BUS SCHEDULING AS A GRAPH COLORING PROBLEM

The fundamental vehicle-scheduling problem (VSP) is usually formulated as a matching problem for which there exists a polynomial algorithm. The formulation of VSP as a graph coloring problem may seem not to be practical since a graph coloring problem is NP-hard and it is not a good idea to solve a polynomial problem using a non-polynomial algorithm. However, no polynomial algorithms have been found for generalizations of VSP and hence the graph coloring formulation offers good approximation algorithms for their solution.

This paper was supported by VEGA as a grant No. 1/0490/03.

1. Introduction


Standard vehicle scheduling is a polynomial problem, which can be polynomially reduced to a matching problem. However, its generalizations are no longer polynomial. This paper brings a graph coloring formulation of a fleet minimization problem. Such formulation is of no use for standard scheduling, but for its generalizations (for two bus scheduling problem) it offers a non-polynomial algorithm which can be used as a heuristic, since it offers a good feasible solution in any stage of computing.

Trip is a travel from a starting point to a finishing point of a route and is considered to be an elementary amount of the work of a bus.

We will say that the trip $s_j$ precedes the trip $s_i$ and we will write $s_i < s_j$ if the trip $s_j$ can be linked after the trip $s_i$ into a running board for one bus. If $s_j = (dp_j, ap_j, dt_j, at_j)$, $s_i = (dp_i, ap_i, dt_i, at_i)$, then it holds $s_j < s_i$ if and only if

$$dt_j \geq at_i + M(dp_i, dp_j),$$

where $M(dp_i, dp_j)$ is the travel time from arrival place of $s_i$ to departure place of $s_j$. Relation $<$ is irreflexive and transitive.

Running board, or running board of a bus is an arbitrary nonempty sequence $T = s_1, s_2, ..., s_n$ of trips with property:

$$s_i < s_j \ldots < s_m$$

The number $m$ will be used for length of running board $T$. We will write $s_i \rightarrow s_j$, if both trips $s_i, s_j$ are provided by the same bus and the trip $s_j$ is linked immediately behind the trip $s_i$. Note that $s_i \rightarrow s_j$ implies $s_i < s_j$.

For any given set $S = \{s_1, s_2, ..., s_n\}$ of trips with precedence relation $<$ we can construct a bus schedule. Bus schedule of the set $S$ of trips is a set of running boards $O = \{T_1, T_2, ..., T_k\}$ of the form

$$T_1 = s_{1,1} \rightarrow s_{1,2} \rightarrow ... \rightarrow s_{1,n(1)} $$

$$T_2 = s_{2,1} \rightarrow s_{2,2} \rightarrow ... \rightarrow s_{2,n(2)} $$

$$...$$

$$T_k = s_{k,1} \rightarrow s_{k,2} \rightarrow ... \rightarrow s_{k,n(k)} $$

so that every trip of the set $S$ occurs exactly in one running board of $O$.

To every bus schedule $O = \{T_1, T_2, ..., T_k\}$ the number $C(O)$ - the cost of bus schedule $O$ is assigned. We will say that the cost $C(O)$ is separable if it can be expressed as a sum of costs of all running boards $T_1, T_2, ..., T_k$, i. e.

$$C(O) = \sum_{i=1}^{k} c(T_i),$$

where $c(T_i)$ denotes the cost of running board $T_i$. In most simple cases the cost of running board $T = s_1 \rightarrow s_2 \rightarrow ... \rightarrow s_m$ is the sum of all linkage costs:

$$c(T) = \sum_{i=1}^{m-1} c(s_i, s_{i+1}),$$
The linkage cost denoted \( c(s_i, s_j) \) represents mainly dead mileage expenses, however, it may include waiting costs and other penalties as well.

In this case we will say that the cost \( c(T) \) is linear. We will say that the cost \( C(O) \) of bus schedule \( O \) is linear, if \( C \) is separable and the cost \( c \) of running board is linear.

2. Two fundamental bus scheduling problems

The general goal of vehicle scheduling optimization is to find the bus schedule \( O \) with minimum cost \( C(O) \). However, it showed useful to decompose the optimization process into two stages - minimization of the number of used buses and the minimization of the bus schedule cost provided the minimum number (or any feasible fixed number) of vehicles is used. This approach follows from practical experience when bus operators demand in most cases the bus schedule with a minimum number of vehicles.

There are two classical fundamental bus scheduling problems:

**Fundamental problem I (FP I)** To find a bus schedule with a minimum number \( k_0 \) of running boards (i.e. with a minimum number of vehicles).

**Fundamental problem II (FP II)** From all bus schedules with the given number \( k \) of running boards to find a bus schedule with a minimum total cost. (In most cases \( k = k_0 \), where \( k_0 \) is the minimum number of vehicles obtained by FP I.)

3. Mathematical model for linear bus schedule cost

For linear bus schedule cost \( C(O) \) and \( k = k_0 \) we have the following mathematical model.

Let’s denote

\[
d_d = \begin{cases} \end{cases} \begin{align} \left( c(s_i, s_j) \right), & \text{for } i, j \text{ so that } s_i < s_j \\ \infty, & \text{otherwise} \end{align} \quad (6)\]

Let \( x_{ij} \in \{0, 1\} \) be a decision variable. If \( x_{ij} = 1 \) and \( s_i < s_j \) it means that the trip \( s_j \) is linked after the trip \( s_i \) in a running board and \( d_d = \infty \) represents the corresponding linkage cost. If \( x_{ij} = 1 \) and \( s_i < s_j \) it means that the trip \( s_j \) is the last trip in another (or maybe the same) running board. In this case \( d_d = \infty \). So the sum \( \sum_{i=1}^{n} \sum_{j=1}^{n} d_d x_{ij} \) includes all linkage costs and as many items \( d_d \), \( x_{ij} = \infty \) as the number of running boards represented by variables \( x_{ij} \).

The mathematical model for FP II can be formulated as follows:

\[
\begin{align} \sum_{i=1}^{n} x_{ij} &= 1 \quad \text{for all } i = 1, 2, ..., n \\ \sum_{j=1}^{n} x_{ij} &= 1 \quad \text{for all } j = 1, 2, ..., n \\ x_{ij} &\in \{0, 1\} \end{align} \quad (8) \]

Conditions (8) say that each trip can have only one predecessor and only one successor. Last formulation is a matching problem for which we have several effective polynomial algorithms. The same model works for FP I, too, but it can be even simplified by setting

\[
d_d = \begin{cases} 0, & \text{for } i, j \text{ such that } s_i < s_j \\ \infty, & \text{otherwise} \end{cases} \quad (9) \]

For \( k > k_0 \) the model for FP II can be several ways – one of them is the following. Replace \( \infty \) by a great number \( K \) in (6) and add the constraints \( \sum_{i=1}^{n} \sum_{j=1}^{n} d_d x_{ij} \geq kK \) to (8). The graph theory formulation of FP II with given number \( k \) of running boards leads to the minimum cost flow problem with the given flow magnitude equal to \( k \).

4. Heuristic approaches for regular bus schedule cost

There are many practical cases when the cost of bus schedule is separable but no longer linear. Thus for return bus scheduling problem, when the cost of bus schedule \( T = s_1 \rightarrow s_2 \rightarrow ... \rightarrow s_m \) includes travel cost from arrival place of \( s_{m-1} \) to departure place of \( s_1 \)

\[
c(T) = \sum_{i=1}^{m-1} c(s_i, s_{i+1}) + c(s_1, s_{m-1}) \quad (10) \]

the FP II is NP-hard. Other example of similar problems is a multiple depot bus scheduling problem, bus scheduling with special conditions such as constraints for working time, safety break, meal break etc. For none of them any polynomial algorithm has been found till now.

The FP I doesn’t depend on the form of objective function, so the matching model gives us the solution with a minimum number of buses. We take this solution as the starting one for one of following neighbourhood search heuristic procedures:

- horizontal splitting – neighbourhood obtained from existing solution by combining trips of two running boards
- vertical splitting – neighbourhood obtained by combining heads and tails respectively bodies and mids of all running boards from the existing bus schedule.

The vertical splitting leads to the multiple application of matching algorithm. From many practical experiences it follows that the mentioned procedures are very successful for separable bus schedule cost.
5. Graph coloring model for FP I

Let \( S \) be a set of trips with precedence relation \(<\). We will say that the trips \( s_1 \in S, s_2 \in S \) are incompatible if \( s_1 < s_2 \) and \( s_2 < s_1 \). Otherwise we will say that the trips \( s_1, s_2 \) are compatible. Let \( G = (V, E) \) be a graph with vertex set \( V = S \) and edge set \( E \) defined by

\[
E = \{(u, v) | u \in V, v \in V, u < v \}.
\]

(11)

In other words the edge set \( E \) is the set of all incompatible pairs of trips from \( S \).

Let \( T = s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_m \) be a running board with the trips from \( S \). Then for every couple \( s, s_i, i = 1, 2, \ldots, m \), \( i \neq j \) it holds \( s_1 < s_j \) or \( s_j < s_1 \) and consequently \( (s_1, s_j) \in E \) - the set \( V_T \) of all trips from running board \( T \) is an independent subset of vertices in graph \( G \).

On the other hand the given independent subset of trips \( V_T \subseteq V \), for arbitrary pair \( s_j, s_j \in V_T \), it holds \( s_j < s_j \) or \( s_j < s_j \) is a complete ordering of finite set \( V_T \) and hence the trips from \( V_T \) can be ordered into a linear ordered sequence - running board \( T \). In the terminology of graph coloring problem the subset \( V_T \subseteq V \) is a running board if and only if all vertices from \( V_T \) can be colored by the same color. Then FP I can be formulated as follows: To solve FP I means to find a coloring of the graph \( G \) with a minimum number of colors.

6. Exact graph coloring algorithm

The following exact graph coloring algorithm comes from the outstanding Demel’s book [4]. Suppose \( V = \{1, 2, \ldots, |V|\} \) \( n = |V| \). Let \( V_x \) be the set of all neighbours of the vertex \( x \) in graph \( G \).

Set \( P(x) := V_x \cap \{1, 2, \ldots, x - 1\} \) (12)

\[
F(x) := \min_i \{1 \leq i, \forall j \in P(x) i \neq B[j]\}
\]

(13)

\[
G(x) := \min_i \{1 \leq i, B(x) < i, \forall j \in P(x) i \neq B[j]\}
\]

(14)

\( F(x) \) is the lowest feasible color number for vertex \( x \). Provided the vertex \( x \) is colored with color \( B(x) \), \( G(x) \) is the lowest feasible color number greater than \( B(x) \) which can be used for vertex \( x \).

- **Step 0.** Set \( B[1] := 1 \) and sequentially for every \( x = 2, 3, \ldots, n \) \( B[x] := F(x) \).
- **Step 1.** Set \( F_{MAX} := \max_x B[x] \).
- **Step 2.** Find in array \( B[] \) the lowest \( y \) such that \( B[y] = F_{MAX} \).
- **Step 3.** Set \( x := \max_{x \in P(y)} \).
- **Step 4.** If \( x = 1 \), STOP. Chromatic number of graph \( G \) is \( \chi(G) = F_{MAX} \) and the corresponding graph coloring of \( G \) is in array \( RECORD[] \).
- **Step 5.** If \( G(x) \approx F_{MAX} \) or if \( G(x) > (\max_{1 \leq i \leq n} B[i]) + 1 \), set \( x := x - 1 \) and goto Step 4.

Otherwise set \( B[x] := G(x), z := x + 1 \).

- **Step 6.** \( B[z] := F(z) \).

If \( B[z] \approx F_{MAX} \), set \( y := z \) and goto Step 3.

If \( y < n \) set \( z := y + 1 \) and repeat Step 6.

If \( z = n \) we have a new better solution. Goto Step 1.

The bad news is that this algorithm is not polynomial and there is no hope that it finishes in reasonable time even for the smallest real world instances. The good news is that after executing Step 1 at least once we have a feasible (but not necessary optimum) solution in the array \( RECORD[] \), so the algorithm can be interrupted in any time (after executing Step 1. at least once) and the coloring stored in array \( RECORD[] \) can be used as a suboptimum solution. This algorithm was implemented in C-language and practical experiences showed that it was able to find optimum relatively fast, but there still remained plenty of computing to confirm that there is no better solution. Since the matching model for FP I is exact and very fast, in this stage this approach is of no practical importance.

There are graph coloring algorithms with better performance e.g. methods based on constraint programming in [5], but the investigation how to modify them for two bus type problem is not finished yet.

7. Two bus type problem

Suppose we have two types of buses say type 1 and type 2 with different capacities \( c_1 > c_2 \) and different costs of running board \( c_1 > c_2 \). The set \( S \) of all trips consists of two disjoint subsets \( S = S_1 \cup S_2, S_1 \cap S_2 = \emptyset \). The trips from \( S_1 \) can be performed only by buses of the type 1, because they need higher capacity. The trips from \( S_2 \) can use both types of buses.

A type of the bus has to be assigned to every running board \( T = s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_m \). If for some the trip \( s_j \) from running board \( T \) it holds \( s_j \in S_1 \) the running board \( T \) has to be accomplished by a bus of the type 1 (otherwise \( T \) is infeasible). If the running board \( T = s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_m \) consists only of elements of the subset \( S_2 \) it is possible to assign to it both types of buses. However, the better solution is to use smaller type 2 because it is less expensive. So we define the cost \( c(T) \) of running board \( T \) as follows

\[
c(T) = \begin{cases} c_1(T) & \text{if } T \text{ contains least one trip from } S_1 \\ c_2(T) & \text{if all trips from } T \text{ are from } S_2 
\end{cases}
\]

(15)

The cost of bus schedule \( O = \{T_1, T_2, \ldots, T_k\} \) is separable, i.e. it holds (4).

The general two bus type problem is to find a bus schedule with a minimum cost \( C(O) \).

The original idea was the same as in a single bus type case described in section 4 of this paper - first to solve FP I (to find the bus schedule with a minimum number of buses regardless of the bus type) and then to use heuristics for minimization of the cost \( C(O) \). But the described heuristics working well in most of a single bus type cases failed in a two-bus type problem. The main reason is probably in the fact that there are not enough simple
paths from solution with a larger number of vehicles of the type 1 to the solution with a lower number. Another reason is that the objective function \( C(O) \) doesn’t express the “distance” of the existing solution to the solution with a lower number of buses of type 1 and hence the neighbour searching procedure cannot distinguish which solution is “closer” to diminishing the number of vehicles of the type 1. If we start from the existing solution with an optimum fleet structure we could avoid the described problems.

There are several possible analogies of FP I for two-bus type scheduling.

**Fundamental Two Bus-type Problem a (FTBP Ia).** For \( k_1 \) - the minimum number (or any given feasible number) of buses of the type 1 to find a bus schedule with a minimum number \( k \) of all buses.

**Fundamental Two Bus-type Problem b (FTBP Ib).** For \( k \) - the minimum number (or any given feasible number) of all buses to find a bus schedule with a minimum number \( k_1 \) of buses of the type 1.

Formulation of the analogy of FP II. for the two-bus type case is the following:

**Fundamental Two Bus-type Problem II (FTBP II).** From all bus schedules with given numbers \( k_1, k_2 \) of running boards of the type 1 and type 2 to find a bus schedule with a minimum cost \( C(O) \).

Both FTBP Ia and FTBP Ib seem to be NP-hard, but I didn’t succeed to prove this nor to find mentioned problems can be formulated as a known graph theory problem - list coloring problem.

The list coloring problem is the following generalization of a graph coloring problem: Let \( G = (V, E) \) be a graph, \( V = \{v_1, v_2, \ldots, v_n\} \), let \( C_1, C_2, \ldots, C_n \) be sets of colors where \( C_i \) is a set of feasible colors for vertex \( v_i \) (list of allowed colors for \( v_i \)). The problem is to find a feasible coloring - to assign for every \( v_i \in V \) a color from \( C_i \) such that no two adjacent vertices are assigned the same color.

We can define a further generalization of the list coloring problem - the minimum list coloring problem: Given a graph \( G = (V, E) \) and a list of allowed \( C_1, C_2, \ldots, C_n \) to find a feasible coloring with minimum total number of used colors.

Let \( G = (S_1 \cup S_2, E) \) be the graph with vertex set \( S_1 \cup S_2 \) and edge set \( E \) containing all incompatible pairs of different trips from \( S_1 \cup S_2 \). By solving the FP I for the set of the trips \( S_i \) we obtain \( k_1 \) - the minimum number of large buses of the type 1. For vertices \( v_i \in S_i \) set \( C_i = \{1, 2, \ldots, k_1\} \) for set \( C_j = \{1, 2, \ldots, n\} \). Then to solve FTBP Ia means to solve the minimum list coloring problem in graph \( G \) with list of allowed colors \( C_1, C_2, \ldots, C_n \).

The following modification of the exact graph coloring algorithm was designed for the mentioned special case of minimum list coloring problem.

- **Step I.** Set \( G_i = (S_i, E_i) \) were \( E_i \) is the set of all incompatible pairs of different trips from \( S_i \), set \( G = (V, E) \) where \( V = S_1 \cup S_2 \) and where \( E \) is the set of all incompatible pairs of different trips from \( V \). Sort the vertex set \( V \) so that vertices from \( S_1 \) proceed all vertices from \( S_2 \).
- **Step II.** Solve FP I for the set of vertices \( S_1 \) to find \( k_1 \).
- **Step III.** Use exact graph coloring algorithm for the graph \( G_1 \) and stop it as soon as you find the first feasible solution with \( k_1 \) colors. Keep result coloring in array \( \text{RECORD}_1 \).
- **Step IV.** Set \( P(x) := V_s \cap [1, 2, \ldots, x - 1] \)

\[
F(x) := \min\{i | i \in C_x, 1 \leq i, \forall j \in P(x), i \neq B[j]\}
\]

\[
G(x) := \min\{i | i \in C_x, B[i] < i, \forall j \in P(x), i \neq B[j]\}
\]

\( F(x) \) is the lowest feasible color number for vertex \( x \) with respect to color list \( C_x \). Provided the vertex \( x \) is colored with color \( B[x] \), \( G(x) \) is the lowest feasible color number greater than \( B[x] \) which can be used for vertex \( x \) with respect to color list \( C_x \). Functions \( F(x) \), \( G(x) \) are set minimums. Remember that if the set is empty the corresponding minimum is \( +\infty \). Since \( C_x = \{1, 2, \ldots, n\} \) for \( x \in S_1 \) then \( F(x) \leq \infty \) for all \( x \in S_2 \).

- **Step V.** Set \( B[x] := \text{RECORD}_1[x] \) for all \( x \in S_1 \) and sequentially for every \( x \in S_2 \) \( B[x] := F(x) \).

(Let’s note that it can never happen \( F(x) = \infty \) for \( x \in S_2 \).)

- **Step VI.** Set \( F_{\text{MAX}} := \max_{1 \leq i \leq n} \{B[i]\} \).

Copy array \( B \) into array \( \text{RECORD}_2 \).

- **Step VII.** Find in array \( B \) the lowest \( y \) such that \( B[y] = F_{\text{MAX}} \).

- **Step VIII.** Set \( x := \max_{x \in P(y)} \{k\} \).

- **Step IX.** If \( x = 1 \), STOP. Chromatic number of graph \( G \) is \( \chi(G) = F_{\text{MAX}} \) and the corresponding optimum graph coloring of \( G \) is in array \( \text{RECORD}_2 \).

- **Step X.** If \( G(x) \geq F_{\text{MAX}} \) or if \( G(x) > (\max_{1 \leq i \leq n} \{B[i]\} + 1) \), set \( x := x - 1 \) and Goto Step IX.

Otherwise set \( B[x] := G(x), z := x + 1 \).

- **Step XI.** \( B[z] := F(z) \). If \( B[z] \geq F_{\text{MAX}} \), set \( y := z \) and Goto Step VIII.

If \( z < n \) set \( z := z + 1 \) and repeat Step XI.

If \( z = n \) we have a new better solution. Goto Step VI.

Similarly to the exact coloring algorithm this algorithm will not finish in reasonable time, but it offers a feasible good solution in array \( \text{RECORD}_2 \) in any time after executing Step V at least once.

References


