Automating Rolling Stock Diagramming and Platform Allocation

Mark Withall[1], Chris Hinde[1], Tom Jackson[2], Iain Phillips[1], Steve Brown[3] and Robert Watson[3]

Abstract

Rolling stock allocation is the process of assigning timetable schedules to physical train units. This is primarily done by connecting together schedules at their terminal locations (known as schedule associations). Platforming allocation is the process of assigning those associations to particular platforms. A simple last-in, first-legal-out algorithm is used for rolling stock allocation that performs comparably to the traditional manual approach but only takes a few seconds as opposed to days or weeks in many manual cases. A simple stochastic hill-climbing approach is used for assigning associations to platforms to provide a conflict-free platform allocation within a few seconds. These two approaches are tested on real train planning problems with excellent results that would allow an expert to rapidly produce optimal or near optimal solutions. The time saving using these approaches can be used by the train planner to try out various options or have greater checking of robustness of the solutions created.

1 Introduction

This paper covers the train planning problems of rolling stock allocation (also known as rolling stock/unit diagramming) and platform allocation (also known as platforming). This usually takes place after a timetable has been created. The traditional manual approach is time consuming and can be error prone. Current tools for producing rolling stock diagrams, such as Dispo[4], fail to take into account sufficient infrastructure constraints and do not provide platforming. Other tools, such as VoyagerPlan[5] are little more than editors for manually creating rolling stock diagrams, with very little checking of the created diagrams.

The approaches presented in this paper are designed to be very fast, as opposed to producing optimal solutions (although they do in many cases). The final optimisation, where necessary, is left to the experienced train planner. The approach speeds up the traditionally time consuming train planning process, which typically gives a more limited amount of time to the problems of rolling stock allocation and platforming than work on, for example, timetable creation. This allows for more time to be spent on trying out different options and testing the robustness of the plan.

The majority of previous work in rail planning has been on timetable generation; Bussieck et al. and Cordeau et al. provide a good overview of work up to the late 1990s, (1) and (2). A considerable amount of work on the problem of rolling stock allocation and particularly platforming has been done by Carey et al., (3), (4), (5), (6), (7).

This work is part of a wider project looking at improving computer support to the complete train planning process; inspired by the work of Watson (8). Work has already been published on a simpler version of the platform allocation, that did not take into account the wider infrastructure constraints of the stations being platformed (9); and on timetable generation (10).

The presented approaches to rolling stock diagramming and platform allocation are tested on two real train planning problems and they perform nearly as well as the results of the manual approach used for developing the production diagrams. This performance is achieved with running times measured in only a few seconds on standard PC hardware; whereas the manually created diagrams may have taken days or even weeks to produce.

The paper starts by describing the basic rolling stock diagramming and platforming problems. It then outlines the data requirements required for Basic Rolling Stock Diagramming and details the simple

[1] Department of Computer Science, Loughborough University, UK
[2] Department of Information Science, Loughborough University, UK
[3] RWA Rail Ltd., Loughborough, UK

[4]http://www.ivembh.de/dispo/index.en.htm
[5]http://www.na.atosorigin.com/en-us/services/industries/public_sector/transport/trusted_in_transport/default.htm

last-in, first-legal-out algorithm used. The performance of the rolling stock allocation algorithm is then explored through two case studies. Both of the case studies are real train planning problems, with no simplifications. The first case study looks at rolling stock allocation on the Essex Thameside region of the UK rail network. The second case study looks at platforming a section of the Stockholm Metro Green Line. The paper finishes with conclusions and future work.

2 Rolling Stock Diagramming

The task of rolling stock allocation is the process of allocating physical train units to schedules within a timetable. The key features of a schedule that are needed for the allocation of rolling stock are:

- origin location
- origin departure time
- destination location
- destination departure time
- schedule stock type
- number of units of rolling stock forming the schedule

In addition to the schedule data, rules for how schedules can be connected together are needed (referred to in this paper as 'schedule association'). These rules are taken from a document called 'The Rules of the Plan' published in the UK by Network Rail[6] (from 2012 this will be known as 'Timetable Planning Rules'). The main rules applicable to this problem are the 'minimum turnround time' (the minimum amount of time a train must wait after arriving at its destination before it can legally start a new journey) and the 'minimum attachment/detachment time' (the minimum amount of time required to add or remove units from a train after it has arrived at its destination).

The problem of rolling stock diagramming can essentially be specified as the creation of a set of schedule associations (turnrounds, attachments and detachments). This set of schedule associations is then trivially converted into a set of partial unit diagrams. These diagrams can then be joined to form diagrams covering the whole day and thus minimise the number of required units.

There is one additional requirement to produce working rolling stock diagrams and that is the addition of empty train schedules (known as ECS moves) to bring in trains from depots at the start and end of diagrams and for connecting movements part-way through diagrams to optimise the use of the rolling stock. ECS moves are also needed when a station exceeds its capacity i.e., if there are four trains in a four platform station, then one of the trains must leave before another arrives, even if there are no scheduled services departing.

3 Platform Allocation

The task of platform allocation is the process of allocating each schedule association, at a given location, in the rolling stock diagrams to a platform. Only platform allocation at terminal locations is considered in this paper; the problem of choosing platforms *en route* is not.

There are two main parts to platform allocation: the first is resolving platform occupation conflicts (trains occupying the same platform at the same time) and the second is resolving routing conflicts within the station throat (train collisions when arriving at or departing the station). The assumption is made that the timetable is valid and, hence, trains will never arrive at the same time on the same line. The following sections detail the research approach taken to create rolling stock diagrams and platform allocations.

4 Rolling Stock Diagramming Algorithm

The starting information required by the rolling stock diagramming algorithm is as follows:

- Schedules
  - Origin locations/time
  - Destination location/time
  - Stock type
  - Train length
- 'Rules of the Plan'

[6] http://www.networkrail.co.uk/browse%20documents/Rules%20Of%20The%20Route/Roprhome.pdf

    - Minimum turnround time
    - Minimum attachment/detachment time
- Running times of empty train moves
- Station capacity

From the timetable, the information used from each schedule is the time and location of the initial departure and the time and location of the final arrival. Additional required information is the stock type that was used to time the schedule and the desired length (in terms of number of units).

In addition to information from the timetable, the ‘Rules of the Plan’ minimum turnround and minimum attachment/detachment times are used. These can be specified as a default value and a number of exceptions based on location, train length, stock type, origin, destination and association type (i.e. passenger to passenger, passenger to empty, etc.).

The running time of empty trains is also required. The list of running times is also used to restrict which empty train movements are allowed in the diagramming process. Some information about the infrastructure is also used: the capacity (number of available platforms) at each terminal location, for making sure there are not too many trains in a station at one time; and the distances between terminal locations are used in the outputting of the results but not by the algorithm.

The basis of the rolling stock diagramming algorithm is a simple last-in, first-legal-out algorithm; each departing schedule will be connected to the most recent arriving schedule that it can be legally connected to under the ‘Rules of the Plan’ restrictions.

There are a number of additions made to this basic algorithm to improve the quality of the resulting diagrams. These improvements are in three main areas: addition of empty train movements (both to and from depots and between terminal locations); connection of diagrams at depots (effectively merging of partial-day diagrams); and a number of ‘repair’ functions that resolve problems with the basic algorithm.

The algorithm has the following ten-step, high-level structure:

1. Last-in, first-legal out simple associations
2. Split long associations
3. Swap crossing associations
4. Create attachment/detachment associations
5. Add start and end depot moves
6. Remove unnecessary depot moves
7. Split long associations
8. Add connections within depots
9. Create diagrams from associations
10. Merge diagrams

The details of each of these steps are described in the following sections.

4.1 Last-in, first-legal-out

The last-in, first-legal-out algorithm simply looks at each departure in turn and connects it to the most recent arrival. If there is no arrival available, it looks at nearby locations for trains that can be run empty to this location. It also keeps track of how many trains are in the station and adds ECS moves to free up space if there are more arrivals than departures. The following pseudo-code describes the initial assignment of associations.

```
• Loop through events at terminal locations (arrivals/departures) in time order
    • Skip if already connected
    • If arrival
        • store
        • if overflows station capacity
            • send earliest unconnected arrival to nearest depot
    • if departure
        • find most recent arrival that can legally be connected
        • if no legal connection available
            • find the nearest location that has an unconnected arrival that can be moved via ECS
              and connected
```

```
                • Include arrivals that have been sent to the depot
```

The reason that last-in, first out was preferred to first-in, first-out, is that, with the addition of ECS moves when a station is full, the associations got gradually longer throughout the day. It was therefore better to use a last-in, first-out algorithm and perform localised repairs to make it first-in, first-out where appropriate.

4.2 Split long associations

The following pseudo-code describes the first repair function that removes excessively long associations.

```
• Examine all associations and remove any that are longer than 1 hour and longer than twice the
  minimum turnround time
```

4.3 Repair crossing turnrounds

The following pseudo-code describes the second repair function that swaps associations that cross in time. Essentially, this performs a localised first-in, first out.

```
• Loop through pairs of connections (ordered by the time of the arriving schedule)
    • Swap departing schedules for any that have a dwell period that falls completely within the
      other, e.g. A1, A2, D2, D1 --> A1, A2, D1, D2
```

4.4 Attachment/Detachments

The following pseudo-code describes the process of creating attachment and detachment schedule associations.

```
• Loop through events at terminal locations
    • Skip generated ECS moves
    • If arrival
        • store if not used or connected to ECS move
    • If departure
        • if unconnected or connected to ECS move
            • find most recent arrival that has the number of available units equal to the number of
              needed units that can be legally connected
            • if still unconnected
                • connect part of the most recent arrival that is longer than needed by the
                  departure
            • if still unconnected
                • find if a combination of shorter arrivals can be joined to form departure
                    • for 2 unit departure, find 1 + 1
                    • for 3 unit departure, find 2 + 1 or 1 + 1 + 1
• Loop through terminal locations
    • If arrival
        • Store if unused or connected to ECS move
    • If departure
        • Find a combination of shorter arrivals than can be joined along with an ECS move to form
          departure
            • for 2 unit departure, find 1 + ECS
            • for 3 unit departure, find 2 + ECS or 1 + 1 + ECS or 1 + ECS
```

This step is in two-parts: the first part looks for attachments and detachments that can be achieved with available arriving schedules; the second looks for attachments that need to include an arriving ECS train to create the full departing train.

4.5 Add ECS depot moves at beginning and end

Once all of the schedules have been connected together with associations, it is necessary to add movements from and to the depot at the beginning and end of the day respectively. The following pseudo-code describes this process.

- Loop through arrivals
    - Send to depot if unconnected
- Loop through departures
    - Get from depot if unconnected

4.6 Repair ECS-surrounded associations

If there is a sequence of associations that start with an ECS to depot and end in an ECS from the depot, then these can be removed and the associations stepped down one (see Figure 1). The following pseudo-code describes this process.

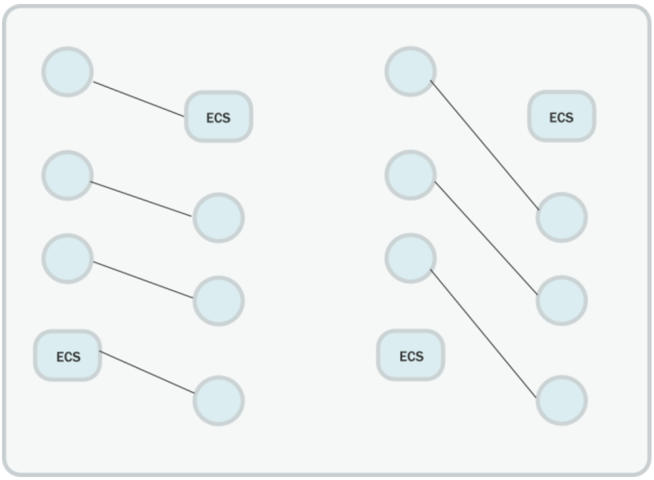

Figure 1 - Repairing ECS surrounded associations

- Loop through arrivals and departures
    - If the departure is an ECS
        - Look ahead for the next departure with an ECS arrival
        - If found and the trains are the same length and an excessively long association won't be created
            - Loop through associations in reverse order, swapping departure with the next departure
            - Delete the two ECS moves

In many cases this repair will save a diagram but it will also reduce the empty mileage of the set of diagrams.

4.7 Add depot connections

In many cases, where a sequence of associations end mid-way through the day, that sequence can be connected to another sequence which starts later to form a single diagram. The pseudo-code for this process is as follows.

- Loop through all arrivals at depots
    - Find first departure that can connect
    - Or find departure that can be connected via an ECS move

4.8 Create diagrams from associations

At this point in the algorithm we have a complete set of associations, with every schedule accounted for. This set of association is converted into a set of diagrams. In many cases a sequence of associations will represent multiple diagrams when all schedules in the sequence are of more than one unit in length. For example, a sequence of associations connecting two-unit schedules will form two diagrams; one for each unit.

4.9 Merge diagrams

The final step in the process is to connect any diagrams that can be joined in the depot that weren't covered by the 'add depot connections' stage due to requiring attachments or detachments of a change of depot. The pseudo-code for this process is described below.

- Loop through diagrams ordered by finishing time
    - Find earliest starting diagram that can be joined to it (directly or via an ECS move) and merge

4.10 Visualisation of results

Once the set of diagrams have been created using the above algorithm, a number of outputs are generated to allow the train planner to evaluate the diagrams. There are three main outputs: a formatted text file containing the timetable for each unit along with mileage information; a pictorial representation of the associations at each terminal location; and a chart showing the sequence of schedules for each unit. Examples of these can be found in the Results sections (Figure 2, Figure 3 and Figure 4).

The association visualisation allows the connections between schedules to be seen clearly at each terminal location. It further allows the attachment and detachments to be seen and the ECS moves that have been inserted (both those to the depot and those between terminal locations).

The unit diagram visualisation allows an overview of how much of the day each unit is running along with how often they are returning to the depot or otherwise running empty. It also clearly shows how much of the fleet is utilised in the off-peak times.

5 Platform Allocation Algorithm

The starting information required by the platform allocation is as follows:

- A set of schedule associations (rolling stock diagrams)
    - Location
    - Arrival times
    - Departure times
- 'Rules of the Plan'
    - Platform reoccupation time
    - Junction margin

From the rolling stock diagrams, the set of associations are required (we are only dealing with platforming a set of diagrams with no attachments or detachments here). For each schedule association we need to know the location it occurs at and the arrival and departure times. The algorithm will then allocate a platform to the arrival and departure schedules (which will be the same platform where possible).

In addition to the rolling stock diagrams, the 'Rules of the Plan' platform reoccupation time is required; this is the amount of time that must be left between one train departing a platform and the next arriving. Also, the junction margin for the station throat is required; this is the time required between two trains crossing the same piece of track.

The algorithm used for allocating platforms is a simple hill-climber with some 'clever' mutation functions. During each iteration of the algorithm a new set of platform allocations is generated, based on the previous iteration, and if it is better than the previous platform allocation it is kept.

There are three parts to the platform allocation hill-climbing algorithm:

- Initial platforming
- Violation check
- Mutation of previous platforming

These parts are described in the following sections.

5.1 Initialise Platforming

The initial platform is constructed by cycling through the platforms at each terminal location as each train arrives. For a three-platform station, the first train would arrive on platform one, the second on platform two and the third on platform three. The fourth train would arrive on platform one again and so on.

- Start cycling through platforms, i.e. first turnround on plat 1, second on 2, third on 3, fourth on 1, ...

5.2 Check Violations

There are four areas checked when evaluating a given platform allocation: junction margin conflict, platform occupation conflicts, route conflicts, and change of platform conflicts.

The junction margin conflicts are tested by checking pairs of schedules travelling through the station to see if their paths cross. If they do, the time between them is compared to the junction margin time.

The platform occupation conflicts are tested by checking pairs of associations at a platform to see if their dwell times overlap (including platform reoccupation time).

The route conflicts are tested by checking that the assigned platform for each schedule can be reached given the arrival line and the departing line can be reached from the assigned platform.

The 'change of platform' conflicts are checked by comparing the assigned platform for both the arrival and departure schedules of an association. In cases where it is not possible to depart from the arrival platform a shunt is required and the location will be exempt from the check.

The following pseudo-code shows the process of checking for platforming violations.

```
• At each terminal location
    • loop through pairs of arrivals and departures
        • if time between is within junction margin
            • add violation if routes cross
    • loop through pairs of dwells at each platform
        • if concurrently in the station (including margin)
            • add violation if same platform
    • loop through arrivals and departures
        • add violation if no route is available to chosen platform
    • loop through turnrounds
        • if arrival and departure are not on same platform
            • add violation if no shunt is required
```

5.3 Optimise (hill-climber)

After the initialisation, the hill-climbing algorithm continually mutates the current platform allocation and then tests for violations. If the new allocation has less violations, then it is retained, otherwise the original allocation continues to be used as the basis for the next set of mutations.

To help avoid getting stuck in local minima, a number of 'clever' mutators are used. Firstly, the schedule having its platform modified is chosen from the list of those that contain violations; this reduces the number of ineffective mutations.

There are four types of mutation: inverting a sequence of platforms; assigning the same random platform to both the arrival and departure platform; assigning a random platform to the arriving schedule of an association; and assigning a random platform to the departing schedule of an association.

The following pseudo-code shows the process of hill-climbing and the creation of new platforming assignments.

```
• loop until no violations
    • store current platforming
    • create a list of schedules with violations
    • select a random schedule
    • if there are margin conflicts
        • with probability 0.2 randomly swap platforms for an interleaved sequence of conflicting
          turnrounds
    • with probability 0.5 assign a random platform to a turnround
```

    - with probability 0.5 assign the random platform to the origin turnround of the above random schedule
    - else assign to destination turnround of above random schedule
- else with probability 0.5 assign a random platform to the above random schedule's origin
- else assign a random platform to the above random schedule's destination
- get list of violations for new platforming
- if less violations keep new platform, else revert back to previous platforming

6 Case Study: Essex Thameside Rolling Stock

The first case study is looking at rolling stock allocation on the Essex Thameside region of the UK rail network[7]. This primarily runs trains between Shoeburyness and Fenchurch Street. The size of the problem for a typical weekday is as follows:

- 355 schedules
- 13 terminal stations
- 4 depots

The restrictions on the rolling stock being use is as follows:

- 1 stock type (class 357 (100 mph EMU units, each with 4-cars))
- trains range from 1 to 3 units long
- attaching and detaching trains is allowed

The services vary greatly throughout the day; with the majority of schedules arriving in Fenchurch Street in the morning and arriving in Shoeburyness in the evening. Trains are run with fewer units during off-peak times. Currently, these schedules are operated using a total of 69 units.

Figure 2 shows the results of the rolling stock allocation for the Essex Thameside problem (red blocks are ECS schedules and blue are passenger schedules). The algorithm produces 70 diagrams (compared with the real diagrams having 69) in a matter of a few seconds. The data preparation time for a problem of this size varies from a few minutes to a few hours; depending on the quality of the source infrastructure data and the complexity of the planning rules being used.

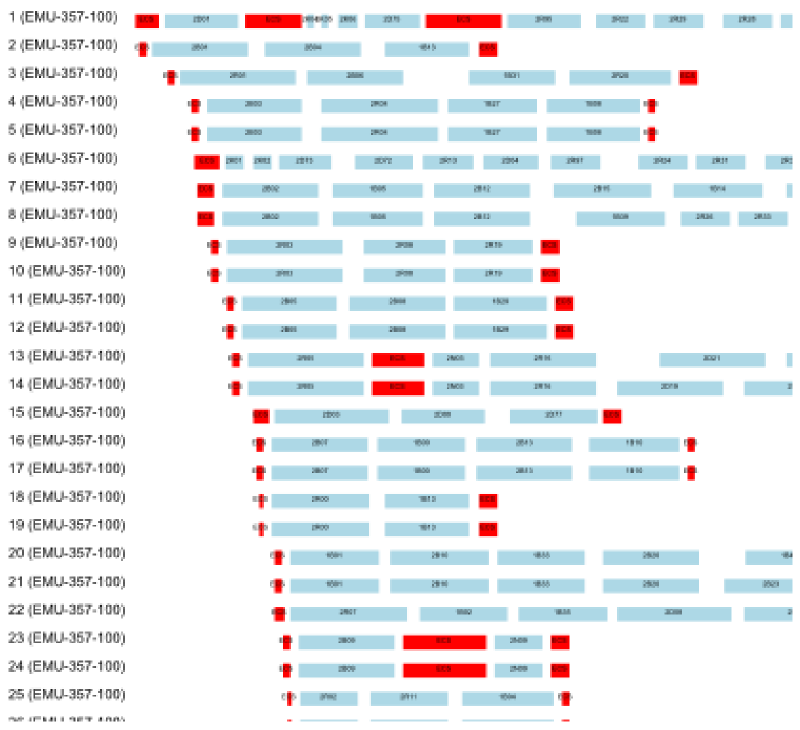

Figure 2 - Unit diagram visualisation for Essex Thameside case study

[7] http://www.dft.gov.uk/pgr/rail/passenger/franchises/essex/

The associations created by the algorithm can be seen in Figure 3 (shown for Southend Central). The red schedules indicate ECS moves that have been added by the algorithm; red lines indicate a minimum turnround, orange up to twice minimum and green longer; blue turnrounds indicate that one of the schedules is provisionally timed; and dotted lines indicate an express train connecting to an ordinary passenger train. The figure shows that all of the associations are the same as those that would most likely have been chosen if the process was being performed manually.

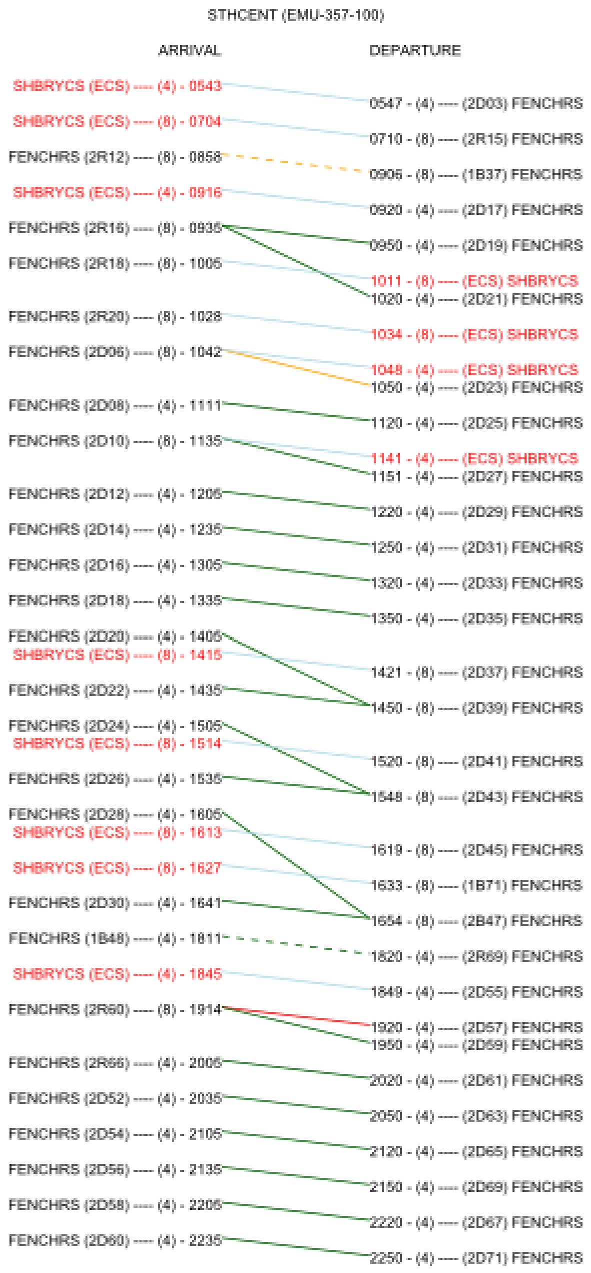
STHCENT (EMU-357-100)
ARRIVAL
DEPARTURE
SHBRYCS (ECS) ---- (4) - 0543
SHBRYCS (ECS) ---- (8) - 0704
FENCHRS (2R12) ---- (8) - 0858
SHBRYCS (ECS) ---- (4) - 0916
FENCHRS (2R16) ---- (8) - 0935
FENCHRS (2R18) ---- (8) - 1005
FENCHRS (2R20) ---- (8) - 1028
FENCHRS (2D06) ---- (8) - 1042
FENCHRS (2D08) ---- (4) - 1111
FENCHRS (2D10) ---- (8) - 1135
FENCHRS (2D12) ---- (4) - 1205
FENCHRS (2D14) ---- (4) - 1235
FENCHRS (2D16) ---- (4) - 1305
FENCHRS (2D18) ---- (4) - 1335
FENCHRS (2D20) ---- (4) - 1405
SHBRYCS (ECS) ---- (8) - 1415
FENCHRS (2D22) ---- (4) - 1435
FENCHRS (2D24) ---- (4) - 1505
SHBRYCS (ECS) ---- (8) - 1514
FENCHRS (2D26) ---- (4) - 1535
FENCHRS (2D28) ---- (4) - 1605
SHBRYCS (ECS) ---- (8) - 1613
SHBRYCS (ECS) ---- (8) - 1627
FENCHRS (2D30) ---- (4) - 1641
FENCHRS (1B48) ---- (4) - 1811
SHBRYCS (ECS) ---- (4) - 1845
FENCHRS (2R60) ---- (8) - 1914
FENCHRS (2R66) ---- (4) - 2005
FENCHRS (2D52) ---- (4) - 2035
FENCHRS (2D54) ---- (4) - 2105
FENCHRS (2D56) ---- (4) - 2135
FENCHRS (2D58) ---- (4) - 2205
FENCHRS (2D60) ---- (4) - 2235
0547 - (4) ---- (2D03) FENCHRS
0710 - (8) ---- (2R15) FENCHRS
0906 - (8) ---- (1B37) FENCHRS
0920 - (4) ---- (2D17) FENCHRS
0950 - (4) ---- (2D19) FENCHRS
1011 - (8) ---- (ECS) SHBRYCS
1020 - (4) ---- (2D21) FENCHRS
1034 - (8) ---- (ECS) SHBRYCS
1048 - (4) ---- (ECS) SHBRYCS
1050 - (4) ---- (2D23) FENCHRS
1120 - (4) ---- (2D25) FENCHRS
1141 - (4) ---- (ECS) SHBRYCS
1151 - (4) ---- (2D27) FENCHRS
1220 - (4) ---- (2D29) FENCHRS
1250 - (4) ---- (2D31) FENCHRS
1320 - (4) ---- (2D33) FENCHRS
1350 - (4) ---- (2D35) FENCHRS
1421 - (8) ---- (2D37) FENCHRS
1450 - (8) ---- (2D39) FENCHRS
1520 - (8) ---- (2D41) FENCHRS
1548 - (8) ---- (2D43) FENCHRS
1619 - (8) ---- (2D45) FENCHRS
1633 - (8) ---- (1B71) FENCHRS
1654 - (8) ---- (2B47) FENCHRS
1820 - (4) ---- (2R69) FENCHRS
1849 - (4) ---- (2D55) FENCHRS
1920 - (4) ---- (2D57) FENCHRS
1950 - (4) ---- (2D59) FENCHRS
2020 - (4) ---- (2D61) FENCHRS
2050 - (4) ---- (2D63) FENCHRS
2120 - (4) ---- (2D65) FENCHRS
2150 - (4) ---- (2D69) FENCHRS
2220 - (4) ---- (2D67) FENCHRS
2250 - (4) ---- (2D71) FENCHRS

Figure 3 - Turnround visualisation for Essex Thameside case study

As sample unit diagram, in text form, is shown in Figure 4.

```
Diagram number: 2 ---- Stock type: EMU-357-100

                           ARR    DEP           Mileage

Shoeburyness Carriage Sid         0411    ECS      0.08

Shoeburyness               0416   0420   2B01     36.55

Fenchurch Street           0528   0540   2B04     73.02

Shoeburyness               0648   0705   1B13    109.49
```

```
Fenchurch Street              0805  0812   ECS   115.55

East Ham Depot                0825       STABLD

East Ham Depot                      1656   ECS   121.61

Fenchurch Street              1708  1715  1B46   158.08

Shoeburyness                  1814  1821   ECS   158.16

Shoeburyness Carriage Sid 1826           STABLD


Mileage:  (Loaded) 145.88    (Empty) 12.28    (Total) 158.16
```

Figure 4 - Example unit diagram for Essex Thameside case study

On a different problem, this time the East Coast Main Line in the UK rail network, the algorithm performs equally well. For this problem there is no attachment and detachment of units, all trains are of a fixed formation, but there are multiple stock types operating. The real diagrams are as follows:

- 11 HST (High Speed Trains)
- 27 Class 410 Electric trains
- 3 Class 180

The diagrams created by the algorithm are as follows:

- 10 HST
- 28 Class 410
- 3 Class 180

This is an equivalent result, as one of the HSTs is operating a Class 410 diagram in practice.

7 Case Study: Stockholm Metro Platform Allocation

The second case study is looking at platforming a section of the Stockholm Metro Green[8]. The Green Line has three branches; this case study looks at the branch running between Skarpnack and Hasselby Strand (known as T17). The size of the problem is as follows:

- 222 schedules
- 8 terminal stations
  - Platform counts
    - Akeshov (AKH) – 3
    - Alvik (ALV) – 2
    - Gullmarsplan (GUP) – 4
    - Hasselby Strand (HAS) – 2
    - Odenplan (ODP) - 2 (requires shunt for turnrounds)
    - Racksta (RAC) – 2
    - Skarpnak (SNK) – 2
    - Vallingby (VBY) - 3
- 3 depots

This results in 48 train diagrams (each train is a fixed formation of 3 C20 stock type units each).

Each terminal location has only one arriving line and one departing line, so there is no choice over which line a schedule arrives on. This means that the stations can be treated independently for platforming.

The relevant ‘Rules of the Plan’ constraints for this network are:

- Junction margin - trains must be at least 2 minutes apart when crossing in the station throat
- Platform reoccupation - trains must arrive at a platform at least 2 minutes after the previous train has departed

[8] http://en.wikipedia.org/wiki/Stockholm_Metro

7.1 *Results of hill-climbing*

The platform allocation algorithm is implement in Perl 5.10, running on a PC with Intel Pentium 4 3.2GHz processor with Windows XP.

Table 1 shows the results of 10 runs of the platforming algorithm on the Stockholm Green Line problem. It is clear that this isn't a very challenging problem and that the algorithm performs consistently well on it. Further examination of the timetable shows that departures are generally around ten minutes apart; with arrivals being more variable. This means that there is not a great deal conflict in the station throat between departing trains.

Table 1 - Results for Stockholm Green Line case study

Random Seed	Iterations Until Conflict Free	Running Time (s)
0	756	10
1	739	11
2	906	12
3	837	11
4	816	11
5	758	9
6	850	11
7	820	10
8	715	9
9	720	9

8 Conclusions and Future Work

Two simple algorithms for creating rolling stock diagrams and platform allocations have been presented. Both of these produce results very quickly and produce results that are comparable with those created by manual train planning experts. The diagrams are also presented in a manner, which allows the experienced train planner to easily identify and correct any errors or take advantage of other opportunities to improve the generated diagrams. The speed of diagram and platform allocation creation allows more time in the train planning process for various options to be explored and additional robustness checking to be done. The techniques presented are being implemented in the Tracsis[9] product TRACS-RS.

There are various areas that need address to make these tools more widely useful. It would be of benefit to be able, in some circumstances, to vary from the stock type that was used to time the schedules and also to connect different, but compatible, units together. One of the most useful extensions would be some automation of distributing a fixed fleet of rolling stock across a timetable based on the expected demand profile for the timetable. For the platforming, giving some more flexibility by allowing variation in incoming line to a terminal location where the option exists may allow better platform allocations to be created. Also, support for permissively worked platforms, where multiple trains can stop on the same platform (i.e. multiple concurrent associations) would be necessary for more complex scenarios.

9 http://www.tracsis.com/

9 References

1. *Discrete optimisation in public rail transport.* **Bussieck, M, Kreuzer, P and Zimmermann, U.** 1997, Mathematical Programming, Vol. 79, pp. 415-444.

2. *A survey of optimization models for train routing and scheduling.* **Cordeau, J F, Toth, P and Vigo, D.** 1998, Transportation Science, Vol. 32, pp. 380-404.

3. *A model and strategy for train pathing with choice of lines, plat- forms and routes.* **Carey, M.** 1994, Transportation Research Part B, Vol. 28, pp. 333-353.

4. *Testing schedule performance and reliability for train stations.* **Carey, M and Carville, S.** 2000, Journal of the Operational Research Society, Vol. 511, pp. 666-682.

5. *Scheduling and platforming trains at busy complex sta- tions.* **Carey, M and Carville, S.** 2003, Transportation Research Part A, Vol. 37, pp. 195-224.

6. *Scheduling trains on a network of busy complex stations.* **Carey, M and Crawford, I.** 2007, Transportation Research Part B, Vol. 41, pp. 159-178.

7. *A model, algorithms and strategy for train pathing.* **Carey, M and Lockwood, D.** 1995, Journal of the Operational Research Society, Vol. 46, pp. 988-1005.

8. *Train Planning - an exploration of processes and systems.* **Watson, R. 2008,** Loughborough : Loughborough University, PhD Thesis.

9. *Train Timetable Generation Using Genetic Algorithms.* **Hinde, C.J., Withall, M.S., Phillips, I.W., Jackson, T.W., Brown, S. and Watson, R.,** Valencia : SciTePress, 2010. Proceedings of the International Conference on Evolutionary Computation (ICEC 2010). pp. 170-175.

10. *Allocating Railway Platforms Using A Genetic Algorithm.* **Clarke, M., Hinde, C.J., Withall, M.S., Jackson, T.W., Phillips, I.W., Brown, S. and Watson, R.** Cambridge : Springer-Verlag, 2009. Proceedings of AI-2009, the Twenty-Ninth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence. pp. 421-437.