An estimation of distribution algorithm for public transport driver scheduling

Yindong Shen
Key Laboratory of Image Processing and Intelligent Control of Education Ministry, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China
Email: yindong@hust.edu.cn

Jingpeng Li*
Division of Computer Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK
Email: jli@cs.stir.ac.uk
*Corresponding author

Kunkun Peng
Key Laboratory of Image Processing and Intelligent Control of Education Ministry, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China
Email: pengkunkun@126.com

Abstract: Public transport driver scheduling is a process of selecting a set of duties for the drivers of vehicles to form a number of legal driver shifts. The problem usually has two objectives which are minimising both the total number of shifts and the total shift cost, while taking into account some constraints related to labour and company rules. A commonly used approach is firstly to generate a large set of feasible shifts by domain-specific heuristics, and then to select a subset to form the final schedule by an integer programming method. This paper presents an estimation of distribution algorithm (EDA) to deal with the subset selection problem which is NP-hard. To obtain a candidate schedules, the EDA applies a number of rules, with each rule corresponding to a particular way of selecting a shift. Computational results from some real-world instances of drive scheduling demonstrate the availability of this approach.

Keywords: metaheuristic; estimation of distribution algorithm; EDA; Bayesian network; driver scheduling.

1 Introduction

Driver scheduling for public transport (e.g., train, bus and tram) is a complex problem which involves creating shifts (i.e., the work that a single driver carries out in one day) from a set of predetermined trips and then assigning shifts to a number of notional drivers in the most efficient way to cover all the required work (Kwan, 2011; Shen and Chen, 2014). As solving the problem efficiently may have a significant impact on costs and quality of service for public transportation companies, research into this area has been active especially since the 1990’s (Caprara et al., 1997; Wren et al., 2003; Hickman et al., 2008; Goel et al., 2012; Chen and Shen, 2013; Tóth and Krész, 2013; Shen et al., 2015).

Note that in this paper, we only deal with the driver scheduling problem which means that the shifts are only notional because they are not yet assigned to actual drivers. In real world implementation, we still need to deal with the subsequent driver rostering problem which generates for each group of drivers a cyclic or non-cyclic roster with the work assigned to actual personnel on a weekly or monthly basis, by taking into account issues such as fairness, personal preference and safety labour laws.

Figure 1 shows a portion of vehicle graph containing a number of points, each of which represents a pair of time and station where a driver may leave the vehicle to take a break or to transfer to another vehicle. The work between two consecutive points on the vehicle graph is called a piece of work.
In real implementation, a driver normally works on several spells, each of which contains consecutive pieces of work on a vehicle. Not all points are suitable for driver switch. A shift normally starts when the driver signs on at a depot, and ends when the driver returns to depot to sign off. Figure 2 illustrates the composition of a 3-spell shift in relation to some vehicle work.

The problem addressed in this paper is similar to that in many literatures (Kwan et al., 2001; Li and Kwan, 2003, 2005; Aickelin et al., 2009; Shen et al., 2013). The main constraint are to assign every piece of work to a number of shifts while complying with all the operational constraints and labour rules on the shifts (Hickman et al., 2008; Chen et al., 2013). The objectives are to minimise the total number of shifts and the total shift cost, where the former has priority over the latter. The objectives are somewhat different from that in some literatures. For example, in Lourenço et al. (2001), there was not priority between the two objectives. Moreover, several other objectives are included such as minimising the total number of one-spell shifts and the number of vehicle changes etc. In Dias et al. (2002), only one objective was considered, i.e., minimising the total shift cost.

Driver scheduling can be solved with the following generation-and-selection approach (Fores et al., 2002): a large set of candidate shifts is first generated by simple heuristics that are parameterised to reflect on the driver work rules of individual companies; a set covering model is then applied to select the least cost subset that covers all the work.

The difficulty of the approach lies in its selection phase due to the NP-hard nature of the set covering problem. A lot of work has therefore been done to efficiently explore the near-optimal solutions. Over the past decades, meta-heuristics have been widely used, in which genetic algorithms (GAs) are mostly used (Wren and Wren, 1995; González Hernández and Corne, 1996; Kwan and Wren, 1996; Kwan et al., 2001; Li and Kwan, 2003; Shen et al., 2013). Attempts have also been made by using other meta-heuristics such as tabu search (Cavique et al., 1999; Shen and Kwan, 2001; Yaghini
et al., 2015), simulated annealing (Hanafi and Kozan, 2014), greedy randomised adaptive search procedure (De Leone et al., 2011a, 2011b), squeaky wheel optimisation (Aickelin et al., 2009), constraint programming (Layfield et al., 1999; Curtis, 2000), ant colony optimisation (Forsyth and Wren, 1997; Huang et al., 2011), etc.

In this paper, we propose an estimation of distribution algorithm (EDA) (Larranaga and Lozano, 2001; Aickelin et al., 2007) for driver scheduling. The proposed EDA belongs to the general class of GAs. However, unlike previous work that used GAs where learning is implemented implicitly, learning in the proposed EDA is explicit, that is, the building blocks to form good solutions are identified explicitly. The proposed EDA generates schedules by choosing a suitable rule, from a set containing a number of candidate rules, for the assignment of each piece of work. In this case, a solution to the problem is denoted as a rule string (or a sequence of rules) corresponding to work pieces from the first one to the last one.

To derive good rule sequences, the EDA employed Bayesian network (Pearl, 1998), which is a directed acyclic graph with each node (i.e., variable) corresponding to the option of a rule by which a piece of work is assigned. A directed edge between the two corresponding nodes represents the causal relationship between two variables.

The EDA learns to identify good partial solutions (i.e., building blocks) and to complete them by building a Bayesian network of the joint distribution of solutions (Pelikan et al., 1999). Based on a set of current solutions, the conditional probability values for each variable in the network are computed. These values are then used to generate a new instance for each variable to form a solution (i.e., a rule sequence). A new set of rule sequences can be generated in this way, some of which replace previous sequences by a fitness selection strategy. Based on the updated set of rule sequences, the conditional probability values for all variables in the Bayesian network are updated again using the updated set of rule sequences until a stopping condition met. We therefore say the approach tries to explicitly identify and mix promising building blocks.

2 Rules for shift structure evaluation

After a large set of potential shifts is generated, the driver scheduling problem can be solved as a set covering integer programming model. The structure of each shift in the set of potential shifts varies significantly, and a good schedule is more likely to include more well-structured (or more effective) shifts. However, it is not easy to judge the shift effectiveness because the criteria bear some uncertainty. In this paper, we use fuzzy evaluation method which gives each shift a quantitative value according to its structure. The higher the value, the fitter the structure is.

Based on our domain knowledge, the EDA can choose from the following six rules for shift evaluation.

- $R_1$ – total working time
- $R_2$ – ratio of total working time to paid time (i.e., hours from sign on to sign off)
- $R_3$ – number of pieces of work
- $R_4$ – number of spells that a shift has
- $R_5$ – fractional cover obtained by linear programming (LP) relaxation
- $R_6$ –...
• $R_6$ – overall evaluation by a more advanced evaluation function.

These rules describe quantitatively the characteristic of shift structure from different aspects. Obviously, more heuristic rules could be used to build schedules.

2.1 Rule R1

Not all of a driver’s paid hours from sign on to sign off are working time. For economic consideration, shifts with longer working time are more desirable than those with shorter working time. Thus, we may assume that the goodness of a potential shift $S_j$ increases with its total working time. Furthermore, since a much smaller number of shifts are selected from the set of potential shifts to produce schedules, we wish the variation of goodness among these elite shifts is small. For shifts with relatively longer working time, their goodness should increase as smoothly as possible so that they have more chances to be selected at later iterations if they are not selected at the current iteration.

Based on this thought, the type of increase should be nonlinear. An S-shape quadratic membership function $\mu_{R_1}$ therefore is used to define rule $R_1$ as

$$\mu_{R_1} = \begin{cases} 
2 \left( \frac{x_1 - a_{\min}^1}{a_{\max}^1 - a_{\min}^1} \right)^2, & a_{\min}^1 \leq x_1 < a_{\min}^1 + \frac{a_{\max}^1 - a_{\min}^1}{2} \\
1 - 2 \left( \frac{x_1 - a_{\max}^1}{a_{\max}^1 - a_{\min}^1} \right)^2, & a_{\min}^1 + \frac{a_{\max}^1 - a_{\min}^1}{2} \leq x_1 \leq a_{\max}^1
\end{cases}$$

(1)

where $x_1$ is the total working time of $S_j$, $a_{\max}^1$ and $a_{\min}^1$ are the maximum and minimum total working time among all the shifts, respectively.

2.2 Rule R2

Apart from the first rule referring to the absolute working time, the relative ratio of actual working time to the paid hours may be regarded as the second rule. Obviously, a shift with larger ratio is more desirable. For the same reason explained in Section 2.1, an S-shape quadratic membership function $\mu_{R_2}$ is applied to define rule $R_2$ as

$$\mu_{R_2} = \begin{cases} 
2 \left( \frac{x_2 - a_{\min}^2}{a_{\max}^2 - a_{\min}^2} \right)^2, & a_{\min}^2 \leq x_2 < a_{\min}^2 + \frac{a_{\max}^2 - a_{\min}^2}{2} \\
1 - 2 \left( \frac{x_2 - a_{\max}^2}{a_{\max}^2 - a_{\min}^2} \right)^2, & a_{\min}^2 + \frac{a_{\max}^2 - a_{\min}^2}{2} \leq x_2 \leq a_{\max}^2
\end{cases}$$

(2)

where $x_2$ is the ratio of total working time to spreadover for $S_j$, $a_{\max}^2$ and $a_{\min}^2$ are the maximum and minimum ratio among all the shifts, respectively.

2.3 Rule R3

A shift may contain several spells, with each spell consisting of a number of consecutive pieces of work on the same vehicle. The number of shift in a final schedule might
potentially be reduced if each shift tries to cover as more pieces of work as possible while satisfying the constraints. The membership function of the third rule can be designed based on such assumption. However, the nonlinear functions used for rules $R_1$ or $R_2$ are not suitable as the number of work pieces that a shift covers is usually a discrete value smaller than 30. Hence, we use the following linear membership function $\mu_{R_3}$ to define rule $R_3$ as

$$
\mu_{R_3} = \frac{x_3 - a_{3_{\min}}}{a_{3_{\max}} - a_{3_{\min}}},
$$

where $x_3$ is the number of pieces of work contained in $S_j$, $a_{3_{\max}}$ and $a_{3_{\min}}$ are the maximum and minimum number of pieces of work, respectively.

2.4 Rule $R_4$

The heuristic used to generate a large set of potential shifts is normally parameterised to generate shift with up to four spells, because a shift with over 4-spell is rarely efficient in real operation, and also the combination of work pieces to form such shifts would be so many to cause computational difficulty.

Among the shifts with up to four spells, there are still differences in terms of user preferences. A one-spell shift may either be an overtime shift, or a shorter shift without a meal break in between the work pieces. This type of shifts is generally discouraged by transport operators, even if they are sometimes critical in forming optimal schedules.

Among the shifts with two, three and four spells, the two-spell shift is the most preferable as it is inherently more robust, and the three-spell shift is more preferable than the four-spell shift as a shift with more spells would inevitably lead to an extra meal break or more time for the driver to switch to another vehicle. Hence, membership function $\mu_{R_4}$ for rule $R_4$ is designed as

$$
\mu_{R_4} = \begin{cases} 
0, & \text{if } x_4 = 1 \text{ or } x_4 = 4 \\
\frac{1}{2}, & \text{if } x_4 = 3 \\
1, & \text{if } x_4 = 2
\end{cases}
$$

where $x_4$ is the number of spells contained in shift $S_j$.

2.5 Rule $R_5$

As we mentioned earlier, the driver scheduling problem can be traditionally formulated as a set covering problem ILP model. We firstly define the notations as follows:

- $n$ – number of potential shifts
- $m$ – number of pieces of work to be covered
- $c_j$ – cost of shift $j$
- $a_{ij}$ – 0–1 integer constants, with 1 indicating shift $j$ covers piece $i$ and 0 otherwise
• $x_j - 0–1$ integer variables, with 1 indicating shift $j$ is included in the solution and 0 otherwise.

We then have

\[
\text{Min} \sum_{j=1}^{n} c_j x_j
\]  

(5)

s.t. \[
\sum_{j=1}^{n} a_{ij} x_j \geq 1, \ i \in \{1, 2, \ldots, m\}
\]

(6)

\[
x_j = 0 \text{ or } 1, \ j \in \{1, 2, \ldots, n\}
\]

(7)

Objective (5) is to minimise the total cost, constraint (6) ensures that each piece of work is covered by at least one driver, and constraint (7) requires that all the shifts be considered.

ILP techniques, such as Branch-and-Bound, are limited by the size of search space to be explored within realistic time. Hence, the ILP processes often have to be stopped before finding any of the integer solutions. However, the relaxed LP problem which ignores the integer constraints (i.e., constraint 7) can usually be solved quickly. Kwan et al. (2001) experimentally demonstrated that, for a shift $S_j$, the closer its fractional value in the relaxed LP to 1, the higher its chance to be selected in the final schedule. Hence, we consider the relaxed LP solution as an extra rule $R_5$.

For the membership function of rule $R_5$, it should be nonlinear as the shifts in the integer solution mostly have fractional values close to 1. Having the characteristics of being smooth and non-zero at all points, the Gaussian distribution function $\mu_{R_5}$ is employed to define rule $R_5$ as

\[
\mu_{R_5} = \begin{cases} 
\left(\frac{x_j - \alpha}{\beta}\right)^2, & \text{if } S_j \text{ is in the fractional cover,} \\
0, & \text{otherwise}
\end{cases}
\]

(8)

Let $\mu_{R_5} = 1$ when $x_5 = a^5_{\max}$, and $\mu_{R_5} = 0.01$ when $x_5 = a^5_{\min}$, where $x_5$ is the fractional value of $S_j$ in the relaxed LP solution, $a^5_{\max}$ and $a^5_{\min}$ are the maximum and minimum values in fractional cover, respectively. We can then get

\[
\begin{align*}
\alpha &= a^5_{\max} \\
\beta &= \frac{\left(a^5_{\max} - a^5_{\min}\right)^2}{\ln 0.01}
\end{align*}
\]

(9)

\subsection{Rule $R_5$}

The aforementioned five rules $R_i$ ($i = 1, \ldots, 5$) are used to evaluate the shift structure from different aspects. Since these rules are compensative each other (Li, 2002), a natural thought is to combine them together to give an overall evaluation, by using any of the commonly-used aggregation operators such as product, arithmetic mean and geometric
mean. For simplicity reason, we use the arithmetic mean operator to define a new concept called structural coefficient \( f_1(S_j) \) of shift \( S_j \) and formulate it as

\[
f_1(S_j) = \sum_{i=1}^{5} w_i \mu_i, \quad \forall j \in J, \tag{10}
\]

where \( w_i (w_i \geq 0) \) denote the weights (or importance) of rules \( R_i (i = 1, \ldots, 5) \), satisfying

\[
\sum_{i=1}^{5} w_i = 1. \tag{11}
\]

To determine which shift should be used during the schedule constructing process, Li and Kwan (2003) proposed an integration of over-cover ratio, \( f_2(S_j) \in [0,1] \), to form the following evaluation function \( F(S_j) \):

\[
f(S_j) = f_1(S_j) \times f_2(S_j), \quad \forall j \in J. \tag{12}
\]

Over-cover means a piece of work is covered by more than one shifts. The over-cover ratio of a shift is the ratio of the non-overlapped working time to the total working time of this shift, formulated as

\[
f_2(S_j) = \frac{\sum_{k=1}^{\left|S_j\right|} (\alpha_{jk} \times \beta_{jk})}{\sum_{k=1}^{\left|S_j\right|} \beta_{jk}}, \quad \forall j \in J, \tag{13}
\]

where \( \left|S_j\right| \) = number of pieces of work in \( S_j \), \( \beta_{jk} = \) working time for work piece \( k \) in \( S_j \), and

\[
\alpha_{jk} = \begin{cases} 
0, & \text{if work piece } k \text{ in } S_j \text{ has been covered by any other used shifts } S_i; \\
1, & \text{otherwise.}
\end{cases}
\]

3 EDA for driver scheduling

This section discusses an EDA for driver scheduling, including the construction of a Bayesian network and learning by the network.

3.1 Bayesian network

Graphical models are a marriage between probability theory and graph theory (Jordan, 1999). They are graphs where vertices represent random variables and edges represent conditional dependence assumptions (Edwards, 2000). In a directed graphical model, namely Bayesian network, each of its nodes represents one variable, and each of its edges represents a relationship between two corresponding nodes (or variables). Bayesian networks can therefore be used to reveal the structure of a problem as each variable corresponds to one position in the sequences representing a particular solution. They can also be used to generate new variables instances with properties similar to those of given data.
Any complete probabilistic model of a domain must represent the joint distribution of every possible event defined by the values of all the variables. The number of such events is exponential: for \( n \) binary nodes, the full joint needs parameters as many as \( O(2^n) \). To achieve compactness, Bayesian networks factor the joint distribution into local conditional distributions denoted as

\[
P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | pa(X_i)),
\]

where \((X_1, \ldots, X_n)\) denotes the vector of all the variables in the problem, \(pa(X_i)\) is the set of parents of \(X_i\), and \(P(x_i | pa(X_i))\) is the conditional probability of \(X_i\) conditioned on its parent \(pa(X_i)\).

**Figure 3** A Bayesian network for driver scheduling

In our driver scheduling problem, the number of pieces of work is fixed, and the goal is to create schedules by assigning a subset of shifts from the large set to cover all the pieces of work in the most efficient way. The EDA approach achieves this by using one suitable rule, from a rule set containing six rules, for each work piece’s assignment. Thus, a potential solution is represented as a sequence of rules corresponding to the assignment of pieces of work from the first one to the last one. We choose this approach, aiming to simulate the explicit scheduling process by humans who build schedules based on a set of simple rules. The schedules generated by humans are mostly of high quality due to the human ability to switch between rules, based on different states encountered in the scheduling process. We expect our proposed EDA to perform similarly this role.

**Figure 3** illustrates the Bayesian network constructed for the driver scheduling problem. The vertex \(N_{ij}(i \in \{1,2,\ldots,m\}; j \in \{1,2,\ldots,n\})\) denotes that work piece \(i\) is assigned by rule \(j\). The directed edge from vertex \(N_{ij}\) to vertex \(N_{i+1,j'}\) denotes a causal
relationship of ‘\(N_j\) causing \(N_{i+1,j}\)’. In this network, a possible solution (i.e., a rule sequence) is represented as a directed path from work piece 1 to work piece \(m\) connecting \(m\) vertices.

In our proposed EDA, the learning type is ‘known topology and full observation’ and the goal of this learning is to find the variable values of all vertices that maximise the likelihood that the training data contains a number of independent instances. Learning in our EDA is in essence counting, and we use ‘\(#\)’ to denote ‘the number of’ in the following equations. We calculate the conditional probabilities of each possible value for each variable given all possible values of its parents. For example, for node \(N_{i+1,j}\) with a parent node \(N_{ij}\), its conditional probability is

\[
P(N_{i+1,j} | N_{ij}) = \frac{P(N_{i+1,j}, N_{ij})}{P(N_{ij})} = \frac{\#(N_{i+1,j} = \text{true}, N_{ij} = \text{true})}{\#(N_{i+1,j} = \text{true}, N_{ij} = \text{true}) + \#(N_{i+1,j} = \text{false}, N_{ij} = \text{true})}.
\]

For node \(N_{ij}\) with no parent, its probability is calculated as

\[
P(N_{ij}) = \frac{\#(N_{ij} = \text{true})}{\#(N_{ij} = \text{true}) + \#(N_{ij} = \text{false})}.
\]

These probability values are used to generate new rule sequences (or new solutions). Since the rule used in the first decision point has no parent, it will be chosen by the probabilities given in equation (16). The next rule will be chosen from nodes \(N_{ij}\) according to the probabilities conditioned on the previous nodes calculated as equation (15). This process is repeated until the last node is chosen. A link from piece 1 to piece \(m\) therefore forms, representing a new possible solution.

In summary, a solution corresponds to a rule sequence whose length is equal to the number of pieces, that is, each piece is assigned by a rule which selects a shift to cover it. After a set of solutions is obtained, for each piece \(i\), we could obtain the number of times that work piece \(i\) is assigned using a rule (say rule \(j\)), denoted by \(N_{ij}\). Then according to equations (15) and (16), we would further calculate the conditional probabilities for each vertex \(N_{ij}\). Finally, we would use the roulette-wheel strategy to select a rule for each piece, where rule \(j\) assigns piece \(i\) with the probability.

### 3.2 The EDA approach

Our EDA first generates at random an initial population of rule sequences, among which a set of more promising rule sequences is selected. Any selection method biased towards better fitness may be used, and in this paper, the traditional roulette-wheel selection is applied to select a rule for each piece of work. The conditional probabilities of each node in the Bayesian network are computed. New rule sequences are generated by using these conditional probability values, and are added into the previous population, replacing some of the underperformed rule sequences. In more detail, the steps of the EDA for driver scheduling are presented as follows:

1. Set \(t = 0\), and generate an initial population \(P(0)\) at random, where each individual in the population consists of a sequence of rules denoted as \(P_j = (R_1, \ldots, R_n)\).
For each individual $P_j$ in the population $P(t)$, calculate its fitness value, which is achieved by firstly generating its corresponding schedule and then evaluating it by the following steps:

2.1 Set $i = 1$.

2.2 Use rule $R_i$ to select a shift $S_i$ to cover the work piece $i$.

2.3 If shift $S_i$ also covers work piece $k$, $S_i$ should be given high priority when applying rule $R_k$ as otherwise the overall cost and the number of shifts would tend to be large. The high priority can be obtained by adding a large constant (such as 5,000) to the evaluation result on $S_i$.

2.4 Set $i = i + 1$, and go to Step 2.2 until $i \leq m$ where $m$ is the total number of pieces of work.

2.5 Evaluate the resulting schedule by calculate the \( \sum_{j=1}^{l} (c_{j*} + C) \), where $l$ is the number of shifts in the schedule, $c_{j*}$ is the cost of shift $S_j*$ and $C$ is a large constant. Since in most driver scheduling problems the first objective is to minimise the number of shifts, a large constant of $C$ per shift gives priority to this.

3 Use roulette-wheel to select a set of promising rule sequences $S(t)$ from $P(t)$.

4 Compute the conditional probabilities of each vertex according to this set of promising solutions.

5 For the assignment of each piece of work, the roulette-wheel method is used to select one rule based on the conditional probabilities of all available vertices, thus obtaining a new rule sequence. A set of new rule sequences $O(t)$ will be generated in this way.

6 Create a new population $P(t+1)$ by replacing some rule sequences from $P(t)$ with $O(t)$, and set $t = t+1$;

7 If the termination condition is not met (we use 100 generations), go to Step 2.

4 Computational results

The EDA approach was coded in C++. All problems were run on the same Intel Core Duo CPU 2.10GHz with 3GB RAM PC and Windows XP operating system. For consistency, the same weight distribution, $W = (0.15, 0.15, 0.15, 0.15, 0.40)$, for membership functions of the sixth rule formulated in Equation (10) is applied to all algorithms on all test problems presented in this paper. The weight distribution used is a good combination of the parameter values, which was proposed by Li and Kwan (2003) for a different set of driver scheduling problems in the UK. Note that the time of getting the best parameter values for each instance is time-consuming, and our proposed EDA does not aim to find the best parameters, because our EDA could generate high quality schedules due to the ability of switching between different rules, instead of just using the sixth rule.
4.1 Test problems

Table 1 shows the characteristics of 11 test instances, all of which were derived from the real-world problems in China. Since the major concern about using heuristics is the quality of the obtained solution, for comparison purpose, the computational results in terms of number of shifts and total cost by a genetic algorithm with fuzzy evaluation (GAFE) proposed by Li and Kwan (2003) are also given in Table 1. The runtime of GAFE is approximately 10–30 seconds per data instance.

Table 1 Size of test problems and the results of the GAFE

<table>
<thead>
<tr>
<th>Data</th>
<th># of blocks</th>
<th># of pieces</th>
<th># of possible shifts</th>
<th>GAFE’s schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td># of shifts</td>
</tr>
<tr>
<td>P1</td>
<td>8</td>
<td>184</td>
<td>14,218</td>
<td>24</td>
</tr>
<tr>
<td>P2</td>
<td>14</td>
<td>162</td>
<td>9714</td>
<td>28</td>
</tr>
<tr>
<td>P3</td>
<td>18</td>
<td>491</td>
<td>31,712</td>
<td>35</td>
</tr>
<tr>
<td>P4</td>
<td>30</td>
<td>389</td>
<td>74,864</td>
<td>51</td>
</tr>
<tr>
<td>P5</td>
<td>30</td>
<td>750</td>
<td>93,853</td>
<td>66</td>
</tr>
<tr>
<td>P6</td>
<td>40</td>
<td>528</td>
<td>31,493</td>
<td>69</td>
</tr>
<tr>
<td>P7</td>
<td>43</td>
<td>302</td>
<td>48,267</td>
<td>71</td>
</tr>
<tr>
<td>P8</td>
<td>46</td>
<td>467</td>
<td>97,386</td>
<td>72</td>
</tr>
<tr>
<td>P9</td>
<td>49</td>
<td>584</td>
<td>137,355</td>
<td>90</td>
</tr>
<tr>
<td>P10</td>
<td>54</td>
<td>410</td>
<td>55,977</td>
<td>101</td>
</tr>
<tr>
<td>P11</td>
<td>62</td>
<td>830</td>
<td>112,142</td>
<td>91</td>
</tr>
</tbody>
</table>

4.2 Results of the EDA

We run each instance 20 times by varying the pseudo random number seed for the proposed EDA (denoted as EDA₀ to distinguish from the later improved EDA). The average results of the 20 runs are listed in Table 2. Meanwhile, the relative percentage deviations (RPDs) over the GAFE in terms of shift number and total cost are also displayed in the Table. The results are unsurprisingly worse than the GAFE’s. Specifically, solution of the EDA₀ has 21.64% more shifts in terms of total shift number, and is 25.64% more expensive in terms of total cost on average. It should be noticed that amongst the six rules used in the EDA₀, only the sixth rule (same as that used in the GAFE) considers over-cover ratio. Since this ratio affects profoundly the number of shifts to be used in the resulting schedule, it is speculated that the first five rules would disturb the learning process instead of playing a positive role. Table 2 also shows the results produced by two versions of improved EDAs (namely EDA_C and EDA_W), which will be discussed in Sections 4.3 and 4.4 individually.
### Table 2
Comparative average results of 20 runs for the EDAs

<table>
<thead>
<tr>
<th>Data</th>
<th>EDA0’s schedule</th>
<th>EDAc’s schedule</th>
<th>EDAw’s schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of shifts</td>
<td>RPD (%)</td>
<td>Cost (h:m)</td>
</tr>
<tr>
<td>P1</td>
<td>26.75</td>
<td>11.46</td>
<td>197:34</td>
</tr>
<tr>
<td>P2</td>
<td>29.65</td>
<td>5.89</td>
<td>225:35</td>
</tr>
<tr>
<td>P3</td>
<td>40.85</td>
<td>16.71</td>
<td>326:07</td>
</tr>
<tr>
<td>P5</td>
<td>84.25</td>
<td>27.65</td>
<td>648:45</td>
</tr>
<tr>
<td>P6</td>
<td>79.3</td>
<td>14.93</td>
<td>617:33</td>
</tr>
<tr>
<td>P7</td>
<td>92.95</td>
<td>30.92</td>
<td>637:35</td>
</tr>
<tr>
<td>P8</td>
<td>94.35</td>
<td>31.04</td>
<td>709:24</td>
</tr>
<tr>
<td>P9</td>
<td>122.05</td>
<td>35.61</td>
<td>928:26</td>
</tr>
<tr>
<td>P10</td>
<td>122.6</td>
<td>21.39</td>
<td>893:36</td>
</tr>
<tr>
<td>P11</td>
<td>116.1</td>
<td>27.58</td>
<td>885:25</td>
</tr>
<tr>
<td>Avg.</td>
<td>21.64</td>
<td>25.64</td>
<td>1.70</td>
</tr>
</tbody>
</table>
4.3 Results of an improved EDA with over-cover ratio

An improved EDA with over-cover ratio for all rules (EDA_C for short) is proposed, in which the evaluation function $F(S_j)$ for each of the first five rules $R_i$ ($i=1,2,3,4,5$) is reformulated as

$$F(S_j) = \mu_{R_i} \times f_{2}(S_j), \forall j \in J,$$

where $\mu_{R_i}$ denotes the membership function for rule $R_i$, and $f_{2}(S_j)$ denotes the over-cover ratio of the shift $S_j$ as defined in equation (13).

The average results of 20 runs for the EDA_C are also compiled in Table 2. Compared with the GAFE, solution of the EDA_C has 1.70% more shifts in terms of total shift number, and is 4.11% more expensive in terms of total cost on average. The EDA_C’s results are still slightly worse than those of the GAFE, but much better than those of the EDA_0. This demonstrates the importance of the over-cover ratio.

To further demonstrate the robustness of the EDA_C, we summarise its results of 20 runs in Table 3. Columns 2–5 show the shift number of 20 runs in terms of Average (rounded), minimum, maximum and difference between maximum and minimum respectively, where the number in parentheses denotes the number of runs reached these values respectively. Moreover, the last four columns show the solution costs of 20 runs in terms of average, minimum, maximum and standard deviation. The runtime is approximately 10–30 seconds per data instance.

Table 3 Summary of results of 20 runs with different random seeds for the EDA_C

<table>
<thead>
<tr>
<th>Data</th>
<th>Ave. (# of runs)</th>
<th>Min. (# of runs)</th>
<th>Max. (# of runs)</th>
<th>Max-Min</th>
<th>Ave. Cost (h:m)</th>
<th>Min. Cost (h:m)</th>
<th>Max. Cost (h:m)</th>
<th>Std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>24(17)</td>
<td>24(17)</td>
<td>25(3)</td>
<td>1</td>
<td>172:52</td>
<td>167:17</td>
<td>178:33</td>
<td>2.91</td>
</tr>
<tr>
<td>P2</td>
<td>24(20)</td>
<td>24(20)</td>
<td>24(20)</td>
<td>0</td>
<td>184:02</td>
<td>174:12</td>
<td>187:00</td>
<td>4.69</td>
</tr>
<tr>
<td>P3</td>
<td>35(13)</td>
<td>34(5)</td>
<td>36(2)</td>
<td>2</td>
<td>271:46</td>
<td>259:28</td>
<td>283:47</td>
<td>6.62</td>
</tr>
<tr>
<td>P4</td>
<td>49(17)</td>
<td>48(3)</td>
<td>49(17)</td>
<td>1</td>
<td>373:15</td>
<td>354:48</td>
<td>375:46</td>
<td>5.07</td>
</tr>
<tr>
<td>P5</td>
<td>66(16)</td>
<td>64(2)</td>
<td>67(2)</td>
<td>3</td>
<td>503:36</td>
<td>491:33</td>
<td>508:07</td>
<td>3.37</td>
</tr>
<tr>
<td>P6</td>
<td>66(9)</td>
<td>64(2)</td>
<td>67(8)</td>
<td>3</td>
<td>507:05</td>
<td>491:43</td>
<td>515:09</td>
<td>6.36</td>
</tr>
<tr>
<td>P7</td>
<td>79(11)</td>
<td>77(1)</td>
<td>80(4)</td>
<td>3</td>
<td>547:06</td>
<td>528:42</td>
<td>580:14</td>
<td>14.92</td>
</tr>
<tr>
<td>P8</td>
<td>78(7)</td>
<td>75(1)</td>
<td>79(8)</td>
<td>4</td>
<td>566:09</td>
<td>551:58</td>
<td>579:28</td>
<td>8.51</td>
</tr>
<tr>
<td>P9</td>
<td>102(6)</td>
<td>97(1)</td>
<td>104(5)</td>
<td>7</td>
<td>756:34</td>
<td>731:43</td>
<td>772:40</td>
<td>13.05</td>
</tr>
<tr>
<td>P10</td>
<td>107(15)</td>
<td>104(1)</td>
<td>107(15)</td>
<td>3</td>
<td>807:48</td>
<td>764:17</td>
<td>819:18</td>
<td>19.00</td>
</tr>
<tr>
<td>P11</td>
<td>94(10)</td>
<td>90(1)</td>
<td>95(4)</td>
<td>5</td>
<td>694:29</td>
<td>673:52</td>
<td>716:26</td>
<td>11.39</td>
</tr>
</tbody>
</table>

Table 3 shows that the shift numbers found in the 20 runs by the EDA_C for most cases vary within 3 shifts, meanwhile, there is no remarkable variation between the 20 runs in terms of solution costs. We therefore deem the results of the EDA_C are robust.
4.4 Results of an improved EDA with weighted rules

To enhance the performance of our EDA, we propose a further improved version called EDAW. The basic idea is to use a better set of initial schedules as the input. Based on our experience and intuition, each of the six rules should have different influence on the formation of the near-optimal schedule. Therefore, we weight each rule during the process of forming initial rule string. In this way, the rule with higher weight would have more chance to be selected. This implicates that a relatively good population can be generated during the initialisation phase.

The average results of 20 runs for the EDAW’s are also illustrated in Table 2. From the table, it can be seen that the EDAW outperforms the GAFE in terms of both shift number (with the average RPD being –6.44%) and total cost (with the average RPD being –6.37%). One can also conclude that the EDAW performs much better than the EDA, which demonstrates the significance of considering different weights for rules.

Again, to verify the robustness of the proposed EDAW, we summarise the results of the 20 runs in Table 4 which shows that the shift numbers found in the 20 runs for 10 out of 11 cases vary within 1 shift. In addition, there is no remarkable variation between the 20 the runs in terms of solution costs, which indicates that, the robustness of the EDAW.

Table 4 Summary of results of 20 runs with different random seeds for the EDAW

<table>
<thead>
<tr>
<th>Data</th>
<th># of shifts</th>
<th>Cost (h:m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ave. (# of runs)</td>
<td>Min. (# of runs)</td>
</tr>
<tr>
<td>P1</td>
<td>23(20)</td>
<td>23(20)</td>
</tr>
<tr>
<td>P2</td>
<td>24(17)</td>
<td>24(17)</td>
</tr>
<tr>
<td>P3</td>
<td>31(20)</td>
<td>31(20)</td>
</tr>
<tr>
<td>P4</td>
<td>45(18)</td>
<td>45(18)</td>
</tr>
<tr>
<td>P5</td>
<td>62(19)</td>
<td>61(1)</td>
</tr>
<tr>
<td>P6</td>
<td>65(11)</td>
<td>64(9)</td>
</tr>
<tr>
<td>P7</td>
<td>68(18)</td>
<td>68(18)</td>
</tr>
<tr>
<td>P8</td>
<td>70(20)</td>
<td>70(20)</td>
</tr>
<tr>
<td>P9</td>
<td>87(20)</td>
<td>87(20)</td>
</tr>
<tr>
<td>P10</td>
<td>99(14)</td>
<td>99(14)</td>
</tr>
<tr>
<td>P11</td>
<td>86(14)</td>
<td>85(3)</td>
</tr>
</tbody>
</table>

According to the computational results, the EDAW is a robust algorithm and performs much better than the GAFE in terms of both number of shifts and total cost for all data instances. Therefore, we suggest that the proposed EDAW could be applied as an alternative approach to well solve the driver scheduling problem.

5 Conclusions

This paper presents a Bayesian networks-based EDA approach for driver scheduling. This is the first time that Bayesian networks are applied to the well-studied field of transportation driver scheduling, in which an novel idea is proposed to address the
problem about how to implement explicit learning from past solutions. Unlike most existing rule-based approaches, the new approach has the ability to build schedules by using flexible, instead of fixed rules, at the different stages of schedule building process. Computational results from some real-world problems in China demonstrate the strength and availability of the proposed approach.

Although the work presented here is on public transport driver scheduling, it appears that the main idea of the approach may be applied to other scheduling problems where the schedules are built step by step by some domain-specific rules. It also appears that this research would shed some light on how to incorporate explicit learning into scheduling algorithms and thus may be of some interest to practitioners and researchers in areas of scheduling and evolutionary computation.

Acknowledgements

The work was supported by National Natural Science Foundation of China (Grants No. 70971044 and 71171087) and the Major Program of National Social Science Foundation of China (Grant No. 13&ZD175).

References


An estimation of distribution algorithm for public transport driver scheduling


