Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 10, No. 1, pp. 113–136, January 2014

A COLUMN GENERATION ALGORITHM FOR CREW SCHEDULING WITH MULTIPLE ADDITIONAL CONSTRAINTS*

Yin-dong Shen† and Shi-jun Chen

Abstract: Crew scheduling in public transport is to cut vehicle works to a set of feasible shifts which are implemented by crews or drivers. Most crew scheduling problems are often based on standard set covering formulations, which cannot meet some particular requirements imposed on shift generation, for example, the number or the percentage of a particular type of shifts must be within a predefined range. Taking the realistic characteristics of crew scheduling into account, an extended set covering formulation with three types of additional constraints is proposed to satisfy various specific requirements and constraints. A column generation algorithm is presented to solve the formulation. To deal with the difficulties brought by additional constraints, a decomposition strategy is proposed to solve the column generation subproblems, in which all shifts are reclassified into several sets of disjoint shifts according to the types of additional constraints, and each subproblem corresponds to the generation of one set of shifts in a directed graph. Moreover, a branching strategy based on relief opportunities is designed to get integer solutions. The proposed model and algorithms are tested on twelve instances derived from real-world problems. The experimental results show that the schedules satisfying to the transit operators' requirements are produced.

Key words: *public transport, crew scheduling, additional constraints, column generation*

Mathematics Subject Classification: *90B06, 90C08, 90C10, 90C90*

1 Introduction

Crew scheduling plays an important role in public transport. An efficient crew schedule can save an enormous amount of money for transit enterprises. The crew scheduling problem has attracted much research interest since 1960s. Up to now, a large number of scheduling models and sophisticated solution methods have been proposed, many of which have been reported in a series of international conferences on computer-aided scheduling for public transport, e.g. [19, 39].

Since crew scheduling problems in different areas (e.g. urban transit, railway and airline) have different characteristics, it needs to design appropriate formulations and algorithms to deal with each type of the problems. In this paper, our major concern is the crew scheduling problems of urban public transport. The objective of crew scheduling problems is to find the most efficient way to cut vehicle (bus) works into a set of shifts which are to be implemented by crews or drivers. The problems are very complex because many of

*This work was supported by the National Nature Science Foundation of China (No. 70971044, 71171087 and 61304175).
†Corresponding author

labor agreement rules and operational constraints are to be considered. As indicated by Laplagne [25], all labor rules and constraints can be regarded as two levels of constraints: shift-level and schedule-level. The shift-level constraints consist of a set of labor agreement rules and other regulations which govern the legality of a shift. Different types of shifts have their own corresponding constraints. The schedule-level constraints represent restrictions imposed on the final solution of the problems (i.e. schedule consisting of a set of shifts). For crew scheduling problems, an indispensible schedule-level constraint is that each piece of vehicle work must be covered by at least one shift in the schedule. Consequently, they can be naturally represented as a set covering formulation (SCF), as usually done in previous literatures [21, 24, 26, 30, 32, 34].

In the SCF, each binary variable means whether a shift is selected from a large set of shifts into a solution. Although a standard SCF has been extensively used to solve crew scheduling problems [8, 22, 23, 29, 30, 40], the solutions obtained may not be applicable due to the following two reasons. One reason is that the standard SCF does not consider some practical operational constraints. For example, the number of a type of shifts that must be executed by the crews with particular skills may exceed available crews, so the solution cannot be implemented in practice. A typical issue arising in the crew scheduling problems with multiple crew bases is that the number of shifts assigned for a crew base may exceed corresponding available crews. Some researches proposed base constraints to restrict the number of shifts for each crew base [1, 7, 21, 34]. Other similar schedule-level constraints have also been presented to reflect some operational properties in crew scheduling. The reader can be referred to Ernst et al. [12], Dias et al. [11], Portugal et al. [32] and Abbink at al. [3]. Another reason is that the standard SCF cannot reflect the transit operators' preferences or expectations about some special types of shifts. For example, transit operators may require that some specific type of shifts must take at least (or at most) certain proportion of all shifts in the schedule. Although the preferences can be handled by imposing a big penalty cost to the non-preferred shifts [2], it will spend much time to adjust the penalty costs such that the solutions fit for the operator's expectation. Moreover, setting the values of penalty costs often frustrates the human schedulers lacking optimization knowledge [32]. In many cases, transit operators' requirements are hard constraints that must be satisfied. Therefore, it is necessary to extend the standard SCF with the additional constraints to respect these requirements.

Up to now, only few literatures have addressed some of the additional constraints [2, 3, 11, 14, 15, 40]. In these researches, some works neglected the additional constraints or did not provide detail information for handling the additional constraints when solving the problems [14, 15, 40]. Other works detailed solving approaches to deal with additional constraints, but they are only applicable to the problems with specific features [2, 3, 11]. Consequently, it needs to develop new formulations and corresponding solving method to adapt to the changes on scenarios or specific requirements. Based on the previous works, this paper presents a more generic formulation with considering five typical types of additional constraints from the view point of practice. The five additional constraints can include most of additional constraints proposed in previous literatures as special cases.

As the SCF is well-known NP-hard [18], a large number of meta-heuristic solution approaches such as genetic algorithm [11], tabu search [35], ant colony optimization [33], and greedy randomized adaptive search procedure [27, 28] have been proposed. These approaches can solve the problems quickly, but the quality of solutions cannot be guaranteed. Moreover, the additional constraints will make the problem more complex and difficult to solve [11, 32]. In order to solve the proposed formulation, we design a column generation algorithm, which belongs to the exact approach family and is one of the most successful methods to solve various large-scale practical integer Linear programming (ILP) problems in transportation and industry, such as line planning [5], vehicle routing [10], crew scheduling problems [34], and production planning and scheduling in iron and steel industry [36, 37, 38]. For the reviews of column generation technique, the reader can refer to Barnhart et al. [4], Lubbecke and Desrosiers [31], Desaulniers et al. [9]. Column generation is an iterative improving method, in which a restricted master problem and a subproblem are solved in turn iteratively. When using the column generation to solve crew scheduling problems, the subproblem is called to generate new columns, i.e. the shifts with the least negative reduced cost to improve the solution of current restricted master problem. Under the standard SCF, one or more shifts with the least reduced cost can be generated by solving a resource constrained shortest past problem (RCSPP) in a directed graph. And the graph is constructed by all pieces of vehicle work to be covered [20]. However, the difficulty of solving the subproblem will be encountered when considering the SCF with additional constraints. This is mainly because a traditional formula of calculating the reduced cost of a shift cannot represent the additional constraints. Therefore, some new strategies have to be designed to deal with additional constraints. For crew scheduling problems with crew base constraints, the subproblem is usually partitioned into smaller subproblems, each of which is employed to generate shifts for one crew base [17, 34]. Inspired by this decomposition idea, we firstly reclassify all shifts into several sets of disjoint shifts according to the types of additional constraints proposed, and then design a more generic decomposition approach to solve the column generation subproblem based on reclassified set of shifts. The proposed approach can deal with multiple additional constraints in schedule-level. Moreover, some of the column generation based approaches dealing with additional constraints (e.g. crew base constraints) in previous literature can be regarded as its special cases.

Finally, in order to get integer solutions we design a branching strategy based on relief opportunities (ROs). First, a relief opportunity is selected according to a predefined rule (see Section 3.3) for the current node with fractional solutions. Then, the solution space is divided into two disjoint branches. One branch requires that the selected RO not be used and the other branch requires the selected RO must be used. When one fails to select a RO, we use a traditional branch-and-bound method (branching on fractional variables) to search integer solutions within the shift set that corresponds to the solution of current node.

The two major contributions of this paper are as follows. One is to build an extended set covering formulation while considering five typical specific requirements or constraints derived from real-world problems. Another is to design a generic column generation approach to solve the subproblem with multiple additional constraints.

The remaining sections are organized as follows. Section 2 presents an extended set covering formulation with five typical additional constraints after some basic notions about the crew scheduling problem is introduced. Section 3 describes our proposed column generation algorithm, decomposition approaches for solving the subproblem and a RO-based branching strategy for getting integer solutions. The experimental tests on real crew scheduling problems are displayed in Section 4 while the conclusions are drawn in Section 5.

2 Problem Description

2.1 Crew Scheduling Problem

To describe the crew scheduling problem, some definitions are first given as follows.

Assume the vehicle schedule is given in advance, which consists of a group of *vehicle blocks.* A *block* represents a set of vehicle tasks to be operated consecutively by one vehicle during a day, beginning with a *pull-out* from, and ending with a *pull-in* to, a depot. Usually it contains a sequence of predetermined *relief opportunities* (*ROs*) which can be used for crew's relieving, where a *relief opportunity* (*RO*) is formed by a *relief time* and a *relief point* (i.e. a time/location pair). Note that any RO may be used or not be used in the final solution.

In order to define the notion of *shift*, some other definitions are needed to be given. The vehicle task between any two adjacent ROs is defined as a *piece of work* (*piece* for short). One or more consecutive pieces contained in the same block form a *spell* of work. A shift is formed by one spell or several spells which are separated by one or more *breaks*. In addition, a shift generally starts working with a *sign-on* activity and finishes working with a *sign-off* activity. Figure 1 displays an example of two blocks in a vehicle schedule, where each dot denotes a relief opportunity and its corresponding relief point is coded by a single alphabet. For example, block 1 contains five ROs and their corresponding relief points are A, C, B, C, A. The means of piece, spells are also illustrated in Figure 1, where both spell 1 and spell 2 contain two pieces. Moreover, spell 1 and spell 2 can be linked to form a shift (if it satisfies the constraints described in the following context).

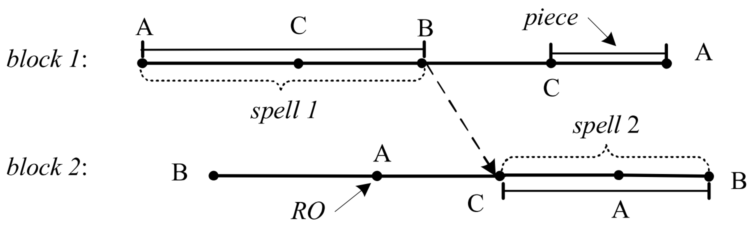

Figure 1: Vehicle work and crew shift

Notice that a feasible (or valid) shift must respect to a series of labor rules or regulations (i.e. shift-level constraints) according to union agreement. The content of these regulations may vary in different operations enterprises and may vary according to the shift type. In our context, according to the practical situations from Chinese public transport operations, we consider three types of shift, which are *straight* shift, *split* shift and *tripper* shift. A straight shift consists of two or three spells, and there must be has at least one meal break (or short break) between them. A split shift consists of two spells, and there must be has a long rest break between them. Tripper shift only contains one spell. A shift is valid if it satisfies the following seven shift-level constraints: (1) the length of any spell contained is no longer than a given value $Q^1_{\max}$; (2) the number of spells contained is no more than a given number $Q^2_{\max}$; (3) its driving time is no longer than a given value $Q^3_{\max}$; (4) its working time is no longer than a given value $Q^4_{\max}$; (5) its spread-over is no longer than a given value $Q^5_{\max}$; (6) its meal-break time is no shorter than a given value $Q^1_{\min}$ if it is a straight shift; (7) its long break time is no shorter than a given value $Q^2_{\min}$ if it is a split shift.

Thus, crew scheduling consists of generating a set of notional shifts (i.e. a crew schedule) such that:

(1) Each piece of work is assigned to a shift;

(2) All shifts must be subject to all labor rules and operational constraints;

(3) The total number of shifts and/or the total cost (payable work in minutes) is minimized.

2.2 Standard Set Covering Formulation

Generally, the crew scheduling problem can be represented as a set covering formulation. Let $M = \{1, 2, \ldots, m\}$ be the set of pieces and $N = \{1, 2, \ldots, n\}$ be the set of all valid shifts. The cost of shift j is denoted by d_j which can be calculated by some predefined computational methods. With each piece $i \in M$ and shift $j \in N$, we associate a binary parameter a_{ij} that takes value 1 if shift j covers piece i, and 0 otherwise. Finally, define a binary decision variable x_j for each shift $j \in N$ that indicates whether shift j is selected in a solution or not. Now, crew scheduling problem can be formulated as a set covering problem:

$$\text{Minimize} \sum_{j \in N} d_j x_j \tag{2.1}$$

$$\text{subject to}: \sum_{j \in N} a_{ij} x_j \geq 1, \forall i \in M \tag{2.2}$$

$$x_j \in \{0, 1\}, \forall j \in N. \tag{2.3}$$

The objective function (2.1) is to minimize the total cost of shifts in the final schedule. Constraints (2.2) ensure that each piece is covered by at least one shift. Constraints (2.3) define the decision variables as binary. In general, minimizing the number of crew shifts is more important than the total operational time. Hence, we calculate the d_j (called weighted cost) as follows:

$$d_j = W + w_j c_j \tag{2.4}$$

Where c_j is the payable working time (in minutes), W is a constant value (which is larger than $\max_{j \in N}\{w_j c_j\}$ and is set to 5000 in this context) and it ensures minimizing the number of shifts is the overriding objective, and w_j is the preference coefficient that is defined as:

$$w_j = \begin{cases} 1 & \text{If } j \text{ is a straight shift} \\ 1.2 & \text{If } j \text{ is a split shift} \\ 1.5 & \text{If } j \text{ is a tripper shift} \end{cases} \tag{2.5}$$

According to the practical situation, the cost c_j is calculated by the following way: for straight and tripper shifts, c_j is the sum of sign-on time, sign-off time, driving time and mealbreak time; for split shifts, c_j is the sum of sign-on time, sign-off time and driving time. For the cost d_j defined by (2.4), its two special cases are often appeared. If $d_j (j \in N)$ are set to 1, then the objective is only to minimize the number of shifts; If W is set to 0, the objective is only to minimize the payable working time.

2.3 Extended Set Covering Formulation with Multiple Additional Constraints

In this section, we first consider five typical additional constraints from the view point of practice. Then, we present an extended set covering formulation with multiple additional constraints.

2.3.1 Five Types of Additional Constraints

As indicated in Section 1, the single SCF may not be applicable when there are specific hard requirements or limits on the numbers/percentages for some particular types of shifts in crew schedules (e.g. a requirement is that the percentage of straight shifts has to be more than 85% in overall shifts of the schedule). Note that such requirements may vary from different countries or different scenarios and they are determined by the practical characteristics of the problems to be solved and transit operators' preferences. In this context, we consider five typical types of specific requirements from the viewpoint of general cases and some requirements are derived from the practice of Chinese urban public transport. Each type of the specific requirements is used to restrict some attributes (e.g. number, percentage) of one or several sets of shifts in crew schedules.

For the convenience of description, specific requirement is also called additional constraint in the following context. Now, we detail the five types of additional constraints by using some examples that may arise in real public transport.

(1) Transit operators may require that some preferred types of shifts should take more percentage (e.g. 80%) in all shifts of the schedule. For example, straight shifts, especially two-spell shifts (two spells contained) working more than six hours with a mealbreak in between, are preferred. Denote $N_{P_1} \subset N$ the preferable shift set, the number of N_{P_1} should take a percentage at least α in the crew schedules, which can be represented as:

$$\sum_{j \in N_{P_1}} x_j - \alpha \sum_{j \in N} x_j \geqslant 0 \tag{2.6}$$

(2) Some types of preferred shifts may be required that their numbers must be at least a given value. For the preferred shift set $N_{P2} \subset N$, its number is at least a given number u, which can be represented as:

$$\sum_{j \in N_{P2}} x_j - u \geqslant 0 \tag{2.7}$$

(3) Some non-preferred types of shifts may be required that their percentages should be equal or less than a given percentage (e.g. 10%) in all shifts of the schedule. For example, some non-preferred types of shifts may be tripper shifts, the shifts with the spread-over more than ten hours, the shifts with the number of spells more than three, etc. Denote $N_{UP1} \subset N$ the non-preferred shift set, the number of N_{UP1} should take a percentage at most β in shift set N, which can be represented as:

$$\sum_{j \in N_{UP1}} x_j - \beta \sum_{j \in N} x_j \leqslant 0 \tag{2.8}$$

(4) In a similar way, the number of some non-preferred types of shifts may be required to be no more than a given number. For the shift set $N_{UP2} \subset N$, its number is no more than a given number v, which can be represented as:

$$\sum_{j \in N_{UP2}} x_j - v \leqslant 0 \tag{2.9}$$

(5) In some situations, comparison-type requirements may be imposed to ensure that the number of some types of shifts should be more than (or less than) the others type of shifts. For example, the transit operators may require that the number of two-spell shifts be more than shifts with three spells. Given two shift set $N_{C1} \subset N$ and $N_{C2} \subset N$, the number of N_{C1} is at least γ times more than that of N_{C2}, which can be represented as:

$$\sum_{j \in N_{C1}} x_j - \gamma \sum_{j \in N_{C2}} x_j \geqslant 0 \tag{2.10}$$

Note that the five types of constraints described above can include most of the requirements or constraints presented in previous literatures as special cases. For example, the formula (2.7) can represent the constraints on the number of available crews (i.e. crew base constraints) that most frequently used in literatures [1, 3, 7, 34]. The formula (2.9) can represent the constraints on the maximum number of shifts that cover each piece of work [32]. In addition, by adding appropriate coefficients for some (or all) variables, the formulas (2.6) to (2.10) can represent the constraints on the duration or workload of shifts in crew schedules. For example, the formula (2.7) can represent the workload constraints presented by Ernst et al. [12] and the operational constraints addressed by Dias et al. [11]. The formula (2.9) can represent the constraints on the average duration of a set of shifts presented by Abbink at al. [3] and Freling et al. [16]. And the formula (2.10) can represent the constraints on comparison between the total amount of work on A-train and B-train [3]. In summary, most of the constraints on certain attributes of shifts (e.g. number, percentage, working time) can be represented by the above five types of constraints.

$\boxed{2.3.2}$ Extended Set Covering Formulation with Multiple Additional Constraints

In this section, we present an extended set covering formulation by taking into account the five types of constraints (2.6) to (2.10). Since the only difference between formula (2.6) and formula (2.8) is the inequality signs, they can be seen as the same type of constraints and be handled in the same way. In similar, formulas (2.7) and (2.9) can be seen as the same type of constraints. Consequently, we neglect the constraint (2.8) and (2.9) and only consider three types of constraints, i.e. (2.6), (2.7), and (2.10). In addition, for each type of additional constraints, several concrete constraints belonging to it may be required in practice. For example, for the type of constraint (2.6), the transit operators may require that the percentage of straight shifts take more than 80% in crew schedules, and also require that the percentage of two-spell shifts take more than 90%. Similarity, for each of the other two types of additional constraints (2.7) and (2.10), there may also exist several specific constraints that correspond to it. Therefore, each type of constraints (2.6), (2.7), and (2.10) corresponds to a set of additional constraints. Before describing the extended set covering formulation, some notations are firstly defined.

|S|, the cardinality of a given set S

$H = \{1, 2, \ldots, |H|\}$, a set of additional constraints belonging to the type of constraint (2.6)

$K = \{1, 2, \ldots, |K|\}$, a set of additional constraints belonging to the type of constraint (2.7)

$S = \{1, 2, \ldots, |S|\}$, a set of additional constraints belonging to the type of constraint (2.10)

$N^h_{P_1}$, a shift set that restricted by the additional constraint h, $h \in H$

N^k_{P2}, a shift set that restricted by the additional constraint k, $k \in K$

N^s_{C1}, the first shift set that restricted by the additional constraint s, $s \in S$

N^s_{C2}, the second shift set that restricted by the additional constraint s, $s \in S$

b_{hj}, a coefficient, which takes the value 1 if the shift $j \in N^h_{P_1}$, and 0 otherwise; $h \in H$, $j \in N$

e_{kj}, a coefficient, which takes the value 1 if the shift $j \in N^k_{P2}$, and 0 otherwise; $k \in K$, $j \in N$

f_{sj}, a coefficient, which takes the value 1 if the shift $j \in N^s_{C1}$, and 0 otherwise; $s \in S$, $j \in N$

g_{sj}, a coefficient, which takes the value 1 if the shift $j \in N_{C2}^{s}$, and 0 otherwise; $s \in S$, $j \in N$

An extended set covering formulation with three types of additional constraints can be represented as:

$$\text{Minimize } \sum_{j \in N} d_j x_j \tag{2.11}$$

$$\text{subject to : } \sum_{j \in N} a_{ij} x_j \geq 1, \forall i \in M \tag{2.12}$$

$$\sum_{j \in N} (b_{hj} - \alpha_h) x_j \geqslant 0, \forall h \in H \tag{2.13}$$

$$\sum_{j \in N} e_{kj} x_j \geqslant u_k, \forall k \in K \tag{2.14}$$

$$\sum_{j \in N} (f_{sj} - \gamma_{\mathrm{s}} g_{sj}) x_j \geqslant 0, \forall s \in S \tag{2.15}$$

$$x_j \in \{0,1\}, \forall j \in N. \tag{2.16}$$

Where α_h is a lower bound for the percentage of the shifts that is restricted by the additional constraint h, $h \in H$; u_k is a lower bound for the number of the shifts that is restricted by the additional constraint k, $k \in K$; γ_{s} is a constant. Since $|N|$ (i.e. the number of shifts) is usually considerable, the proposed formulation is solved by a column generation algorithm to be described in the next section.

3 Solution Approaches

3.1 Column Generation Algorithm

Column generation is a successful technique to solve large-scale (variables are considerable) linear programming (LP) problems, and it is usually embedded in a branch-and-bound framework to solve the original ILP formulation. For more details, the reader can refer to Desaulniers et al. [9]. The basic idea of column generation for crew scheduling is as follows. When using the column generation to solve the LP relaxation (called master problem, MP) of an original ILP, it only need to consider a serials of restricted master problems (RMP), each of which contains partial variables (columns) of MP. An initial RMP$_0$ can be constructed by using artificial or heuristic methods. At the r^{th} iteration, RMP$_r$ is firstly solved and the dual multipliers that correspond to the constraints of MP are obtained. Then, a subproblem (SP) with the objective function based on the dual multipliers is called to generate one or more new shifts with the least negative reduced costs. The new generated shifts are added into the RMP$_r$ and a new restricted master problem RMP$_{r+1}$ is formed. At the $r+1^{\text{th}}$ iteration, the procedure is similar to the r^{th} iteration, i.e., RMP$_{r+1}$ is firstly solved, then the subproblem SP$_{r+1}$ is solved. The procedure repeats iteratively until the subproblem cannot generate any new shifts with negative reduced costs. In this paper, the master problem MP corresponding to the extended SCF is formed by relaxing the binary variables of the formula (2.16). At the r^{th} iteration, let $\pi_i^r(\geqslant 0)$, $\theta_h^r(\geqslant 0)$, $\delta_k^r(\geqslant 0)$ and $\mu_s^r(\geqslant 0)$ denote the dual multipliers that correspond to the constraint (2.12), (2.13), (2.14) and (2.15) in RMP$_r$ respectively. For each shift $j \in N$, denote

$$F_j = \sum_{h \in H} (\alpha_h - b_{hj}) \theta_h^r - \sum_{k \in K} e_{kj} \delta_k^r + \sum_{s \in S} (\gamma_{\mathrm{s}} g_{sj} - f_{sj}) \mu_s^r \tag{3.1}$$

Then its reduced cost rc_j is calculated as:

$$rc_j = d_j - \sum_{i \in M} a_{ij}\pi_i^r + F_j \tag{3.2}$$

For simplicity, denote SP(D) the subproblem for a given shift set D, which is defined as:

$$p = \arg\ \min_{j \in D} \{rc_j | rc_j < 0\} \tag{3.3}$$

Denote $\bar{N} \subset N$ the set of shifts contained in RMP$_r$, the subproblem SP$_r$ can be represented as SP$(N \backslash \bar{N})$. Because the set of all feasible shifts N is not known in advance, the subproblem SP$(N \backslash \bar{N})$ cannot be solved directly. Therefore, it needs to generate the shift with the least negative reduced cost by constructive approach. Since the shift generated must satisfy all shift-level constraints described in Section 2.1, the subproblem SP$(N \backslash \bar{N})$ can be modeled as a resource constrained shortest path problem (RCSPP) (see the Section 3.2), which can be solved by dynamic programming algorithm [20]. When solving the RCSPP, several shifts with negative reduced cost are to be generated, if exist, instead of only one. Because adding several of shifts more than one to the RMP can accelerate the solution speed of the column generation algorithm [9].

The main steps of column generation algorithm (CGA) are as follows.

Step 1: Construct initial shift set N_0 and form initial restricted master problem RMP$_0$, $r \leftarrow 0$;

Step 2: Solve RMP$_r$ (corresponding to shift set N_r) and obtain dual multipliers π_i^r, θ_h^r, δ_k^r and μ_s^r;

Step 3: Solve the subproblem SP$(N \backslash N_r)$ and obtain $nNRC$ negative reduced cost shifts;

Step 4: If $nNRC = 0$, then return to Step 7;

Step 5: Add $nNRC$ negative reduced cost shifts to N_r and obtain N_{r+1} and RMP$_{r+1}$;

Step 6: $r \leftarrow r + 1$, return to Step 2;

Step 7: If the solution of RMP$_r$ is integer, then the optimal solution is obtained and stop.

Step 8: Implement Step 1 to Step 6 at every node of a branch-and-bound tree to get an optimal (or near-optimal) integer solution.

In the CGA described above, the initial shift set N_0 in Step 1 can be constructed by generating efficient shifts with heuristic methods. Since heuristic methods need problem-specific knowledge, we adopt a more convenient way to construct RMP$_0$, i.e. generating artificial variables. By setting large penalty costs to all artificial variables, they will be eliminated in later iterations of the column generation algorithm. The main work of generating artificial variables is to construct an initial artificial coefficient matrix $\bar{A}_0$ for RMP$_0$ such that RMP$_0$ has at least a feasible solution. Denote $\bar{m}$ the number of constraints contained in RMP0, i.e. $\bar{m} = m + |H| + |K| + |S|$. We construct the initial matrix $\bar{A}_0$ as follows: it consists of m columns and each column is a $\bar{m}$-dimensional vector. The values of elements in the $j^{\text{th}}(j \leq m)$ column are set as follows: $\forall i \in M$, if $i = j$ then $a_{ij} = 1$, otherwise $a_{ij} = 0$; $\forall h \in H$, $b_{hj} = 1$; $\forall k \in K$, $e_{kj} = 1$; $\forall s \in S$, $f_{sj} = 0$, $g_{sj} = 0$. Then, all artificial variables with the value 1 form a feasible solution for RMP$_0$.

The subproblem SP$(N \backslash N_r)$ in Step 3 is formulated as a RCSPP, which will be introduced in Section 3.2. It is very time-consuming to solve the subproblem RCSPP, which is NP-hard and usually needs to be called for hundreds of thousands times. To reduce the times of calling the RCSPP solver embedded in a column generation algorithm, we use an idea similar to the one proposed in literature [6], i.e., construct a shift pool that is used to generate negative reduced cost shifts. In this context, we construct a shift pool *PSP* for each node of branch-and-bound tree. For the root node, let the shift pool $PSP = \emptyset$; for other node, the shifts in *PSP* are inherited from its parent node, which include the shifts in *PSP* of its parent node and the shifts generated by the column generation algorithm in its parent node. When solving a subprolem in a non-root node, the negative reduced cost shifts are first selected from its corresponding *PSP*. The RCSPP solver is called to generate new shifts only when there are no negative reduced cost shifts in *PSP*.

$\boxed{3.2}$ The Subproblem Solution

When using column generation to solve vehicle routing and crew scheduling problem, the subproblem is usually formulated as a resource constrained shortest past problem (RCSPP) [20]. The main idea is to firstly construct a directed graph $G(V, A)$ (V and A represent its node set and arc set respectively) by using all vehicle works (blocks), and properly define the reduced cost on arc set A, such that any feasible shift can be represented by a path in graph G and the shift with the least negative reduced cost corresponds to the shortest path. However, some (partial) paths may be invalid since they may violate the shift-level constraints described in Section 2.1. So it needs to define the shift-level constraints on the arc set A in the form of resource consumptions (one shift-level constraint corresponds to one type of resource), such that the accumulated amount of resources consumed by valid (partial) paths do not exceed their corresponding available resources. When defining the reduced cost on arc set A, it also requires that the reduced cost for any valid path be equivalent to the reduced cost of its corresponding shift. After all resources and reduced cost are defined in graph G, the shortest (i.e. least reduced cost) valid path can be obtained by multi-label dynamic programming. When using dynamic programming, a label represents a partial path originate from the source node. In this paper, we use the dynamic programming framework proposed by Feillet et al. [13] for solving variants of vehicle routing problems.

In the following context, we will detail the construction of graph $G(V, A)$ in Section 3.2.1, and the resource definition in Section 3.2.2. It must be mentioned that it is difficult to directly define the reduced cost rc_j (see formula (3.2)) on arc set A, because we do not know the term F_j (which is determined by the additional constraints) beforehand. In order to deal with the difficulty, we propose a decomposition approach in which the original subproblem is firstly divided into several subproblems according to the additional constraints and then the term F_j can be determined before solving each subproblems. We will detail the decomposition approach (include reduced cost definition and subproblem solving) in Section 3.2.3.

$\boxed{3.2.1}$ Construction of Graph $G(V,A)$

First, define a *source node* S and a *sink node* T, which respectively represent the sign-on point and the sign-off point of a shift. Then, all ROs are used to define two other types of nodes, i.e. *start node* and *end node*, which represent the start point and end point of spells respectively. Since all ROs (except the start RO and the end RO) in a vehicle block can be as the start point of spells or the end point of spells, they are defined as end nodes of spells and their individual duplications are defined as start nodes of spells. At last, the start ROs and the end ROs of vehicle blocks are defined as start nodes and end nodes respectively. Then, some pair of nodes can be linked to form five types of directed arcs in the following:

sign-on: which links the node S with any possible start node of a spell
sign-off: which links any possible end node of a spell with the node T
piece: which links a start node with its adjacent end node in the same vehicle block
null: which links an end node with its adjacent start node in the same vehicle block
break: which links an end node with all possible start nodes that can form mealbreak time

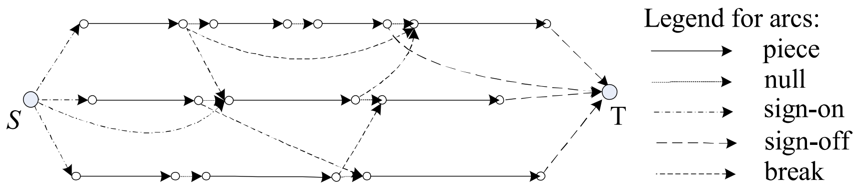

Figure 2: A partial directed graph G for shift generation

Figure 2 illustrates a partial network of the graph G, where 'null' represents a virtual arc with the length of 0. And the lengths of other four types of arcs are defined as: sign-on time (corresponding to sign-on arc), sign-off time (corresponding to sign-off arc), the duration of piece (corresponding to piece arc), the length of break time (corresponding to break arc).

Note that three types of shifts (i.e. straight, split and tripper) are considered in this context, we construct three directed graphs (i.e. straight graph, split graph and tripper graph) for generating each type of shifts. The difference between the straight graph and the split graph is the break arc set, which is constructed according to the minimum break time $Q^1_{\min}$ or $Q^2_{\min}$ stipulated by the shift-level constraints. For the tripper graph, there is no break arc.

$\boxed{3.2.2}$ Resource Definition

As indicated before, it needs to define several types of resources to model the shift-level constraints. Since the constraints about the minimum break time have been incorporated into the graph G, we only need to consider the remaining five shift-level constrains, which are maximum length of a spell $Q^1_{\max}$, maximum number of spells $Q^2_{\max}$, maximum driving time $Q^3_{\max}$, maximum working time $Q^4_{\max}$ and maximum spreadover time $Q^5_{\max}$. So, we define five types of resources: length of a spell (denoted as Tspell), number of spells (denoted as Nspell), driving time (denoted as Tdive), working time (denoted as Twork) and spreadover time (denoted as Tspread).

For the convenience of description, we firstly define some notations. Denote R_p the partial path that originates from source node S to the node p, and denote T^k_p the amount of the k^{th} resource accumulated by R_p, $1 \leq k \leq 5$. Let s^k_{pq} be the amount of k^{th} resource consumed by arc $(p,q) \in A$, $1 \leq k \leq 5$. When using the multi-label dynamic programming to search the path with least reduced cost, each partial path R_p will be extended to node $q \in V$ to form a new partial path R_q, i.e. $R_q = R_p \cup (p,q)$, if $(p,q) \in A$ and $T^k_q \leq Q^k_{\max}$, where $T^k_q = T^k_p + s^k_{pq}(1 \leq k \leq 5)$.

Now the main work is to define $s^k_{pq}(1 \leq k \leq 5)$, $\forall(p,q) \in A$, such that T^k_q $(1 \leq k \leq 5$, $\forall q \in V)$ can correctly reflect the validity of partial path R_q. For each arc $(p,q) \in A$, let l_{pq} be its length (see Section 3.2.1). According to the arc type, the amount s^k_{pq} of k^{th} resource consumed by $(p,q) \in A$ is defined in Table 1.

In table 1, $rate = 1$ if the type of shifts that will be generated is straight; otherwise $rate = 0$. Because a spell is defined as the vehicle works of the same block, it needs to recalculate the length of a new spell when a partial path passes break arcs. Therefore, s^1_{pq}

Table 1: Definition of resource consumption

Arc type	Tspell	Nspell	Tdrive	Twork	Tspread
sign-on-arc	l_{pq}	1	l_{pq}	l_{pq}	l_{pq}
sign-off-arc	l_{pq}	0	l_{pq}	l_{pq}	l_{pq}
null-arc	0	0	0	0	0
piece-arc	l_{pq}	0	l_{pq}	l_{pq}	l_{pq}
break-arc	$-T_p^1$	1	0	$l_{pq} \times rate$	l_{pq}

is set to $-T_p^1$ if (p,q) is a break arc. In addition, note that the resource definition is usually problem-specific and depends on the types of shift-level constraints. For more details about the general resource definitions, the reader can refer to Irnich and Desaulniers [20].

3.2.3 A Decomposition Approach to Solve the Subproblem

In this section, we consider the reduced cost definition and propose a generic decomposition approach that can deal with multiple additional constraints for solving the subproblem in a column generation algorithm.

As addressed before, the subproblem solver aims to generate a shift with the minimum negative reduced cost in the directed graph G. Therefore, it needs to set a reduced cost for each arc in G, such that the reduced cost of any feasible path in G is equal to the reduced cost of its corresponding shift. For any arc $(p,q) \in A$, its cost and reduced cost are denoted as c_{pq} and rc_{pq} respectively. Where, c_{pq} is equal to the amount of working time consumed by the arc (p,q) (according to the cost definition of a shift described in Section 2.2). Now, we need to calculate rc_{pq} and define it on arc (p,q). Assume a shift $j \in N$ corresponds to a path R in the graph G, the calculation formula (3.2) of its reduced cost rc_j can be transferred into as follows:

$$\begin{aligned} rc_j &= d_j - \sum_{i\in M} a_{ij}\pi_i^r + F_j \\ &= W + w_j c_j - \sum_{i\in M} a_{ij}\pi_i^r + F_j \\ &= W + w_j \sum_{(p.q)\in R} c_{pq} - \sum_{(p.q)\in R} \pi_{pq}^r + F_j \\ &= W + \sum_{(p.q)\in R} (w_j c_{pq} - \pi_{pq}^r) + F_j \end{aligned} \tag{3.4}$$

Where, if the arc (p,q) corresponds to the piece i, then $\pi_{pq}^r = \pi_i^r$ (dual multiplier); otherwise, $\pi_{pq}^r = 0$. Since W is a constant in formula (3.4), so it can be defined as a part of cost of all sign-on arcs to ensure that it contributes to the cost and the reduced cost of any shift to be generated. Similarly, the term $(w_j c_{pq} - \pi_{pq}^r)$ can be directly defined on arc (p,q). However, the term F_j cannot be directly defined on an arc $(p,q) \in A$, because we do not know its value beforehand. It cannot be calculated unless we know which the additional constraints restrict shift j (see formula (3.1)).

In order to deal with the difficulty, we present a decomposition strategy. The main idea is as follows. Firstly note that each shift set that is restricted by the additional constraints has some particular attributes (e.g. two-spell), hence it can be seen as a particular type of shifts. So, all shifts in N can be classified into several disjoint types of shift set according to the (particular) types of shift sets restricted by the formulas (2.13) to (2.15). Then, F_j can be calculated for each types of shift set, and can be handled as a constant. Consequently, the subproblem is divided into several subproblems and each subproblem corresponds to a generation of one type of shift set.

First, we classify all shifts in N into several types of shifts. Denote N_{st} the all straight shift set, N_{sp} the all split shift set, and N_{tr} the all tripper shift set (i.e. $N_{tr} = N \backslash N_{st} \backslash N_{sp}$); Denote $NA = \{N^h_{P_1}(h \in H),\ N^k_{P2}(k \in K),\ N^s_{C1}(s \in S),\ N^s_{C2}(s \in S),\ N_{st},\ N_{sp},\ N_{tr}\}$. In fact, some shift set in NA may be the same. If so, we remove the duplicate shift set in NA and get $|Z|$ types of shift set $N_i (i \in Z)$. Now, we reclassify shift set NA into several of more detailed disjoint shift set. Note that for any $j \in N$ and any $i \in Z$, there exist two possibilities: $j \in N_i$ or $j \notin N_i$. So the type of shift j can be represented by a $|Z|$-dimension 0-1 vector, where the i^{th} element of the vector indicates whether j belongs to N_i. Therefore in theory, all shifts in N can be reclassified into $2^{|Z|}$ disjoint types of shift set. However in practice, $|Z|$ is not too large (usually less than 15) and many combinations (i.e. 0-1 vectors) are infeasible because of the restrictions of labor rules. For example, if a shift is straight then it cannot be split or tripper. Similarly, a tripper shift cannot be a two-spell or three-spell shift. Therefore, we use an enumerative approach to remove all infeasible 0-1 vectors and obtain the remaining disjoint types of shift set $D_i (i \in V)$.

Then, the subproblem is decomposed into $|V|$ subproblems SP (D_i) $(i \in V)$, and each subproblem SP (D_i) is solved independently. Before solving SP (D_i), F_j $(j \in D_i)$ is first calculated by the formula (3.2), then the reduced cost of arc $(p, q) \in A$ is defined. At the i^{th} iteration, the solving process for SP (D_i) is the following:

Step 1: $F_j := 0$;

Step 2: For each $h \in H$, if $D_i \subseteq N^h_{P_1}$, then $F_j := F_j + (\alpha_h - 1)\,\theta^r_h$; $F_j := F_j + \alpha_h \theta^r_h$ otherwise;

Step 3: For each $k \in K$, if $D_i \subseteq N^k_{P2}$, then $F_j := F_j - \delta^r_k$;

Step 4: For each $s \in S$, if $D_i \subseteq N^s_{C1}$, then $F_j := F_j - \mu^r_s$;

Step 5: For each $s \in S$, if $D_i \subseteq N^s_{C2}$, then $F_j := F_j + \gamma_s \mu^r_s$;

Step 6: For each $(p, q) \in A$, its reduced cost rc_{pq} is defined as follows:

$$rc_{pq} = \begin{cases} W + (w_j c_{pq} - \pi^r_{pq}) + F_j & \text{If } (p,q) \text{ is a sign-on arc} \\ (w_j c_{pq} - \pi^r_{pq}) & \text{otherwise} \end{cases} \tag{3.5}$$

Step 7: Define some additional resources to make the shifts generated belong to D_i;

Step 8: Call the dynamic programming algorithm proposed by Feillet et al. [13] to solve the RCSPP in graph G, and get a negative reduced cost shift set $NEG(D_i)$.

The approach proposed to solve subproblem above is generic, and it can deal with the crew scheduling problems without additional constraints or with particular constraints. For example, when solving the problems without additional constraints, F_j is equal to 0 and the subproblem solving process corresponds to Step 6 to Step 8. When solving the problems with crew base constraints [17, 34], all shifts are first reclassified by crew bases into disjoint set of shifts (one set of shifts corresponds to one crew base), and then implement Step 1 to Step 8.

3.3 Branching Strategy

If the LP solution is integer, then it is the optimal solution of the extended SCF; otherwise, it needs to design approaches (e.g. branch-and-bound method) to obtain integer solutions.

In order to get integer solutions, one simple way is to use the traditional branch-and-bound method to search the integer solution within the set of shifts generated by the column generation. Although the approach is easy to implement, it may not obtain an optimal or good integer solution.

In order to get optimal or near-optimal solutions, we design a branching strategy to search integer solutions in a binary tree and implement the column generation at each node of the search tree. In the traditional branch-and-bound approach for solving 0-1 integer programming, branching on variables (0-1 branching rule) is the generic strategy to branch the search tree. One branch (1-branch) requires that the fractional variable selected must be within the solution while the other branch (0-branch) requires that the variable must not be within the solution. However, this branching strategy isn't suitable for column generation due to two reasons. One is that it makes the search tree unbalanced, and the other is that it is difficult to implement the 0-branch when solving the subproblems, as the 0-branch will change the subproblem structure. In this paper, a relief opportunity (RO) branching rule that is similar to the one proposed by Fores et al. [15] is designed based on the characteristics of crew scheduling problems. The RO branching rule bans all shifts which use the selected RO on one hand, and bans all shifts which do not use the selected RO on the other hand. However, Fores et al. [15] did not give exact RO selection rules. Moreover, it is easier to implement RO branching rule in Fores et al. [15], since they need not to generate new shifts after branching.

Firstly, we give some notations. For the current LP problem with fractional solutions, let $\bar{N}$ be the shift set that their corresponding variables are fractional, and let RO be the set of ROs contained in all vehicle blocks. For each $r \in RO$, we define a shift set J_r as follows:

$$J_r = \{j \mid r \text{ is contained in shift } j, \text{but not used by the shift } j,\ j \in \bar{N}\}.$$

Now, we describe the RO branching strategy adopted as follows:

Step 1: Select a RO $p \in RO$ according to a selection rule. In this context, we consider two RO selection rules:

(1) minimum infeasibility:

$$p = \arg\max_{r \in RO} \{\sum_{j \in J_r} x_j \mid 0 < \sum_{j \in J_r} x_j < 1\} \tag{3.6}$$

(2) maximum infeasibility:

$$p = \arg\min_{r \in RO} \{|\sum_{j \in J_r} x_j - 0.5| \mid 0 < \sum_{j \in J_r} x_j < 1\} \tag{3.7}$$

Step 2: If such a p exists, return to Step 4; otherwise, return to Step 3;

Step 3: Search an integer solution for the current node, update the current upper bound.

Denote J the shift set which consists of shifts that have nonzero value in the solution of current node (generally, $|J|$ is not too large). Implement traditional branch-and-bound (0-1 branching) to get optimal integer solution within the shift set J. Where, we use the maximum infeasibility rule to select branching variable, i.e. $x_k = \min\{|x_j - 0.5| \mid 0 < x_j < 1, j \in J\}$, and use best-bound strategy to select the next node to branch.

Step 4: Branch the current node into two nodes:

Left node (not use the RO p):

$$\sum_{j \in J_p} x_j = 1 \tag{3.8}$$

Right node (use the RO p):

$$\sum_{j \in J_p} x_j = 0 \tag{3.9}$$

Such a RO-branching rule is easy to implement when solving the subproblems of column generation. For the left node, we only need to remove the arcs that coming in to the RO p and coming out from its duplicate RO $p^{'}$. For the right node, we only need to remove the null arc that link the RO p with its duplicate RO $p^{'}$.

4 Computational Results

4.1 Test Instances and Parameters Setting

To test the proposed formulation and the solution approaches, we consider the 12 groups of instances which are derived from Chinese public transport. For each instance, Table 2 lists its number of pieces (np), vehicle blocks (nb), and relief points (nr), where all instances are sorted in an increasing order of the number of pieces contained.

Table 2: Description of data instances

Ins.	1	2	3	4	5	6	7	8	9	10	11	12
np	128	145	184	255	300	374	405	410	466	528	584	830
nb	14	16	8	30	44	18	30	54	46	40	49	62
nr	142	162	192	285	344	392	435	464	512	568	633	892

All procedures are implemented in C++ and the computation is carried out on a laptop with a Pentium Dual-Core T4300 2.1 GHz processor and 2 GB of RAM, using CPLEX 12.4 as underlying linear programming solver.

After implementing some preliminary experiments, we found that the column generation algorithm has the property of snow convergence as addressed by many literatures [4, 9]. For some instances, it needs to take too much time to solve the linear relaxation solution. In order to obtain good solutions in reasonable time, we adopt some parameters and strategies to control the process of column generation algorithm:

(1) The maximum number of node to be explored

In preliminary experiments for some instances, the solution of a node improves only a little comparing with its parent node. The possible reason is that there are too many shifts that can substitute the shifts that are banned by branching. So, it needs to explore a large number of nodes to search the optimal solutions. Considering the limited memory space and computational time, the maximum number of node (which is generated by using RO branching rule) to be explored in branch-and-bound tree is set to 500. Especially for the largest instance (i.e. the 12$^{\text{th}}$ instance) that contains 830 pieces, it needs too much time to solve a node of the search tree. So we set its maximum number of node to 200.

(2) Node selection strategy

We test two usually used node selection strategy: depth-first strategy and best-bound strategy.