Size Limited Iterative Method (SLIM) for Train Unit Scheduling

Pedro J. Copado-Mendez, Zhiyuan Lin, Raymond S. K. Kwan

University of Leeds
Leeds, LS2 9JT, United Kingdom
scspjc@leeds.ac.uk, z.lin@leeds.ac.uk, r.s.kwan@leeds.ac.uk

Abstract

In this work, we are developing a hybrid method driving a core ILP solver with an iterative heuristic for the train unit scheduling optimization problem, which is formulated as an integer multi-commodity flow problem. This approach aims at reducing the problem to a minimal size but still retaining all the essential components for an optimal solution. This research is focussed on two aspects, (i) creation of an initial feasible solution, (ii) iterative improvement on a minimal sized problem extracted from incumbent best solution(s).

1 Introduction

A train unit is a self-propelled non-splittable fixed set of train carriages, which is the most commonly used passenger rolling stock in the UK and many other European countries. Train unit scheduling optimization (TUSO) refers to the planning of how timetabled trips are covered by train units in one operational day [4]. TUSO is very important because of the high costs associated with leasing, operating and maintaining a fleet. It involves distributing limited rolling stock resources in the optimal way, by coupling\decoupling train units and\or running them empty, to meet passenger demands. The main objectives are to minimise the number of units used and\or the operational cost.

In the current literature, the TUSO or similar problems are usually formulated as integer multicommodity flow (IMCF) problems [1], and solved either by linear programming (LP) based heuristics [2, 3] or exact methods [6]. The former are efficient but the solution quality may not be guaranteed, while the latter are exact but their capability is limited by the growth of problem size. Observing that the final optimal solution only contains a very small subset of the solution space (arcs) in terms of the original data, we propose a hybridized method combining heuristics and exact methods, trying to take advantage of both of an exact and a pure heuristic to achieve high quality near-optimal solutions. Compared with many commonly used (meta-)heuristics, our iterative hybridized approach regards an exact solver as a black box and mainly operates on the solution space for the solver. In every iteration, the solver only solves a reduced small-sized problem (a subset of all arcs) fast and comfortably, followed by an evaluation and modification of the subset of arcs to be fed to the next iteration such that the objective will be no worse. Finally, the subset of arcs will converge to what is very close to an optimal solution. The hybrid framework design is challenging since there is a huge number of possible combination subsets of arcs that can form a feasible solution, and how the hybrid framework will pick up these optimal ones from a rough initial feasible solution through a series of iterations is non-trivial. Two particular difficult issues are discussed: (i) the generation of an initial feasible solution (Section 3.1); (ii) the iterative heuristic process (Section 3.2). Computational experiments regarding (i) will also be reported in Section 4.

2 Problem Description

Given a set of daily timetabled train trips, each with an origin, a destination, departure time, arrival time and passenger demand, and a set of train units, each with a given number of seats, the problem aims at covering all trips using the minimum number of train units and reducing the operational cost. Two or

more units can be coupled up to meet a train with a high passenger demand. Different types may be incompatible to couple and coupling/decoupling operations may not be allowed at some locations. This problem is transformed to a network framework based on a Directed Acyclic Graph (DAG) denoted by $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where the node set $\mathcal{N} = \mathcal{N} \bigcup \{s, t\}$ is the set of train services, and s and t are the *source* and *sink* node. The arc set is defined as $A = A \bigcup A_0$, where $A = \{(i, j)|i, j \in N\}$ is the *connection-arc* set and $A_0 = \{(s, j)|j \in N\} \cup \{(j, t)|j \in N\}$ is the *sign-on/off* arc set. Finally, an $s - t$ path in $\mathcal{G}$ represents a sequenced daily workload (the train nodes in the path) and the flow on it is the number of units used for serving those trains. Details of this DAG representation can be found in [5].

3 Solution Approach

Our approach, a Size Limited Iterative Method (SLIM), is based on the idea that, any solution of this problem is composed of a small fraction of the arcs of the original problem. Thus, it would be possible to start from a feasible arc set and progress towards the optimal set. Whilst iteratively, heuristic searches are carried out over all possible subset of arcs until some high-quality solution is achieved in a reasonable time. Some other methods are published for both TUSO [2, 3, 5, 6] and IMCF [1], but the difference between ours and theirs is that an exact ILP core-solver is considered as a black box that it is repeatedly called by SLIM during the process.

SLIM is composed of three main components - (i) a method to generate an initial feasible solution. (ii) an ILP core-solver for solving the IMCF model for small/medium instances. The ILP core-solver is currently being developed in parallel with this work and for more details see the publications [5, 6], (iii) an iterative heuristic controller, where in each iteration the incumbent set of arcs is extended heuristically keeping a small size before the ILP core-solver is applied. As we said previously, this extension is carried out by heuristically selecting arcs from the original graph giving rise to near-optimal solution of the original problem. The reduced sub-problems thus converge very close to the optimality at the final iterations. Roughly speaking, the SLIM framework is as following: (1) Generate an initial feasible solution, which is case dependent. The initial solution is set as the incumbent best solution and is stored in a solution set with limited size S. (2) At each iteration, while some stop criterion is not satisfied: (a) A solution $\bar{s}$ is picked up from S, but the selection is biased to the best solution. (b) A sub-graph $\bar{\mathcal{G}}$ is derived from the solution $\bar{s}$. Therefore $\bar{\mathcal{G}}$ only contains those arcs that belong to $\bar{s}$. (c) $\bar{\mathcal{G}}$ is extended by adding new potential arcs using heuristics, and then a new extended sub-graph $\hat{\mathcal{G}}$ is obtained. (d) Solve $\hat{\mathcal{G}}$ using the ILP core-solver, which gets back a new solution $\hat{s}$. (e) If $\hat{s}$ is better than the incumbent best one, it will be added to S, otherwise it will be rejected. (f) Go to step (2).

Note that there are two challenges or difficulties to face in this research: firstly, an initial feasible solution has to be created as discussed in Section 3.1. Secondly, an effective heuristics will be developed as mentioned in Section 3.2. In addition, we will investigate in how to tune the algorithm parameters.

3.1 Initial Solution

Our approach starts from an initial feasible solution, so that, it is usually enough to create it using the greedy first-in-first-out (FIFO) rule. At each station, this rule locally assigns the train units from the train that first arrives to the train that first departs in a practically workable way. But sometimes the FIFO rule produces such poor solutions that ILP core-solver might get stuck in processing this solution. To deal with this, we applied the well-known Divide & Conquer (DC) strategy which lies in three steps: (1) split the original set of trains into two balanced subsets by a time point, making a distinction between morning trains set and evening train set, (2) solve each set separately and (3) merge both solutions in one. This approach often outperforms the FIFO, but it is still unclear where the best time point is and how to distribute those trains crossing it in order to minimize the initial fleet size.

3.2 Heuristics

As we said in each iteration our algorithm systematically produces a new $\hat{\mathcal{G}}$ graph adding new arcs from the original graph $\mathcal{G}$ using some heuristic as criteria. In the following these heuristics are briefly outlined: **Location**: arcs are chosen biased to those locations with more trips, since there are further scope of improvement. Another manner is when it is intended to produce new solutions very similar. To do this, arcs relevant to locations appearing more frequently in the current solution are more likely to be chosen. **Time**: the peak times are the intervals where there are more simultaneous trips, so the arcs that end or start in these interval are more likely to be chosen. **Length**: the length of an arc is defined as the difference of time in minutes between the trains linked by this arc. Logically, it is expected that a good solution contains rather short arcs and only a few long arcs, thus, the selection of arcs is biased to short arcs.

4 Experiments

We are testing our approach on a large-sized real-world dataset from a UK operator. This case consists of 500 trains and 2 compatible types, consequently, the direct acyclic graph arisen contains 500 nodes, around 20000 arcs and a lot of locations banned for coupling/decoupling, giving rise to the corresponding integer IMCF problem of 2 commodities, where the ILP solver alone was not able to solve it. For this case, we have faced with the generation of an initial high-quality solution (framework: (1)). This challenge requires special attention because of the ILP solver will spend too much time for solving those sub-problems derived from the initialization provided by the FIFO method (framework: (a)-(d)). To deal with it, we applied the DC strategy as introduced in Section 3.1, with which the objective function declined by almost 20 units. Nevertheless, we are still working on enhancing this upper bound.

5 Conclusions

In this research, which is divided in two main aspects, we propose a hybrid approach for TUSO which starts from an initial feasible solution and iteratively improves it using the heuristic - ILP solver pair. Although we only present partial results concerning the first aspect and the second aspect, leaving them for a future research, we expect to achieve a high-quality solution in a reasonable time and overcome those existing methods.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Solving a real-world train-unit assignment problem. *Mathematical Programming*, 124(1-2):207–231, 2010.

[3] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. A Lagrangian heuristic for a train-unit assignment problem. *Discrete Applied Mathematics*, 161(12):1707–1718, 2013.

[4] Alberto Caprara, Leo Kroon, Michele Monaci, Marc Peeters, and Paolo Toth. Passenger railway optimization. *Handbooks in operations research and management science*, 14:129–187, 2007.

[5] Zhiyuan Lin and Raymond S K Kwan. A two-phase approach for real-world train unit scheduling. *Public Transport*, 6:35–65, 2014.

[6] Zhiyuan Lin and Raymond S K Kwan. A branch-and-price approach for solving the train unit scheduling problem. *Transport Res B-Meth*, 94:97–120, 2016.