in a week and a day-off is placed between each two duties. Let the last day of the week be the day-off required by the labor law. In this way, the rest time and the maximum weekly working hour constraint will not be violated. In this case, \( M \) is set as 4 (12/3 = 4). The resulting roster can contain all the duties generated from the initial solution within \( M \) weeks.

Additionally, a penalty term, \( P_{w,d} \), is employed in the objective function of the mathematical formulation to minimize the number of days in a roster. As each duty is assigned to a driver, minimizing the number of days in a roster is equivalent to minimizing the number of drivers needed to perform all the duties.

3.3. Mathematical Formulation of the Integrated Bus Driver Scheduling and Rostering Problem. Next, the sets, parameters, and decision variables are summarized which will be used in the mathematical formulation.

Sets:

- \( T \): set of all the tasks, including the working and dummy tasks.
- \( T^w \): set of working tasks, \( T^w \in T \).
- \( T^dum \): set of dummy tasks that represents the end of a group.
- \( T^dum \): set of dummy tasks that represents the end of a duty.

Parameters:

- \( I \): maximum number of groups for construction of the roster.
- \( J \): maximum number of cells in each group.
- \( D \): the number of working days in a week.
- \( M \): the maximum number of weeks in a roster.
- \( W_{\text{check}} \): the maximum daily working hours.
- \( W_{\text{max}} \): the maximum weekly working hours.
- \( R \): the break time between group \( i \) and \( i + 1 \).
- \( R_{\text{day}} \): the number of resting hours specified by the labor laws between two consecutive days.
- \( W_{\text{break}} \): the start time of task \( t \).
- \( W_{\text{end}} \): the end time of task \( t \).
- \( W_{\text{check}} \): the driving time of task \( t \).

Decision Variables:

- \( x_{i,j,t}^w \): If \( x_{i,j,t}^w = 1 \), task \( t \) is assigned to cell \( j \) of group \( i \) of day \( d \) in week \( w \). Otherwise, \( x_{i,j,t}^w = 0 \).
- \( y_{i,j,t} \): If \( y_{i,j,t} = 1 \), then task \( t \) can be performed by a driver after she/he executes task \( t \), which is assigned to cell \( j \) of group \( i \) of day \( d \) in week \( w \). In this case, task \( t_2 \) is assigned to cell \( j + 1 \). Otherwise, \( y_{i,j,t} = 0 \).
- \( z_{i,j,t}^d \): If \( z_{i,j,t}^d = 1 \), task \( t_3 \) is the last task of group \( i \) and \( t_3 \) is the first task of group \( i + 1 \). Otherwise, \( z_{i,j,t}^d = 0 \).
Time between tasks < 15min
Break time ≥ 15min

Figure 1: The structure of a feasible duty.

<table>
<thead>
<tr>
<th>i = 1</th>
<th>i = 2</th>
<th>i = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w = 1</td>
<td>w = 1</td>
<td>w = 1</td>
</tr>
<tr>
<td>d = 1</td>
<td>d = 2</td>
<td>d = 6</td>
</tr>
<tr>
<td>j = 1</td>
<td>j = 1</td>
<td>j = 1</td>
</tr>
<tr>
<td>j = 2</td>
<td>j = 2</td>
<td>j = 2</td>
</tr>
<tr>
<td>j = j</td>
<td>j = j</td>
<td>j = j</td>
</tr>
</tbody>
</table>

Figure 2: An illustrative solution of the integrated rostering and scheduling problem.
\[ \text{scheduling and roster problem:} \]

\[ \text{Journal of Advanced Transportation 7, 1t, 1} \]

\[ \sum_{i,j} x_{t,1,t}^{w,d} = 1, \quad \forall w, \forall t, \forall i, \forall j, \quad (1) \]

\[ \sum_{w,d} \sum_{t 	prod 1} P_{w,d} \times x_{t,1,t}^{w,d} \]

\[ \text{subjects to} \]

\[ \sum_{w,d} \sum_{i,j} x_{t,1,t}^{w,d} = 1, \quad \forall t, \quad (2) \]

\[ \sum_{i,j} x_{t,1,t}^{w,d} + x_{t+1,t} - 2y_{t}^{w,d} \geq 0, \quad \forall w, \forall d, \forall i, \forall j, \quad (3) \]

\[ \forall t_1 \in T, \forall t_2 \in T, \forall t \in T, \forall w, \forall d, \forall i, \forall j, \quad (4) \]

\[ \forall w, \forall d, \forall i, \forall j, \forall t_1 \in T, \forall t_2 \in T, \forall w, \forall d, \forall i, \forall j, \forall t \in T, \quad (5) \]

\[ \sum_{t \in T} \sum_{t' \in T^0} y_{t_1,t_2}^{w,d} = 1, \quad \forall w, \forall d, \forall i, \forall j = 1, \ldots, j - 1, \quad (6) \]

\[ \sum_{t = 1}^{T} \sum_{t' \in T^0_{\text{check}}} y_{t_1,t_2}^{w,d,j} + x_{t_1+1,t_1}^{w,d} - 2z_{t_1,t_2}^{w,d,j} \geq 0, \quad \forall w, \forall d, \forall i, \forall j = 1, \ldots, j - 1, \quad \forall t_1 \in T, \forall t_2 \in T, \quad (7) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, j - 1, \quad (8) \]

\[ \sum_{t = 1}^{T} \sum_{t' \in T^0_{\text{check}}} y_{t_1,t_2}^{w,d,j} + x_{t_1+1,t_1}^{w,d} - z_{t_1,t_2}^{w,d,j} \leq 1, \quad \forall w, \forall d, \forall i = 1, \ldots, j - 1, \quad (9) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, j - 1, \quad (10) \]

\[ \sum_{t = 1}^{T} \sum_{t' \in T^0_{\text{check}}} y_{t_1,t_2}^{w,d,j} (WT_t + CH_{t_1,t_2} + PT_t) \]

\[ \sum_{i = 1}^{T} \sum_{t = 1}^{T} y_{t_1,t_2}^{w,d,j} (WT_t - CH_{\text{homestop}, t} - PT_t) \leq WH_{\text{check}}, \quad \forall w, \forall d, \forall i, \forall j, \forall t \in T, \quad (11) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, j - 1, \quad (12) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i - 1, \quad (13) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i - 1, \quad (14) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i - 1, \quad (15) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i - 1, \quad (16) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i - 1, \quad (17) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i - 1, \quad (18) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i - 1, \quad (19) \]

\[ \forall w, \forall d, \forall i = 1, \ldots, i, \quad (20) \]
The objective function (1) aims to minimize the number of days required to operate the roster. A penalty term, \( r_{wd} \), is used to punish a duty that is assigned to a later week and day. As each duty assigned in each day needs to be performed by a driver, minimizing the days of a roster is equivalent to minimizing the number of drivers needed to execute the roster. Constraint (2) ensures that each cell \((i, j)\) of day \(d\) in week \(w\) has been assigned a task. Constraint (3) ensures that each task \(t\), \(i\), will be performed. Constraints (4) and (5) connect the decision variables \(y_{i, t, j}\), \(x_{i, t, j}\), and \(z_{i, j, t}\). If \(y_{i, t, j} = 1\), indicating that tasks \(t_1\) and \(t_2\) are performed sequentially by a driver, then the decision variables \(x_{i, j, t, 1}\) and \(x_{i, j, t, 2}\) can have the value of 1, meaning that tasks \(t_1\) and \(t_2\) cannot be performed sequentially by a driver. Constraint (6) ensures that each pair of cells has one decision variable, \(y_{i, t, j}\), to define the relationship between these two cells. The decision variable \(y_{i, t, j}\) then guarantees that the connection between each pair of cells does not violate the scheduling rules. Notate that set \(T_{ij}^{t}\) is the set of tasks that can be performed by a driver without violating the scheduling rules after the driver performs task \(t\). Constraints (7) and (8) ensure the relationship among decision variables \(y_{i, t, j}\), \(x_{i, j, t, 1}\), and \(z_{i, j, t}\) and have a similar interpretation as that of Constraints (4) and (5). If \(z_{i, j, t} = 1\), then \(\sum_{t \in T_{ij}^{t}} y_{i, t, j} = 1\) and \(x_{i, j, t, 1} = 1\), meaning that task \(t_1\) is the last task of group \(i\) \((\sum_{t \in T_{ij}^{t}} y_{i, t, j} = 1)\) and \(t_3\) is the first task of group \(i + 1\) \((x_{i, j, t, 1} = 1)\). Otherwise, \(z_{i, j, t} = 0\). Constraint (9) ensures that each pair of groups has one decision variable \(z_{i, j, t}\) to define its relation between these two groups. The decision variable \(z_{i, j, t}\) then guarantees that the connection between each pair of groups does not violate the scheduling rules. Note that set \(T_{ij}^{t}\) is the set of tasks that can be performed by a driver without violating the scheduling rules after the driver performs task \(t\). The maximum daily working hour for each duty is ensured in constraint (10). The working hour of each duty calculated by the clock-out hour \((\sum_{t \in T_{ij}^{t}} y_{i, t, j} \cdot (WT_{t} + CH_{t} + PT_{e})\)) minus the clock-in hour \((\sum_{t \in T_{ij}^{t}} y_{i, t, j} \cdot (WT_{t} - CH_{parking} + PT_{e})\)) should not exceed the maximum allowed daily working hour per day \((WH_{max})\).

Note that if \(x_{i, j, t, 1} = 1\), task \(t_1\) is the first task of the duty, then the start time of the duty or task \(t\) is \(WT_{1} - CH_{parking} - PT_{e}\). Similarly, if \(\sum_{t \in T_{ij}^{t}} y_{i, t, j} = 1\), task \(t_1\) is the last task of the duty. Then, the end time of that duty or task \(t_1\) is \(WT_{t} + CH_{t} + PT_{e}\). Based on contractual and labor laws, a driver needs to have a \(WT_{break}\) minute break after continuously working for \(WT_{check}\) hours. For instance, a driver has to have 30 minutes of break after drive continuously for 4 hours. This limitation is enforced in constraints (11)–(14). The working hour for each group should not exceed \(WT_{check}\) hours, which is constrained in constraint (11). The working hour of each group (i.e., the sum of driving hour \((\sum_{i,j} x_{i,j,t} \cdot WH_{i})\) and connection hour between tasks \((\sum_{i,j} y_{i,j,t} \cdot CH_{i,j})\)) is limited by \(WT_{check}\). In constraints (12) and (13), the intermediate variable \(rb_{wd,j}\) is designed to determine if a \(WT_{break}\) minute break is required between groups \(i\) and \(i + 1\). If the working hour for the two groups \(i\) and \(i + 1\) \((\sum_{t \in T_{ij}^{t}} y_{i,j,t} \cdot WH_{i} + \sum_{t \in T_{ij}^{t}} x_{i,j,t} \cdot CH_{i,j})\) is greater than \(WT_{check}\) then \(rb_{wd,j} = 1\). Otherwise, \(rb_{wd,j} = 0\). Suppose that the working hours for the two groups, \(i\) and \(i + 1\), is less than \(WT_{check}\) then \(rb_{wd,j} = 0\). According to constraint (14), if \(rb_{wd,j} = 1\) based on constraints (12) and (13), the break time between groups \(i\) and \(i + 1\) \((\sum_{t \in T_{ij}^{t}} y_{i,j,t} \cdot WH_{i} + \sum_{t \in T_{ij}^{t}} x_{i,j,t} \cdot CH_{i,j})\) should be greater than \(WT_{break}\).

Consequently, the end time of the duty in day \(d + 1\) \((\sum_{t \in T_{ij}^{t}} y_{i,j,t} \cdot WH_{i} + \sum_{t \in T_{ij}^{t}} x_{i,j,t} \cdot CH_{i,j} + \sum_{d} \sum_{i,j} y_{i,j,t} \cdot CH_{i,j})\) exceeds \(WT_{check}\) then \(rb_{wd,j} = 0\). According to constraint (14), if \(rb_{wd,j} = 1\) based on constraints (12) and (13), the break time between groups \(i\) and \(i + 1\) \((\sum_{t \in T_{ij}^{t}} y_{i,j,t} \cdot WH_{i} + \sum_{t \in T_{ij}^{t}} x_{i,j,t} \cdot CH_{i,j} + \sum_{d} \sum_{i,j} y_{i,j,t} \cdot CH_{i,j})\) should be greater than \(WT_{break}\). Constraint (15) ensures that the rest hours between two consecutive duties is sufficient. Specifically, the start time of the duty in day \(d\) \((\sum_{i,j} y_{i,j,t} \cdot (WT_{t} - CH_{depot} - PT_{e})\)) minus the end time of the duty in day \(d\) \((\sum_{i,j} y_{i,j,t} \cdot (WT_{t} + CH_{t} + PT_{e})\)) should be greater than \(R_{day}\). The maximum weekly working hours are enforced by constraint (16). Namely, the total weekly working hours \((\sum_{i,j} \sum_{t} y_{i,j,t} \cdot WH_{i} + \sum_{i,j} \sum_{t} y_{i,j,t} \cdot CH_{i,j} + \sum_{d} \sum_{i,j} y_{i,j,t} \cdot CH_{i,j})\) should not exceed \(WH_{max week}\). Constraints (17)–(20) ensure that the decision variables are binary integers.

**Proposition 1.** The integrated model has at least NP-hard complexity.

**Proof.** The crew scheduling problem is a set covering problem, which is known to be NP-hard [37]. Since the integrated model discussed in this paper considers the constraints of both the crew scheduling and crew rostering problems, the complexity of this model is at least NP-hard.

As the formulation is difficult to solve, to effectively and efficiently solve the proposed model, this paper devises a BPC algorithm in the following section.

4. The Branch-and-Price-and-Cut Algorithm

As there can be numerous columns in a large integrated problem, most of the columns will not be used in the optimal solution. To reduce the computational effort, this paper forms the restricted master problem (RMP), which is a set-covering problem that only contains a subset of feasible columns. In addition, the variables are temporarily relaxed to make the RMP easier to solve. After solving the RMP, the dual variables of the set-covering constraint are obtained. The pricing program then finds a new column based on the
dual variables from the RMP. If there exists a column with a negative reduced cost, this column is added into the RMP. If there is no such column, the optimal solution of the RMP is obtained. The interaction of the RMP and the pricing program is shown in Figure 3.

After obtaining the optimal solution of the RMP, it is examined whether all the decision variables are integer. If it is not, the branch-and-bound (BB) method is applied to the noninteger decision variable. Equivalently, each node of the BB tree corresponds to a column generation process. From the preliminary computational results, cuts that can trim the infeasible branch are necessary to accelerate the solution procedure. If the solution of the RMP is integer, the algorithm is terminated and reports the incumbent solution.

4.1. Initial Duties. To start the BPC algorithm, an initial solution must be provided. In this paper, a heuristic method is proposed to generate the initial solution and the corresponding duties.

In the heuristic, first a network is constructed and contains a source node and a sink node. The other nodes in the network are the tasks that must be assigned in a duty. An arc connecting two nodes represents that these two tasks can be serviced consecutively. For instance, if an arc can connect tasks 1 and 3, the time (i.e., the end time of task 1 is earlier than the start time of task 3) and location (i.e., the end stop of task 1 is the same as the start stop of task 3) constraints of the two tasks are satisfied.

Each node in the network has three labels: time, predecessor, and distance. Each arc has two properties: time and cost. The arc time is the time required to connect these two tasks. The arc cost is a value that can be modified with the dual variable passed from the RMP, which will be detailed in Section 4.3.

Suppose that the time of a path does not exceed a driver’s maximum daily working hours, then the path beginning from the source and terminating at the sink node represents a duty that can be executed by that bus driver. In the network design, the cost of that path corresponds to the reduced cost of the duty. To find the duty/column that can benefit the RMP (i.e., with a negative reduced cost), this paper develops a constrained shortest path algorithm.

The algorithm is a label-correcting-based algorithm with multiple labels, based on Dial’s bucket design [38] and the algorithmic design by Lin [39]. Figure 4 is used as an example to illustrate the algorithm.

In this network, time buckets are established for each node, numbered from 0 to WH in bucket. Bucket t contains the nodes with temporary time labels equal to t. Then, the buckets are checked sequentially until the first nonempty bucket is found. During the scanning process, the current node in the first nonempty bucket has the minimum distance label. Then, the tentative distance and time are calculated according to this current node. For instance, node 3 can reach node 5 in the network. Between these two tasks, the arc cost is 6 and the arc time is 2. The first nonempty bucket of node 3 is bucket 1. For node 5, the tentative distance is 11 (5 + 6 = 11), and the tentative time is 3 (1 + 2 = 3) from node 3 in bucket 1. Then, three constraints must be checked. First, if node 5 in bucket 3 was previously visited, the distance must be greater than the tentative distance if the label is to be updated. Second, the tentative
4.2. Restricted Master Program (RMP). After obtaining a set of feasible duties from the algorithm in Section 4.1, the RMP is used to generate a roster from the subset of duties. In the RMP, the formulation considers the set covering and the rostering constraints. Note that as only the feasible duties are generated from the pricing program, the decision variable, \( x_{w,d}^{wd} \), collapses to \( x_{c}^{wd} \) in the RMP. That is, if \( x_{c}^{wd} = 1 \), duty \( c \) is assigned to day \( d \) in week \( w \). Otherwise, \( x_{c}^{wd} = 0 \). The additional parameters used are as follows. \( I_{c,t} \) indicates whether task \( t \) is contained in duty \( c \), \( W T_{c}^{s} \) and \( W T_{c}^{e} \) represent the start time and the end time of duty \( c \), respectively, and \( WH \) denotes the working hours of duty \( c \).

With the notations mentioned above, the formulation of the RMP represented below can be simplified from the integrated model described in Section 3.3:}

\[
\begin{align*}
\text{Min} & \quad \sum_{w} \sum_{d} \sum_{c \in C} P_{wd} \times x_{c}^{wd} , \\
\text{subject to} & \quad \sum_{c \in C} x_{c}^{wd} = 1, \quad \forall w, \forall d , \quad (22) \\
& \quad \sum_{w} \sum_{d} \sum_{c \in C} I_{c,t} \times x_{c}^{wd} \geq 1, \quad \forall t \in T^{w}, \quad (23)
\end{align*}
\]

\[\begin{align*}
\sum_{w} \sum_{d} \sum_{c \in C} P_{wd} x_{c}^{wd} - \sum_{w} \sum_{d} \sum_{c \in C} \pi_{c}^{wd} \geq R_{day}, \\
& \quad \forall w, \forall d = 1, \ldots, d - 1 , \quad (24)
\end{align*}\]

\[\begin{align*}
\sum_{d} \sum_{c \in C} W H_{c} \times x_{c}^{wd} \leq W H_{\text{max}}^{\text{week}}, \quad \forall w , \quad (25)
\end{align*}\]

\[\begin{align*}
& \quad x_{c}^{wd} \in \{0, 1\}, \quad \forall w, \forall d, \forall c. \quad (26)
\end{align*}\]

In the simplified model, the objective function (21) is equivalent to equation (1) which aims to minimize the number of drivers required for the roster. Constraint (22) ensures that there is a duty assigned to day \( d \) in week \( w \). Constraint (23) is the set covering constraint that ensures that all the tasks will be covered in the roster. Constraint (15) can be simplified as constraint (24) that ensures the rest hours between two consecutive duties. The maximum weekly working hours’ constraint (16) is reduced to constraint (25). The decision variable is a binary integer, as indicated in constraint (26).

Because the RMP generates a roster based on a subset of duties, the pricing program described in Section 4.3 is used to find the duties that can improve the quality of the roster in the RMP.

4.3. Pricing Program. When identifying the column set that is beneficial to the RMP, it is essential to implicitly price out all the columns that are not in the set to ensure that the optimum has been found\cite{40}. This goal can be achieved via the reduced cost estimated in the pricing program. If duties/columns with negative reduced costs are identified, these duties/columns are added to the duty/column subset of the RMP and then the RMP is resolved.

Let \( \pi \) be the dual variables associated with constraints (22)–(25) (i.e., \( \pi_{ab}^{22} \) is the dual variable associated with constraint (22)). The dual problem of the RMP is as follows:

\[
\begin{align*}
\text{Max} & \quad \sum_{w} \sum_{d=1}^{W} \sum_{b=1}^{D} \pi_{ab}^{22} + \sum_{t \in T^{w}} \pi_{t}^{23} + (R_{\text{day}} - \text{Day}) \times \sum_{w} \sum_{d=1}^{W} \sum_{b=1}^{D-1} \pi_{ab}^{24} \\
& \quad + \sum_{d=1}^{W} \sum_{c \in C} W H_{c}^{\text{max}} \times \pi_{n_{c}}^{25}, \\
\text{subject to} & \quad \sum_{(a,b) \in (w,d)} \pi_{ab}^{22} + \sum_{t \in T^{w}} I_{c,t} \times \pi_{t}^{23} + \sum_{(a,b) \in (w,d-1)} W T_{c}^{s} \times \pi_{ab}^{24} \\
& \quad + \sum_{(a,b) \in (w,d)} (-W T_{c}^{e}) \times \pi_{ab}^{24} + \sum_{w} \sum_{c \in C} W H_{c} \times \pi_{n_{c}}^{25} \leq P_{wd}, \quad \forall c. \quad (28)
\end{align*}
\]

In the RMP, the reduced cost of the corresponding variable, \( x_{w,d,c} \), is
\begin{equation}
P_{u,d} = \sum_{(a,b)\in\{u,d\}} \pi_{a,b}^{22} + \sum_{t \in T^u} I_{c,t} \times \pi_{a,t}^{23} - \sum_{(a,b)\in\{u,d\}} WT_c^u \times \pi_{a,b}^{24} - \sum_{(a,b)\in\{u,d\}} (-WT_c^d) \times \pi_{a,b}^{24} - \sum_{\alpha} WH_{\alpha} \times \pi_{a}^{25}.
\end{equation}

(29)

Specifically, the RMP is composed of a set covering constraint (constraint (23)) and rostering constraints (constraints (22), (24), and (25)). As the proposed algorithm attempts to generate the feasible duties/columns from the pricing program to improve the objective function of the RMP, the rostering constraints are left in the RMP and the feasibility of the generated duties/columns are ensured in the pricing program. Therefore, the pricing program only considers the dual variable associated with the set covering constraint (23), and the objective function to find a duty/column with a negative reduced cost can be reduced to

\begin{equation}
\text{Min} - \sum_{t \in T^u} I_{c,t} \times \pi_{a,t}^{23}.
\end{equation}

(30)

Because the parameter \( I_{c,t} \) indicates whether task \( t \) is contained in duty \( c \), it can be replaced as \( \sum_{j} x_{i,j,t} \). Note that since the week and day will be assigned in the RMP, the decision variable \( x_{i,j,t} \) collapses to \( x_{i,j,t} \) in the pricing program. If \( x_{i,j,t} = 1 \), then task \( t \) is assigned to cell \( j \) of group \( i \). Due to this transformation, the mathematical formulation can be modified as follows:

\begin{equation}
\text{Min} \sum_{i} \sum_{j} \sum_{t \in T^u} (-\pi_{a,t}^{23}) \times x_{i,j,t},
\end{equation}

(31)

subject to

\begin{equation}
x_{i,j,t} + x_{i,j,t} - 2y_{i,t}^{j} \geq 0, \quad \forall i, \forall j = 1, \ldots, j - 1, \forall t_1 \in T, \forall t_2 \in T,
\end{equation}

(32)

\begin{equation}
x_{i,j,t} + x_{i,j,t} - 2y_{i,t}^{j} \leq 1, \quad \forall i, \forall j = 1, \ldots, j - 1, \forall t_1 \in T, \forall t_2 \in T,
\end{equation}

(33)

\begin{equation}
\sum_{t \in T} y_{i,t}^{j} = 1, \quad \forall i, \forall j = 1, \ldots, j - 1,
\end{equation}

(34)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} = 1, \quad \forall i, \forall j = 1, \ldots, j - 1,
\end{equation}

(35)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} + x_{i+1,j,t} - 2z_{i,t}^{j} \geq 0, \quad \forall i = 1, \ldots, i - 1,
\end{equation}

(36)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} + x_{i+1,j,t} - 2z_{i,t}^{j} \leq 1, \quad \forall i = 1, \ldots, i - 1,
\end{equation}

(37)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} + x_{i+1,j,t} - 2z_{i,t}^{j} = 1, \quad \forall i = 1, \ldots, i - 1,
\end{equation}

(38)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} + \sum_{w_{i}} y_{i,t}^{j} + \sum_{w_{i}} y_{i,t}^{j} \leq \sum_{w_{i}} y_{i,t}^{j} + \sum_{w_{i}} y_{i,t}^{j},
\end{equation}

(39)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} \times CH_{t_2,t_2} \leq \sum_{w_{i}} y_{i,t}^{j} \times WT_{check}, \quad \forall i = 1, \ldots, i - 1,
\end{equation}

(40)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} \times CH_{t_2,t_2} \geq \sum_{w_{i}} y_{i,t}^{j} \times WT_{check}, \quad \forall i = 1, \ldots, i - 1,
\end{equation}

(41)

\begin{equation}
\sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} \times WT_{check} - \sum_{j=1 \ldots j-1} \sum_{t \in T} y_{i,t}^{j} \times WT_{check}, \quad \forall i = 1, \ldots, i - 1,
\end{equation}

(42)

\begin{equation}
x_{i,j,t} \in \{0,1\}, \quad \forall i, \forall j, \forall t \in T,
\end{equation}

(43)

\begin{equation}
y_{i,t}^{j} \in \{0,1\}, \quad \forall i, \forall j = 1, \ldots, j - 1, \forall t_1 \in T, \forall t_2 \in T,
\end{equation}

(44)

\begin{equation}
z_{i,t}^{j} \in \{0,1\}, \quad \forall i = 1, \ldots, i - 1, \forall t_1 \in T, \forall t_2 \in T,
\end{equation}

(45)

\begin{equation}
rb_{i} \in \{0,1\}, \quad \forall i = 1, \ldots, i.
\end{equation}

(46)

The objective function (31) is to generate a duty/column that has a negative reduced cost based on the dual variable associated with constraint (23). Constraint (32) ensures that each cell \( j \) of group \( i \) has been assigned a task. The constraint
is equivalent to constraint (2). Constraints (33)–(38) are similar to constraints (4)–(9). As the indices of the week and day can be ignored in the pricing program, these two indices are deleted for the decision variables. Constraint (39) ensures that the maximum daily working hours for the duty generated from the pricing program is not violated. The constraint is equivalent to constraint (10). Constraints (40)–(43), which are comparable to constraints (11)–(14), ensure that a driver has sufficient break time between tasks. Constraints (44)–(47) specify that the decision variables are binary integers.

Next, the pricing program is transformed into a network problem. Using the network depicted in Figure 5 as an explanatory example, the arc time is the time required to connect two nodes (represented as CH\textsubscript{ij} in this example), and the arc cost is associated with the dual variable passed from the RMP. The cost is set as $-\pi_2$ according to the head node $t$. For instance, the arc cost between node 3 and node 7 is $-\pi_2$. The arc time is represented as CH\textsubscript{37} in this example.

This network problem can be solved with the constrained shortest path algorithm, which is similar to the algorithm described in Section 4.1. The flowchart is shown in Figure 6. First, a time bucket set is defined as Time that ranges from 00:00 to 24:00. Let CH\textsubscript{ij} be the time label of arc $(i, j)$, $-\pi_2$ be the cost label of arc $(i, j)$, $T$ be the maximum daily working hours, $S$ be the source, and $A(i)$ is the set that includes the arcs emanating from node $i$. Initially, node $S$ is given a distance label of zero, and each other node $i$ a distance label equal to $\infty$. For the predecessor indices, node $S$ is set as 0, and each other node is set as $-1$ in each Time bucket. Then, node $S$ at time 0 is added into the Queue list. At each iteration, the first node $i_1$ and its time label $t_1$ are taken from the Queue list and the arcs in $A(i)$ is scanned to reach its adjacent nodes, $j \in A(i)$. The procedure then checks whether the current labels for nodes $i$ and $j$ satisfy the conditions, $t_1 + CH_{ij} \leq T$ and $dist(i_1, t_1) - \pi_2 \leq dist(j, t_1 + CH_{ij})$. If these conditions are satisfied, node $j$ and its time label $t_1 + CH_{ij}$ are added into the Path list and the predecessor indices are used to trace back the path. If each node in the path has a minimum break time $WT_{break}$ after continuously working for $WT_{check}$ hours, the node $j$ and the time label $t_1 + CH_{ij}$ are added into the Queue list, and its distance label and the predecessor indices are renewed as $dist(j, t_1 + CH_{ij})$ and $prep(j, t_1 + CH_{ij})$, respectively. In other words, the abovementioned procedure checks whether the connection violates the labor laws. After all the arcs in $A(i)$ have been scanned, node $(i_1, t_1)$ is deleted from the Queue list. The algorithm terminates when the Queue list is empty. At termination, the time buckets of the end node are scanned. Because of the limitation of the maximum daily working hours, this procedure only considers the time buckets smaller than $T$. After finding the minimum distance label from the time buckets, the predecessor indices are used to trace back the shortest path. With this constrained shortest path algorithm, a feasible duty/column can be generated based on the dual variables for the RMP.

4.4. Cut Generation. In this section, the valid cuts are developed according to the roster constraints in the RMP that trim the infeasible solutions for the enumeration tree so that the BPC algorithm process can be accelerated.

4.4.1. Cut I: Avoid Multiple Duties in One Day. Each driver can only execute one duty in a day. Constraint (22) together with the binary decision variable $x_{c,n}^{wd}$ is designed to ensure that, therefore, if $x_{c,n}^{wd} = 1$, duty $c$ is assigned to day $d$ in week $w$ at the $n^{th}$ iteration. Cut (48) can be added at the $n^{th}$ iteration. Note that the cuts are added to the RMP based on the given decision variable $x_{c,n}^{wd}$ at the $n^{th}$ iteration so that the cuts will trim the infeasible solutions at the $n^{th}$ iteration ($\forall n^{th} > n^{th}$):

$$\sum_{c \in C_C} x_{c,n}^{wd} = 1 - \sum_{c} x_{c,n}^{wd}, \quad \forall w, \forall d, \forall n^{th}.$$  \(48\)

Cut (48) ensures that if the duty $c$ is assigned to the day $d$ in the week $w$ at the $n^{th}$ iteration ($1 - \sum_{c} x_{c,n}^{wd} = 0$), then no duty can be assigned to the day $d$ in the week $w$ at the $n^{th}$ iteration $(\sum_{c} x_{c,n}^{wd} = 0)$.

4.4.2. Cut II: Avoid Infeasible Time Connections between Duties. Based on the labor laws, the drivers must have enough rest hours between two consecutive duties. In other words, if the rest hours between two duties are less than $R_{day}$, these two duties cannot be executed sequentially. That is, if $x_{c,n}^{wd} = 1$, the end time of the duty $c$ in day $d$, week $w$, is...
where $\pi_j$ is the path from node $i$ to node $j$. The value of the variable $x_{c,n}^{w,d}$ is constrained as follows:

$$\sum_{c' \in C_c} W_{c'} \times x_{c',n}^{w,d+1} - W_{c'}^b < R_{\text{day}},$$

(50)

That is, in the enumeration tree, when $x_{c,n}^{w,d} = 1$, the end time of the duty $c$ in day $d$ of week $w$ is $W_{c'}^b$. Cut (50) ensures that if $W_{c'}^b - W_{c'}^b < R_{\text{day}}$, the tree node of $x_{c,n}^{w,d+1}$ and its following leaf nodes can be trimmed.

**Proposition 2.** Cuts I-II do not exclude the optimal solution.

Proof. A solution set is defined as $S$ that includes all the solutions of the problem, and the optimal solution of the problem is assumed as $S^* \in S$. Let cuts I-II exclude $S^*$. Since $S^*$ is an infeasible solution excluded by the cuts, it cannot be an optimal solution. This relation contradicts the assumption. \(\square\)

**Proposition 3.** The BPC algorithm can find the optimal solution in a finite number of iterations.

Proof. At each iteration, cuts I-II are added into the master problem to reduce the solution space. According to Proposition 2, cuts I-II do not exclude the optimal solution. Since the solution space is finite, the BPC algorithm can find the optimal solution in a finite number of iterations. \(\square\)

5. **Empirical Study**

To validate the efficacy and evaluate the performance of the proposed BPC algorithm, the BPC is empirically applied to solve various sizes of the integrated problem. For validation, the commercial optimization package Gurobi® 8.0.1 is used to solve the integrated model. The results from the Gurobi® software can be considered as the optimal solutions and are assumed as $S^* \in S$. Since $S^*$ is an infeasible solution excluded by the cuts, it cannot be an optimal solution. This relation contradicts the assumption. All the procedures are implemented in the C# programming language and carried out on a Windows-based machine with an Intel 3.4 GHz CPU processor and 64 GB of memory.

5.1. **Validation and Performance Comparison.** The first experiment solves 6 real-world problem instances with various number of tasks to be covered, and the information of the case study data is summarized in Table 1. The location of the study area is Taichung city, Taiwan, and the data are collected from a portion of the network that is currently operated by the H Bus Company. In the testing data, the number of assigning tasks varies from 18 to 196 and the number of bus routes ranges between 1 and 18. The bus map of Taichung City is shown in Figure 7, which is provided by the Taichung City Government Bureau of Transportation [41]. The data includes the starting stop, ending stop, starting time, ending time, belonging bus route, and depot station for each task.
Table 2 summarizes the computational results of the BPC algorithm in comparison with those of the Gurobi® package.

For the smallest problem instance (instance A), both the Gurobi® package and the BPC algorithm can determine the optimal solution. However, the CPU time required by the BPC algorithm is lower than that of the Gurobi®. Figures 8 and 9 depict the resulting schedule and roster from both solution approaches. As seen from the results, although the objective value is identical, the scheduling is slightly different, indicating that the problem can have multiple optimal solutions.

In instance B, the BPC algorithm can still find the solution with the identical objective value of 4 with less computational effort. However, the duty required by the BPC algorithm is 1 more than that of the Gurobi® package. Because the minimization of the number of drivers is the objective, the two solutions are in fact equally good mathematically. The only difference is that the drivers in the Gurobi® solution can have a day-off in the roster, but the working hours per day is higher. On the other hand, the drivers in BPC solution can have fewer working hours per day; however, they cannot have a day-off within the roster.

As the problem size grows (instances C, D, E, and F), the number of decision variables and constraints grows exponentially. Therefore, the Gurobi® package fails to identify any feasible solutions within 24 hours of CPU time. However, the proposed BPC algorithm can still find the optimal solutions in slightly over one hour, even for the largest problem instance, indicating that the proposed BPC algorithm is efficient.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Number of tasks</th>
<th>Number of bus routes</th>
<th>Bus route</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>18</td>
<td>1</td>
<td>Orange route</td>
</tr>
<tr>
<td>B</td>
<td>24</td>
<td>1</td>
<td>Brown route</td>
</tr>
<tr>
<td>C</td>
<td>52</td>
<td>1</td>
<td>Blue route</td>
</tr>
<tr>
<td>D</td>
<td>114</td>
<td>11</td>
<td>Green, G12-13, and G20-27 routes</td>
</tr>
<tr>
<td>E</td>
<td>154</td>
<td>13</td>
<td>B1-3, B10-13, and B20-25 routes</td>
</tr>
<tr>
<td>F</td>
<td>196</td>
<td>18</td>
<td>Green and G1-17 routes</td>
</tr>
</tbody>
</table>

Figure 7: The bus map of Tainan City [41].
Table 2: The comparison between the Gurobi® package and the BPC algorithm.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Duty¹</th>
<th>Driver²</th>
<th>CPU (seconds)</th>
<th>Duty*</th>
<th>Driver*</th>
<th>CPU (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>5</td>
<td>9.05</td>
<td>5</td>
<td>5</td>
<td>8.54</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>4</td>
<td>35.25</td>
<td>4</td>
<td>4</td>
<td>10.76</td>
</tr>
<tr>
<td>C</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>12</td>
<td>13</td>
<td>56.29</td>
</tr>
<tr>
<td>D</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>9</td>
<td>13</td>
<td>306.63</td>
</tr>
<tr>
<td>E</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>14</td>
<td>16</td>
<td>324.09</td>
</tr>
<tr>
<td>F</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>24</td>
<td>30</td>
<td>3,778.93</td>
</tr>
</tbody>
</table>

*The number of duties required to cover the timetable. **The number of drivers required for the roster.

Figure 8: The scheduling of instance A from the Gurobi® package and the BPC algorithm.

Figure 9: The rostering of instance A from the Gurobi® package and the BPC algorithm.
5.2. Sensitivity Analysis. In this section, the sensitivity analyses of the parameters of the integrated problem with instance A are conducted so that insights can be gained from the results. First, the parameter for the maximum daily working hours is perturbed and the results are summarized in Figure 10.

Only when the maximum daily working hours increase to 17 hours will the number of duties decrease. However, the number of drivers required for the roster remains the same, meaning that increasing the maximum number of working hours does not have a tangible effect on the objective value. When the maximum daily working hours increase to 17 hours a day, there are more tasks that can be assigned in each duty. However, the drivers still need enough rest hours between the duties. If the daily working hours on day \( d \) is too high, a leave duty on day \( d + 1 \) has to be assigned for enough rest hours. Consequently, even if the number of duties decreases, the number of drivers required to complete the roster remains the same.

In this case, to have one more day-off, the maximum daily working hours must be increased to 17 hours. Within the roster, the waiting time is almost 4 hours per day for each driver. It should be emphasized that the waiting time here is actually idle time for each driver (i.e., the time other than the required rest time specified by the labor laws). In other words, each driver has more time to rest than the time regulated by the laws. Therefore, it is not entirely infeasible to operate such rosters since the drivers may still have enough rest. For instance, if there are any unexpected events, a bus company may run such roster just for emergency situations. On the other hand, as the maximum daily working hours decrease, the number of duties increases and the roster requires more drivers to service it.

Next, a sensitivity analysis on the maximum weekly working hours is conducted and the results are presented in Figure 11.

Interestingly, compared to the results shown in Figure 10, the maximum daily working hours have a more critical impact on the roster than the maximum weekly working hours.

Note that some interesting differences between the results of this paper and the results from other studies are found. For instance, in the railway crew management problem, Lin and Tsai [25] found that the roster/schedule is more sensitive to the maximum weekly working hours, instead of the maximum daily working hours. The reason is the data differences. Since the working hours of a single railway task is typically longer, it lacks flexibility when building a daily duty. In contrast, the working hours of a bus task are typically shorter. Therefore, when the maximum daily working hours are adjusted, the number of task combinations is greater for bus scheduling/rostering. As a result, a bus scheduling/rostering process is more sensitive to the maximum daily working hours.

In the analysis of Figure 12, when the rest hours are adjusted, the number of duties remains identical in all the cases. The roster requires the same number of drivers to complete it when the rest hours decrease. However, when the rest hours increase, the number of drivers in the roster increases since more rest hours are required between the two duties.

For comparing the difference between the results of the integrated model and the two-phase approach, in the following experiment, the integrated model solves the driver scheduling and rostering problem simultaneously and obtains the optimal schedule (duties) and roster (drivers). In the two-phase model, the driver scheduling problem is first solved and then the rostering problem generates the solution based on the duties obtained from the scheduling. The comparison of the results from both models is shown in Figure 13.

In this experiment, the results show that when the rest hours are adjusted, the numbers of duties for the both models remain identical in all the cases. Furthermore, when
the rest hours increase to 12, both the integrated and two-phase models require an identical number of drivers for the roster. When the rest hours increase to 13, the integrated model only requires 6 drivers to complete the roster, while the number of drivers is 7 for the two-phase model. This observation can only be found when the scheduling and the rostering problem are integrated in a model, which shows the advantage of the integration in this study.

In Figure 14, this experiment conducts the sensitive analysis of the minimum number of minutes to be considered as a break. If the time between tasks is less than this number, it should not be considered as the break time. In this case study, a time of more than 15 minutes between tasks is considered as a break. As can be expected, when this number increases, fewer tasks can be assigned in a duty. When this number increases to 20 and 25 minutes, the number of duties increases to 6 from 5, and the roster requires 6 drivers to complete it.

In contrast, when this number decreases, the number of duties does not decrease. The managerial insight of these results is that the decrease in this number does not improve the solution and hence is not beneficial for a bus company. Furthermore, the drivers cannot have adequate rest in such solutions. Therefore, the reduction of this number can result in a lose-lose situation.

6. Conclusions

This research investigates the integrated bus driver scheduling and rostering problem, which is formulated as a mixed integer programming problem. To solve this problem efficiently, this paper develops a BPC algorithm that generates the columns on-the-fly, identifies an integer solution using a branch-and-bound framework, and trims the infeasible solutions with cuts. The proposed solution scheme is empirically applied to real-world instances from a portion of the bus network in Tainan city, Taiwan.

In the numerical experiments, the results show that (1) the scheduling and rostering results of the bus drivers are more sensitive to the maximum daily working hours compared to the maximum weekly working hours. (2) When the maximum daily working hours increase, there can be an additional day-off. However, the idle time for each driver can increase. (3) When the rest hours increase, the number of drivers in the roster also increases. However, the number of duties remains the same. (4) A decrease of the minimum number of minutes to be considered as a break can end up considered as a break. If the time between tasks is less than this number, it should not be considered as the break time.

In this case study, a time of more than 15 minutes between tasks is considered as a break. As can be expected, when this number increases, fewer tasks can be assigned in a duty. When this number increases to 20 and 25 minutes, the number of duties increases to 6 from 5, and the roster requires 6 drivers to complete it.

In contrast, when this number decreases, the number of duties does not decrease. The managerial insight of these results is that the decrease in this number does not improve the solution and hence is not beneficial for a bus company. Furthermore, the drivers cannot have adequate rest in such solutions. Therefore, the reduction of this number can result in a lose-lose situation.
with a lose-lose situation for both the bus company and drivers. (5) The integrated formulation yields a better solution than the solution from decomposing the scheduling and rostering into two phases. The empirical studies show that the proposed BPC algorithm can identify solutions that are comparable to the optimal solutions from a benchmark optimization solver in smaller cases. In larger problem instances, the proposed solution scheme can solve the problems that are beyond the reach of the benchmark optimization solver within reasonable computational times.

Although promising results are obtained, this research can be extended in many ways. First, many practical scheduling/rostering problems need to take driver preferences into consideration. To incorporate this type of consideration, the decision variables, parameters, and model constraints must be modified. Second, when unexpected disturbances occur, recovering a schedule or roster to a stable condition is an important task. Further research can analyse this task on the basis of the current work. Finally, stochastic incidents are always possible in the real world. Future research may extend the current work by applying stochastic or robust optimization techniques so that the schedule/roster is more resilient to stochastic events.

Data Availability

The scheduling/rostering data used to support the findings of this study were supplied by SN Bus Company under license and so cannot be made freely available. Requests for access to these data should be made to Chieh-Ju Juan (r58061021@mail.ncku.edu.tw).

Conflicts of Interest

There are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Ministry of Science and Technology, Taiwan, ROC under Grant numbers MOST 105-2628-H-007-003-MY3 and MOST 108-2410-H-007-097-MY4.

References


