Proceedings of the Tenth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, Atlanta, June, 1997.

TRAIN TIMETABLE AND ROUTE GENERATION USING A CONSTRAINT-BASED APPROACH

Hon Wai Chun
Department of Electronic Engineering, City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong
Email: eehwchun@cityu.edu.hk

ABSTRACT

This paper describes research in modelling train timetable and route generation as a constraint-satisfaction problem (CSP). The key objective of this research is to design a constraint-based scheduling algorithm that can be used to generate a train timetable given headway requirements at different times of the day. The key constraint is to avoid track circuit or route contentions while maximising train utilisation. The objective of the scheduling algorithm is to determine how service levels can be increased without jeopardising passenger safety. This research investigated traffic at a train terminus where two types of trains are competing for the use of the same tracks; trains that are reversing and trains that are being dispatched from the depot. The contention problem is particularly serious during the rush hour train build-up. The current timetable and train routing are generated using two separate rule-based systems. However, due to the complexity of constraints involved, the current systems cannot generate a plan that can meet the desired service levels.

INTRODUCTION

This paper describes a constraint-based model and algorithm for a train timetable and route generation program. This program determines how trains can be efficiently dispatched from a depot to meet rush hour service requirements while satisfying resource constraints. Our research focused mainly on generating a timetable and routing for the building up of train services. However, it can also be applied to the reverse case of breaking down the service after rush hour ends. The research was performed using data from one of the world's busiest subway systems. The constraint-based algorithm described in this paper was tested on one of the busiest lines within this subway system.

The subway authority currently uses a rule-based expert system to generate the train timetable based on headway requirements, i.e., the time between the arrival of two consecutive trains. The routing is then generated using a separate semi-automatic system which is based on heuristics. This routing program determines how each train should travel to get to its destination. Subway trains normally travel in a cyclic manner from one final terminus to other and back. The key problem is what happens when the train reaches the final terminus and needs to reverse while at the same time trains are being dispatched from the depot to the terminus. There is a tremendous amount of contention for the same set of tracks during the rush hours. The timetable and route generation is currently performed by separate systems. The timetable generation system must therefore ensure there is enough slack or buffer for the routing system to be able to generate the routes. This extra time buffer causes inefficiency. The route generation program, on the other hand, is confined to work within the timetable that is given. The subway authority cannot see how service levels can be increased without increasing the efficiency of the scheduling algorithm. This paper proposes a constraint-based algorithm that combines timetable and route generation into one process. Headway constraints for timetable generation are considered at the same time as resource constraints for route generation.

Our research focuses on designing and developing a constraint-based resource allocation algorithm [DUNC94, PUGE94b] that can assist the human scheduler by generating different "what-if" scenarios or timetables based on different sets of criteria, such as headway requirements and train speeds. The resources to be allocated in this study are the track circuits and the routes.

Our constraint-based scheduling algorithm performs three main scheduling tasks - (1) generate a train timetable, (2) generate the route sequence each train must travel, and (3) determine the travel time within turnaround and headshunt interlocking area. The most important concern for the subway authority is the ability to increase service levels during rush hours. The scheduling algorithm must be able to generate a timetable and route sequence that can maximise the utilisation of the track circuit resources in order to meet the service level demands.

PHYSICAL CONSTRAINTS

Figure 1 is a simplified diagram of the track circuits and signals within the turnaround and headshunt interlocking area that are used within the study. There are two platforms at the final terminus -- an "Arrival Platform" for trains arriving from the other terminus and the "Departure Platform" for trains travelling to the other terminus. The study only involved this terminus since there is no depot at the other terminus and hence no track-circuit resource contention.

The track circuits within the turnaround and headshunt interlocking area allow trains to reverse from the Arrival Platform to the Departure Platform. Subway trains can travel in both directions and have a driver compartment at each ends of the train. As part of the reversing process, the train driver must walk from one end of the train to the other. In some cases, there may be an additional driver at the other end to reduce the amount of time needed to reverse a train. While trains are using the tracks to reverse, other trains may be dispatched from the depot through "Depot Track 1" and "Depot Track 2." Furthermore, all tracks within the turnaround and headshunt interlocking area can be used for train travel in either directions.

The tracks in the turnaround and headshunt interlocking area are divided into routes. Routes are further divided into track circuits. Figure 1 shows the names of the track circuits. A route is defined to be a set of track circuits between two signals. Since trains can travel in either direction on the same tracks, there are signals for both direction of travel. The relevant signal is on the right hand side of the direction of train travel.

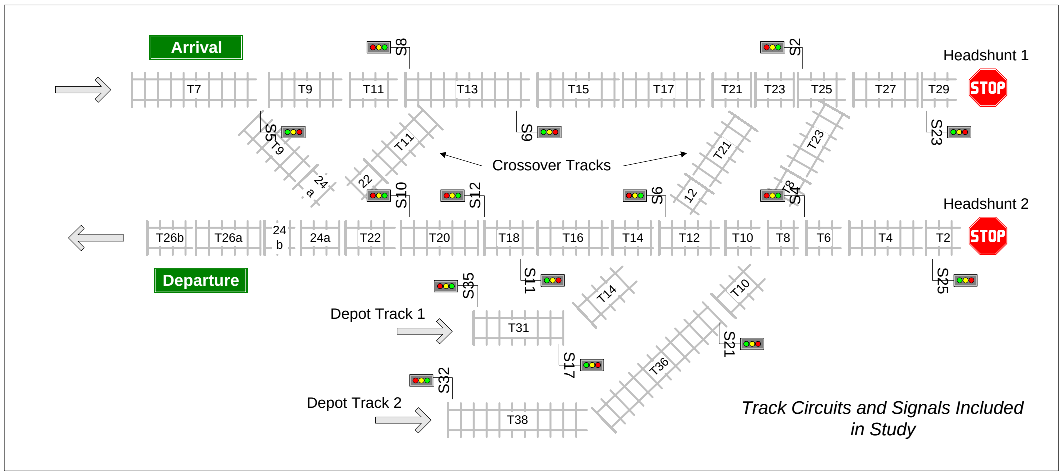

Fig. 1 Track circuits and signals within the turnaround and headshunt interlocking area.

Track Circuits. For the purpose of our research, a "track circuit" is the smallest piece of railway track that can be uniquely identified. However, the length of each track circuit may be different. The *state* of a track circuit may either be "down" indicating that a train is currently over this piece of track circuit or "up" indicating that the track circuit is free. The track circuits in Figure 1 are drawn roughly to proportions. A total of 33 track circuits are included into the study. The same track numbers are used to identify tracks branching out from track circuits, such as the crossover tracks, and their parent branches.

Signals. Figure 1 illustrates all the signals that were used in our research. There are a total of 16 signals that will define the starting point for 42 possible routes that a train may take. However, several routes end at the Departure Platform, which does not have a signal after track circuit T26b. For uniformity, the Departure Platform will be considered as a "pseudo-signal."

Routes. For this study, a "route" is defined as a sequence of track circuits that starts and ends at a signal (including the pseudo-signal). Before a train enters a route, it must first "call" and "set" the route. Setting a route reserves a section of track so that no other train will use the same track resource. A set of Boolean equations or constraints defines when a route can be set. The equations ensure that the route is safe to enter.

Our scheduling system differentiates routes that span two immediate signals as "basic routes." Routes that are composed of more than one "basic routes" are called "composite routes." The scheduling algorithm considers a total of 22 basic routes and 20 composite routes. To travel from a start signal to the target destination signal may require the train to traverse a sequence of basic routes. This sequence is referred to as a "route sequence."

Train Lengths. To determine track circuit occupancy, the length of the train is needed. The length of the trains used in this research is 177 metres from axle to axle, while the end to end length is 182.5 metres.

TIMING CONSTRAINTS

Several different types of timing data were used by our scheduling algorithm. This includes the run times of the train, the signal timing, the time to change the crew, and the transmission time.

Run Times. Only the nominal run times are used in this study. The scheduling algorithm used the run times to determine when routes will be freed up for other trains to use. The run times are defined as the travel time needed between any two signals without stopping.

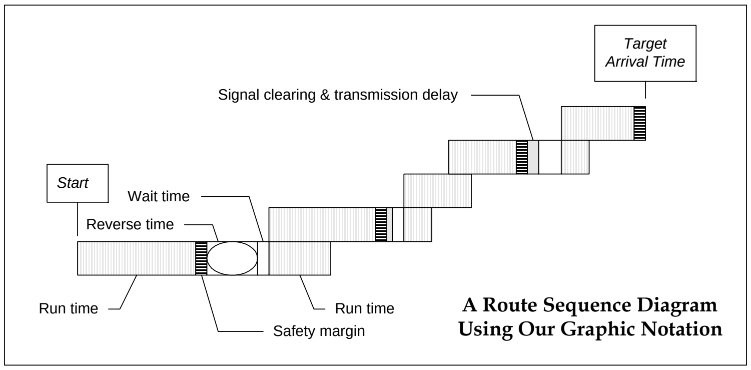
Target Arrival Time
Signal clearing & transmission delay
Start
Wait time
Reverse time
Run time
Run time
Safety margin
A Route Sequence Diagram Using Our Graphic Notation

Fig. 2 The representation of a route sequence and its timing.

Signal Timing. It takes about 7 seconds to set a route if point movement is required. It takes about 3 seconds to set a route if point movement is not required.

Double-ended Timing. Double-ended crew is the case where there is a driver at both ends of a train. This technique is used to reduce the time needed to reverse a train. Normally, to reverse a train, the driver should stop the train and walk from the one end to the other to restart the train. The time required for the whole process is around 4 minutes. With double-ended crew, the additional driver will restart the train after the current driver stops it. Reversing, in this case, only takes about 15 seconds to finish but requires a larger crew size to operate.

Transmission Delay. A 2-second transmission delay from issuing the route setting command (by computer or operator) to actuation of the route setting activity is required.

Figure 2 is a Gantt chart that shows the type of timing information that is represented and used by our scheduling algorithm. Each row, along the vertical axis, represents one particular route and the different types of timing involved when a train travels in that route. The complete sequence of rows represents the sequence of routes a train takes from the start signal, e.g. the depot, to the end signal, e.g. the Departure Platform.

THE CONSTRAINT-BASED MODEL AND ALGORITHM

To solve this problem, we represented the timetable and route generation problem as an object-oriented [LEPA93] constraint-satisfaction problem (CSP) [KUMA92]. Although CSP or constraint-programming has a relatively long history [STEE80], with constraint language extensions found in Prolog [COLM90, VANH89], and Lisp [SISK93], it is only recently that constraint-programming became more popular with the availability of the ILOG's C++ class libraries [PUGE94a]. This library provided a very efficient and clean implementation of constraint-based programming features in a conventional language. Our scheduling algorithm was implemented using the ILOG C++ class libraries.

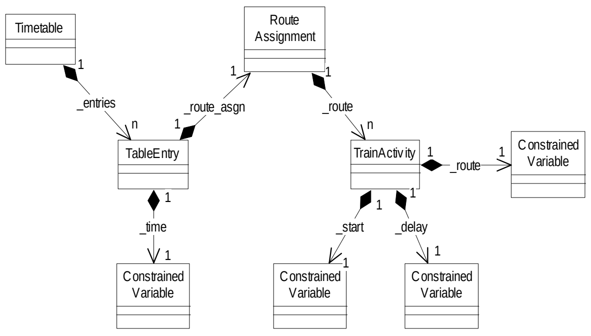

Fig. 3 The UML class diagram of the constrained variables.

In general, any scheduling and resource allocation problems can be formulated as a constraint-satisfaction problem (CSP) which involves the assignment of values to variables subjected to a set of constraints. CSP can be defined as consisting of a finite set of *n* variables $v_1, v_2, ..., v_n$, a set of domains $d_1, d_2, ..., d_n$, and a set of constraint relations $c_1, c_2, ..., c_m$. Each d_i defines a finite set of values (or solutions) that variable v_i may be assigned. A constraint c_j specifies the consistent or inconsistent choices among variables and is defined as a subset of the Cartesian product: $c_j \subseteq d_1 \times d_2 \times ... \times d_n$. The goal of a CSP algorithm is to find one tuple from $d_1 \times d_2 \times ... \times d_n$ such that *n* assignments of values to variables satisfy all constraints simultaneously.

When the train timetable and route generation problem is formulated as a CSP, the problem is represented as a set of constrained variables. The first type of constrained variable represents entries in a train timetable. There are two timetables in our problem – a timetable for the Arrival Platform (T1) and a timetable for the Departure Platform (T2). Each timetable is a list of constrained variables. The domain of the variables will depend on the desired headway values requested by the user.

Each timetable entry also contains a "route assignment." Each "route assignment" represents a sequence of "train activity." Each train activity represents the selection of a particular route, at a particular start time, with a particular delay. The route, start time, and delay are all represented as constrained variables. The domain of the route variable is all routes that can start from the current location of a train. The domain of the start time is related to the previous route's end time. The domain of the delay represents the amount of time a train might need to wait at a signal. Figure 3 is a simplified Unified Modelling Language (UML) class diagram that shows the essential classes in our design.

This constraint-based formulation was designed from the requirement that the input to the scheduling algorithm will be a table of desired headway values at the Departure Platform. Therefore the schedule algorithm will schedule a time for the arrival of a train at the Departure Platform. However, we will need to work backward to figure when the train departs from the depot and how long the train waits at each intermediate signal. On the other hand, this train might be a reversing train from the Arrival Platform. In this case, the scheduling algorithm works forward from when the train leaves the Arrival Platform until it reaches the Departure Platform. This combination of searching backwards and forwards at the same time complicates the algorithm, but is a necessity given the subway authority's input requirement. Figure 4 illustrates this combined search.

Although the search is a combined backward and forward search, the scheduling of the timetable entries is a forward process; timetable entries are instantiated from the earliest entries first. In other words, the timetable is generated from the first train to arrive at the T2 Departure Platform to the last train of the desired scheduling period.

During the constraint-based search, the algorithm predicts when a train leaving the Departure Platform will return back to the Arrival Platform from the other terminus using nominal turnaround times. In addition, the algorithm merges the time a train leaves the Arrival Platform with the timetable entry for the arrival of that same train at the Departure Platform. This merging operation is the major source of backtracking for the scheduling algorithm. The time a train arrives at the Departure Platform from the Arrival Platform might not match exactly the headway required for the Departure Platform and the train must wait inside the turnaround area for the next departure time slot.

For each time value that the algorithm assigns to the timetable, the algorithm also selects a route sequence that will lead the train to arrive/depart at the desired time. The route sequence with the shortest total runtime that does not interfere with any other previously made route assignments will be selected first.

For each potential route sequence that is selected, the algorithm also determines how much time the train should wait at each signal for signal clearance. The minimum amount of waiting time will be used. This is equivalent to "stretching" the route sequence as shown in Figure 5 (the arrows highlight the "stretching" action and not the temporal evolution).

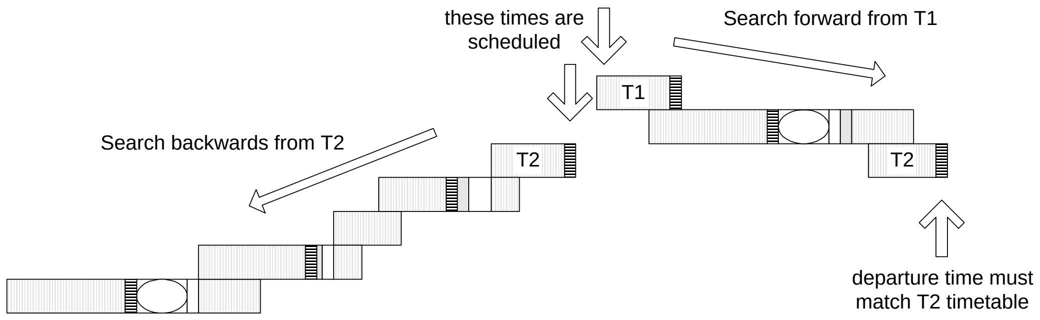
these times are scheduled
Search forward from T1
T1
Search backwards from T2
T2
T2
departure time must match T2 timetable

Fig. 4 The combined backward and forward search.

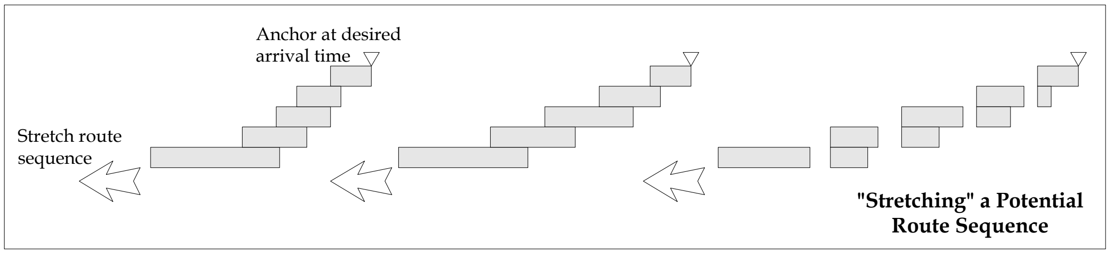
Anchor at desired arrival time
Stretch route sequence
"Stretching" a Potential Route Sequence

Fig. 5 "Stretching" a route sequence by adjusting the waiting time.

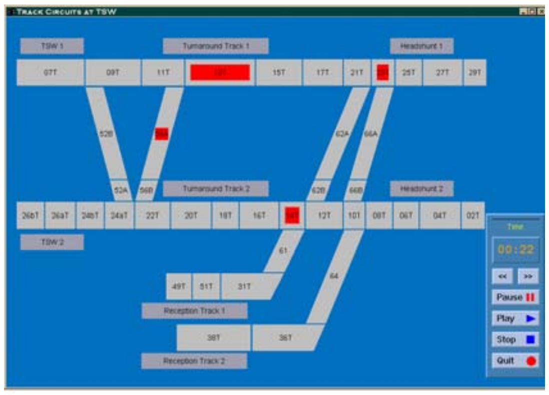

Fig. 6 The graphic simulator used to display the scheduling results.

If the proposed route sequence generated by the scheduling algorithm does not conflict with any other previously assigned routes, then that route will be assigned to the train. On the other hand, if no feasible routes can be found, the algorithm will try to adjust the waiting times at the signals. If no solution can be found, the algorithm then adjusts the arrival/departure times at the station. If still no solution can be found, then the algorithm tries to assign a different route sequence to the previous timetable entry. This "undoing" of previously made assignments is performed automatically by the backtracking mechanism provided by constraint programming. This backtracking will continue until a solution is found or when all possible solutions have been tried. Once the timetables and route sequences have been generated, graphic simulation software is used to display and visualise the resulting schedule (see Figure 6).

ADVANTAGES OF CONSTRAINT PROPAGATION

To illustrate the problem complexity, for the 3 hours of morning train dispatch there are around 70 choice points for just the timetable generation - each with potentially over 100 possible values. For route selection, there are a total of 130 route sequence combinations within the interlocking and headshunt area, each with an average of 4 routes, each route with possibly 60 timing variations. The total complexity of just the morning dispatching is far too much for any rule-based approach.

The constraint-based approach, proposed in this paper, was able to generate timetables and route sequences within a reasonable time because of constraint propagation. Constraint propagation eliminated invalid choices before they can be selected for search. In the case of timetable generation, routes that conflict with assigned routes will be eliminated and timetable entries that violated headway requirements will not be selected. The constraint-based approach makes use of arc consistency [MACK77] and constraint propagation [WALT72] to reduce the domain size of each constrained variable before search. Smaller domain size means smaller and more focused search space.

RESULTS FROM TEST CASES

The constraint-based scheduling algorithm was tested on many test cases. The algorithm was designed to minimise changes in the desired headway value as much as possible. However, minimising headway changes in early morning forces the headway at the final target to fluctuate slightly.

The following table lists test cases that achieved the desired 105-second headway values, i.e., roughly 34 trains per hour. Currently, the trains operate at roughly 112 or 113-second headway values. Also listed is the number of trains that were dispatched from 6am to 8:40am, the number of trains that had a headway value of exactly 105 or 106 seconds, and the average deviation from 105 seconds within the 7:40am to 8:40am peak.

Table 1. Summary of Testing Results

Test Case No	*Total trains since 6am*	*No. of 105 or 106 trains*	*Avg. Deviation from 105 sec.*
2	66	29	1.4
5	72	22	4.7
10	73	21	4.8
13	73	22	4.7
14	73	23	4.7
15	73	23	4.7

As an example, Test Case 15 was produced with the following input headway table.

Table 2. The Headway Table Used for Test Case 15

Station	*Start*	*End*	*Headway*	*No Trains*
TSW2	6:00	6:30	240	8
TSW2	6:30	7:00	180	10
TSW2	7:00	7:25	120	13
TSW2	7:25	7:40	112	8
TSW2	7:40	8:40	105	34

Out of these test cases, the best result was obtained in Test Case 2 where trains were dispatched for the 105-second headway with an average deviation of only 1.4 seconds. The current rule-based systems can at most dispatch up to 112-second headway values.

SYSTEM IMPLEMENTATION

The object-oriented constraint-based [LEPA93] allocation system was implemented in C++ using the ILOG Solver class library [PUGE94a] and the RTL Scheduling Framework developed by Resource Technologies Limited [CHUN96a, CHUN96b]. The graphic user interface that simulates the generated schedule was developed using C++ graphic components provided by ILOG Views. The system was developed using platform independent coding and can execute within Windows 95/NT or Unix environment.

CONCLUSIONS

This paper documents our research in modelling train timetable and route generation as a constraint-satisfaction problem. The constraint-based scheduling algorithm was tested using data from one of the busiest subway systems in the world. The results showed that the scheduling algorithm was able to generate timetables and routes that had a higher service level than that was previously possible with a rule-based approach.

ACKNOWLEDGEMENTS

The author would like to thank the Hong Kong Mass Transit Railway Corporation for the cooperation received and for making actual data available for this research. Part of this research was performed in cooperation with Resource Technologies Limited (http://www.rtl.com.hk/~rtl) in Hong Kong and with assistance from ILOG (http://www.ilog.com) in Singapore.

REFERENCES

[CHUN96a] H.W. Chun, K.H. Pang, and N. Lam, "Container Vessel Berth Allocation with ILOG SOLVER," The Second International ILOG SOLVER User Conference, Paris, July, 1996.

[CHUN96b] H.W. Chun, M.P. Ng, and N. Lam, "Rostering of Equipment Operators in a Container Yard," The Second International ILOG SOLVER User Conference, Paris, July, 1996.

[COLM90] A. Colmerauer, *An Introduction to Prolog III*, Communications of the ACM, 33(7), pp.69-90, 1990.

[DUNC94] T. Duncan, "Intelligent Vehicle Scheduling: Experiences with a Constraint-based Approach," ILOG Technical Report 94-04.

[KUMA92] V. Kumar, "Algorithms for Constraint Satisfaction Problems: A Survey," In *AI Magazine*, 13(1), pp.32-44, 1992.

[LEPA93] C. Le Pape, "Using Object-Oriented Constraint Programming Tools to Implement Flexible "Easy-to-use" Scheduling Systems," In *Proceedings of the NSF Workshop on Intelligent, Dynamic Scheduling for Manufacturing*, Cocoa Beach, Florida, 1993.

[MACK77] A.K. Mackworth, "Consistency in Networks of Relations," In *Artificial Intelligence*, 8, pp.99-118, 1977.

[PUGE94a] J.-F. Puget, "A C++ Implementation of CLP," In *ILOG Solver Collected Papers*, ILOG SA, France, 1994.

[PUGE94b] J.-F. Puget, "Object-Oriented Constraint Programming for Transportation Problems," In *ILOG Solver Collected Papers*, ILOG SA, France, 1994.

[SISK93] J.M. Siskind and D.A. McAllester, "Nondeterministic Lisp as a Substrate for Constraint Logic Programming," In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, pp.133-138, July, 1993.

[STEE80] G.L. Steele Jr., *The Definition and Implementation of a Computer Programming Language Based on Constraints,* Ph.D. Thesis, MIT, 1980.

[VANH89] P. Van Hentenryck, *Constraint Satisfaction in Logic Programming,* MIT Press, 1989.

[WALT72] D.L. Waltz, “Understanding Line Drawings of Scenes with Shadows,” In *The Psychology of Computer Vision,* McGraw-Hill, pp.19-91, 1975.