Newly developed simple railway timetable evaluation program Sujic with the new model to deal with re-scheduling

R. Takagi
Department of Electrical Engineering, Kogakuin University, Japan

Abstract

Many attempts have been made in recent years to automate fully the task of generating railway timetables, mainly by using one of the numerical optimisation techniques. Virtually all of these attempts depend on some kind of technique to evaluate the timetable. Upon evaluation of a railway timetable, it is important to take into account the passengers' points of view, including the disbenefit (inconvenience) for any single passenger during his/her entire journey on the railway. As one of such evaluators, Sujic was first developed in 1988, and re-written in 1996, at the research group at the University of Tokyo, and has since been used by many researchers in a number of universities in Tokyo. However, this program is slow and memory-eating because of its poor implementation of path search algorithms. Also, the program has no capability to deal with re-scheduling.

The standard Dijkstra's algorithm could not be used by the old Sujic program because of the path search criteria. In the new Sujic, the author applied the slightly modified Dijkstra's algorithm and drastically improved the efficiency of the search without changing the criteria, the efficiency of the search has drastically improved.

The evaluation of train re-scheduling is slightly different from that of scheduling because the arrival and departure times of trains that passengers expect (which is the scheduled times) are different from the actual arrival and departure times under re-scheduling. Although evaluation can be performed by simply providing old Sujic with the actual timetables, it yields a result in which passengers act as if they "expect" that an accident will happen in the future. In the new version, this problem is addressed by adding the functionality to use multiple schedules in one evaluation.

In this paper, the author's efforts to enable the above mentioned points are discussed in detail..

Keywords: optimization, simulation, passenger activity, train timetabling, re-scheduling, electric railway.

1 Introduction

Sujic [1] is a simple train schedule evaluation program designed for the evaluation of train schedules for passenger railways. It incorporates simple implementations of the passenger activity simulation models, and is capable of obtaining the evaluation based on the estimates of train congestion and the resulting disbenefit (inconvenience) of all passengers. The first version of this program was developed in 1988; the second version, which is still in use, was developed in 1996 (hereafter called 1996 Sujic).

However, the part of the 1996 Sujic that performs the path search (for distributing passengers to trains to calculate congestion) is far from well implemented, it is difficult to embed this simulation tool into an optimisation procedure. Some of the assumptions in 1996 Sujic have prevented the re-implementation of path search using the well-known algorithms for the shortest path problems such as the Dijkstra's algorithm [1]. In addition, 1996 Sujic is incapable of evaluating train schedules with the coupling/uncoupling of trains in service. Furthermore, by principle, it is impossible to apply this program to evaluate re-scheduling, since the schedule to be evaluated must be given beforehand and the passengers act as if they knew when and where the disruption takes place before it actually happens.

In this paper, the author discusses how Sujic can be re-designed by using the modified version of the Dijkstra's algorithm without discarding any of the assumptions of the 1996 program.

2 Basic assumptions of 1996 Sujic

2.1 Assumptions of passenger activities

1996 Sujic assumes that there are two types of passengers, each type having one activity model. These are:

(1) The time a passenger turns up at the originating station is given at random. After turning up at the originating station, he/she chooses train services so that he/she can reach the destination station at the earliest possible time.

(2) The time a passenger wants to reach the destination station is given at random. A passenger investigates the train schedule before the start of the journey, and decides the services to take so that his/her departure time from the originating station becomes the latest. He/she turns up at the originating station as he/she has planned, and complete the journey as planned.

The activity models (1) and (2) are called “F1” and “F2”, respectively.

In these models, passengers do not take into account the congestion of the trains. The resulting value of congestion rate (load factor) of a train can be any positive real number. It can be unrealistically high, in which case the evaluated train schedule itself is to be considered badly designed.

2.2 Detailed path selection criteria

Hereunder, only the activity model F1 is considered, because F2 is basically the reverse of F1; If the path search of F1 is done from the originating station to the destination station, the path search of F2 is done from the destination station to the original station, and basically the same algorithm can be applied for both models.

As explained in Section 2.1, a passenger acting under activity model F1 chooses services so that the arrival time at his/her destination station is the earliest. However, there may be cases in which a passenger can find two or more paths to destination station with the same arrival time.

To avoid this situation as much as possible, the F1 activity model in 1996 Sujic defines additional criteria. If there are two or more paths for a passenger travelling from the origin station to destination station, the selection of the path is done as follows:-

(1) the path with the earliest arrival time at destination;
(2) the path with the fewest interchanges between trains;
(3) the path with the shortest walking time at interchange stations; and
(4) the path with the shortest wait time before boarding trains.

Here, the walking time at an interchange station is the time during which a changing passenger is required to walk. Also, the wait time is the time during which a passenger is requested to wait for the departure of the train he/she intends to take. For example, if a passenger arrives at platform A of a station on a train and intends to change to another train departing 5 minutes later from platform B of the same station, and the (shortest) walking time between platforms A and B is 2 minutes, then the walking time and the wait time of this passenger for this interchange are 2 minutes and 3 minutes, respectively.

A path search is done for a group of passengers who turn up at the origin station between any two adjacent events at the station in the train timetable and travel to the same destination station (an event in the train timetable is either a departure or an arrival of a train therein). If, by using criteria (1) through (4), a unique path is selected for this group, all passengers in the group are assumed to travel along the path. If the arrival time, the number of interchanges, the walking time and the wait time are the same for N paths, namely path 1, path 2, ..., and path N, the group is divided into N sub-groups, namely sub-group 1, sub-group 2, .., and sub-group N, all with equal number of passengers in it, and it is assumed that passengers in sub-group i ($1 \leq i \leq N$) is to travel along path i.

2.3 Other basic assumptions

The train schedule to be given is a cyclic one, i.e. the same schedule is repeated when one cycle is done.

The passenger demand data is given in the form of an OD (origin-destination) table. Whatever the train schedule, it is assumed that the OD would not change.

2.4 The evaluations that the program yields

Apart from the data on congestion of trains, the program calculates and outputs the following:

(1) Normalised train hours and car hours. These can be calculated instantly using the schedule given as data. If an *N*-car train spends *x* hours in the system, then it accounts for *x* train-hours and *Nx* car-hours. Aggregating the train-hours and car-hours for all trainsets and dividing these values by the length of the cycle of the schedule yields the normalised train hours and car hours. These values represent how many trains or cars are necessary to run the schedule.

(2) Effective congestion rate. This is the average congestion rate that the passenger experiences. Assume that the capacity of train i is given as C_i. If the number of passengers actually on train i for section j, which is as long as t_{ij} [s], is given as P_{ij}, then the effective congestion rate C for this system can be calculated using the next equation:

$$C = \frac{\sum_i \sum_j \frac{P_{ij}{}^2 t_{ij}}{C_i}}{\sum_i \sum_j P_{ij} t_{ij}} \tag{1}$$

(3) Disbenefit converted to costs. Congestion, the number of transfers, the average walking and wait times are all converted to the costs in currency using the coefficients given as input data.

3 Application of the Dijkstra's algorithm

It is intended in the development that the same criteria used for the path search in 1996 Sujic be used in the new Sujic as well. For this to be realised, the following two issues are discussed in this section:- (1) the selection of the weight value set, and (2) the modification of the algorithm so that multiple paths can be obtained when the resulting lengths of the path are identical.

3.1 Definition of the weight value set

The well-known Dijkstra's algorithm [1] can be used to find the shortest paths from one of the vertices to every vertex of a directed graph. Every arc in the graph to be solved by the algorithm is assigned a weight value, w, which is the element of the weight value set W. It is required that addition (+) and comparison (<, =, >) operations are defined over W, and that for any $a, b \in W$, $a + b \leq a$. Also, it is required that a zero element exists within W. If $a, b \in W$ and b is the zero element, then $a + b = a$ for any a.

Generally, the explanation in the textbooks on the algorithm state that the weight is of non-negative real number. However, any set with the above mentioned property can be used as the weight.

In the new Sujic, the author devised a data structure, which is a 4-tuple (t_T, c, t_F, t_W), where t_T, c, t_F and t_W are the travel time, the number of embarking into or disembarking from trains, the walking time and the wait time, respectively, as explained in Section 2.2. All four variables in the tuple are non-negative.

Let $W = \{(t_T, c, t_F, t_W) \mid t_T \geq 0, c \geq 0, t_F \geq 0, t_W \geq 0\}$. Define the addition operation for $a = (a_1, a_2, a_3, a_4) \in W$ and $b = (b_1, b_2, b_3, b_4) \in W$ as:-

$$a + b \equiv (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4) \tag{2}$$

Define that a and b are identical ($a = b$) if and only if $a_1 = b_1$ and $a_2 = b_2$ and $a_3 = b_3$ and $a_4 = b_4$. Also, define that $a < b$ if and only if one of the following four conditions are met:-

(1) ($a_1 < b_1$),

(2) ($a_1 = b_1$ and $a_2 < b_2$),

(3) ($a_1 = b_1$ and $a_2 = b_2$ and $a_3 < b_3$), or

(4) ($a_1 = b_1$ and $a_2 = b_2$ and $a_3 = b_3$ and $a_4 < b_4$).

The set W as defined above satisfies the precondition of the Dijkstra's algorithm, and therefore can be used as the weight value set.

3.2 Modification of the algorithm

The original Dijkstra's algorithm yields only one shortest path per pair of origin and destination vertices, even if there is another path with the same length between them. The application of the so-called k-th path algorithms may have solved the issue; however, in this case only the paths with the same length matter, and therefore a slightly modified version of the Dijkstra's algorithm was implemented.

The original Dijkstra's algorithm works as described hereunder:-

(0) Let E and V respectively be the sets of all arcs and all vertices in the graph to be solved. Let $v_O(e)$, $v_D(e)$ and $w(e)$ respectively be the origin vertex of, the destination vertex of, and the weight assigned to, the arc $e \in E$. Let $l(v)$ and $e_l(v)$ respectively be the length from the starting vertex $v_S \in V$ to vertex $v \in V$ and the arc on which the shortest path from v_S to v resides; both $l(v)$ and $e_l(v)$ may change as the algorithm progresses. Let $E_O(v)$ be the set of arcs that go out from vertex v.

(1) Let V_S be the set of vertices to search. Populate V_S with all the elements of V. At this stage, the number of elements in V_S and V must be the same. For all $v \in V_S$, let $l(v) = \infty$. Identify the starting vertex $v_S \in V$ and let $l(v_S) = w_0$, where w_0 is the zero element in the weight value set.

(2) If $V_S = \phi$ then terminate. Otherwise, go to step (3).

(3) Find within V_S the vertex $v_x \in V_S$ that has the smallest $l(v_x)$, and remove v_x from V_S. If $l(v_x) = \infty$ then terminate.

(4) For all $e_o \in E_O(v_x)$, if $l(v_x) + w(e_o) < l(v_D(e_o))$ then let $e_I(v_D(e_o)) = e_o$ and $l(v_D(e_o)) = l(v_x) + w(e_o)$.

(5) Go back to (2).

In the new Sujic, the following modifications is made to the algorithm as shown above:-

(a) In step (0), instead of defining $e_I(v)$, let $E_I(v)$ be the set of arcs on which the shortest paths from v_S to v reside.

(b) Replace step (4) by the following.

- For all $e_o \in E_O(v_x)$, if $l(v_x) + w(e_o) < l(v_D(e_o))$ then let $E_I(v_D(e_o)) = \{e_o\}$ and $l(v_D(e_o)) = l(v_x) + w(e_o)$. Also, for all $e_o \in E_O(v_x)$, if $l(v_x) + w(e_o) = l(v_D(e_o))$ then add e_o to $E_I(v_D(e_o))$ as one of its elements.

In the standard Dijkstra's algorithm, the shortest path from v_S to v ($v_S \neq v$) exists if and only if $l(v) \neq \infty$, in which case it can be found either by adding arc $e_I(v)$ to the end of the shortest path from v_S to $v_O(e_I(v))$ when $v_S \neq v_O(e_I(v))$, or arc $e_I(v)$ itself otherwise. In the modified version, the shortest path from v_S to v ($v_S \neq v$) exists if and only if $l(v) \neq \infty$, in which case one of the paths can be found either by adding arc e_I to the end of one of the shortest paths from v_S to $v_O(e_I)$ when $v_S \neq v_O(e_I)$, or arc e_I itself otherwise, where $e_I \in E_I(v)$. If there are multiple elements in $E_I(v)$, then there are multiple shortest paths that converge at vertex v.

3.3 Creating the network to solve from the train schedule

For example, assume that we are evaluating the train schedule as shown in Fig. 1. This is a schedule with the cycle of ten minutes, i.e. the same pattern repeats every ten minutes. In one cycle, Train 1 departs Station A at time 0, and Train 2 departs Station A at time 2 and overtakes Train 1 at Station B. Train 1 is

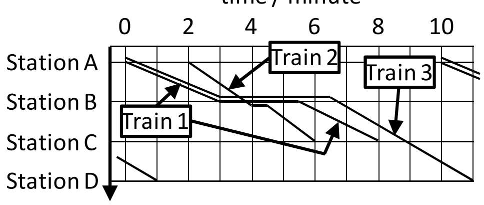

Figure 1: An example train schedule for evaluation.

made up of two trainsets, one towards the front going to Station C and another towards the rear going to Station D. Front and rear trainsets of Train 1 are uncoupled at Station B, and the trainset going to Station D continues its journey as Train 3. At Station B, walking time between Trains 1 and 2 can be ignored.

The graph used to solve Fig. 1 is shown in Fig. 2. Passengers are to emerge at one of the platform arcs of the originating station. Except for the vertices showing the exit of Stations B, C and D, all vertices correspond to either the departure or the arrival of one of the trains. An arrival or a departure of a train is expressed by a vertex on the platform side and one vertex per trainset on the train side. The paths on which passengers travel are to be determined by finding the shortest paths from where the passengers emerge to the exit vertex of the destination stations.

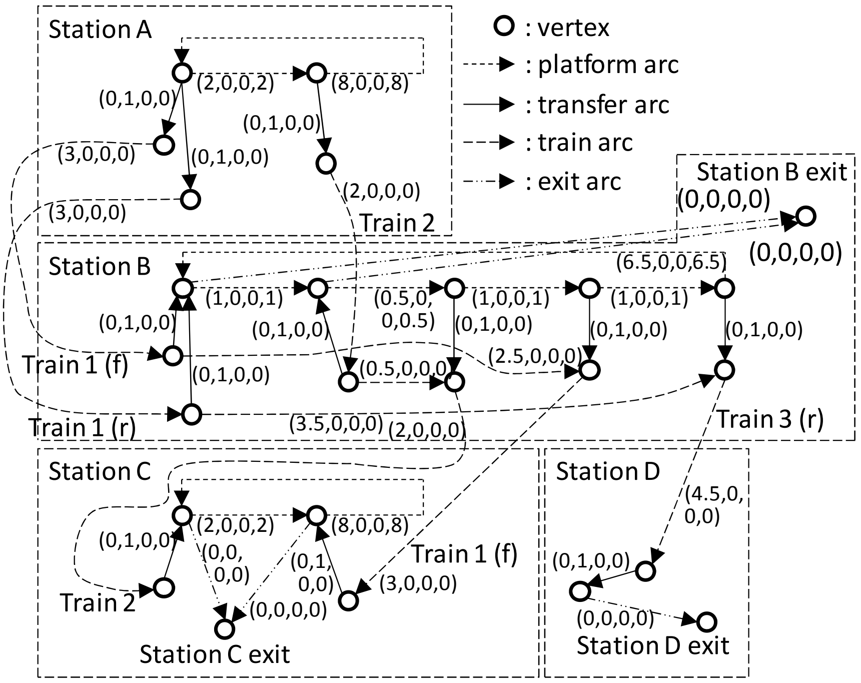

Figure 2: Graph to be used to solve example schedule of Fig. 1.

3.4 Consideration of coupling/uncoupling of trains

Generally, if there are multiple possible paths for passengers, these passengers are equally divided and distributed to individual paths. However, cases in which trains formed with multiple trainsets are involved must be given special consideration.

For example, in Fig. 1, Train 1 is formed by two trainsets. Passengers who are to travel on Train 1 from Station A to Station B may use both trainsets. In this case, it would be realistic to assume that the ratio of the number of passengers using the two trainsets is equal to the ratio of capacity of the two trainsets. However, it is shown in Fig. 2 that each trainset is given a train arc for every inter-station run. In Fig. 2, the train arcs linking Station A and Station B have train names marked near them; Train 1 (f) and Train 1 (r) respectively mean the trainset towards the front and the rear of Train 1. Therefore, using the different trainset of Train 1 between Station A and Station B means the path in the graph is different, and the ratio of the number of passengers is always 1:1 if the equal distribution principle is applied.

Therefore, in the new Sujic, the program looks through all the paths and find where the diversion takes place. If the two paths diverge at a platform node and embark into the different trainsets that form the same train, then the ratio of the number of passengers is determined by the capacity of the trainsets. In all other cases, two paths are given the same number of passengers.

4 Applying the program to re-scheduling problems

Evaluation of the train schedule without disruption and re-scheduling after disruption differs because the re-scheduling plan is arranged after a disruptive event and there are passengers already travelling assuming the original timetable.

By designing the program so that all passengers are expressed as tokens, and that each arc in the graph are assigned a set of tokens that contain those travelling on that particular arc, it is easy to take a snapshot of where passengers are in the system at any time. In the evaluation of re-scheduling, this snapshot can be used as the starting point, avoiding the problem when using 1996 Sujic for re-scheduling evaluation that passengers act as if they know the disruptive event is going to take place before it actually happens.

5 Conclusion

The new Sujic program is expected to be used extensively within our research group and possibly beyond, replacing the original 1996 Sujic. It is expected that the program is used embedded in the optimisation routine, so that the optimal train schedules or re-scheduling plans can be numerically obtained.

References

[1] "Simple Train Evaluation Program Sujic" (in Japanese), http://www.takagi-ryo.ac/railways/sujic/ (accessed June 2012)

[2] Dijkstra, E. W.: "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik 1*, pp. 269–271 (1959).