Time-Tabling of Suburban Train Services

Narayan Rangaraj *
Industrial Engg. and Operations Research, I. I. T. Bombay

Milind Sohoni
Computer Science and Engg., I. I. T. Bombay

1 Introduction

Suburban train services in metro areas are an important mode of urban transport. The planning of such services is a challenging task of resource allocation to meet customer demands in an effective manner. This paper talks about the modelling of the process of timetable generation and some experiences in generating effective timetables. The major elements in this exercise are the modelling of the physical rail network (which basically forms the resource base on which services are run), the representation of activities, the modelling of constraints (especially those which link activities with resource use), and finally, the development of a tool for the actual generation of usable timetables for various purposes. The example chosen to illustrate the approach in detail is from the Bombay suburban train system.

The paper is organised as follows. The next section describes the timetabling problem in detail, and identifies some practical constraints that need to be modelled. The next section (Section 3) proposes a phased approach to timetabling, and also a partitioning of the decisions to be done, in a linked and iterative manner. Specifically, two major parts of a timetable, the schedule of trains and the provisioning of rakes (physical train units) to services, are handled by two procedures; the scheduler and the rake linker, respectively. These are described in Sections 4 and 5. The scheduler allocates times at all station and track segments for the desired services, in a manner that respects all operational constraints, if at all possible. The rake linker allocates services to physical train units and generates linked services that will be handled by the same physical rake. The last two sections (6 and 7) describe the development of a software tool for the scheduler and rake linker and the implementation experience of deploying this tool on a large, real life example.

2 A model for a train timetable

In this section, we attempt to describe a detailed model of a suburban train timetable, which is amenable to representation and manipulation as part of a (partially) automated process of timetable generation. The aim is not to model the rail network and activities in all detail, but to model it sufficiently accurately that all major operational considerations are included. At the same time the

*Both authors acknowledge the partial support of Tata Infotech., Ltd.

model should be such that the "correctness" can be validated by a user and it should be possible to use a computationally tractable tool based on the model for it to be usable.

2.1 Elements of the physical timetable

Practically speaking, for our purposes, a suburban time table consists of the following two documents:

- Working timetable This is essentially a table, with train services as columns and with two rows corresponding to each station, where the arrival and departure of each relevant service at that station are indicated in the table. Figure 1 illustrates a segment of a typical working timetable.
- Rakelink document This is a list of physical train units (rakes) numbered in some order, with the set of services that each rake is supposed to perform. For convenience, the service linkage may also be indicated in the working timetable, in the last row. Figure 2 shows part of a rakelink document.

2.2 Model for the rail network

Railway operations take place on a physical network, and there are strict constraints on the usage of this network. We identify four major objects, which enable us to represent the topology of a rail network. Of these, the first two (hold and line resources) represent stations and tracks between stations, respectively. These are the major parts of the rail network where services originate and terminate and run. The part of the network between stations could also have small track segments that are common to two or more line resources, and these are represented by forks and joins.

HR Hold resource (Station) A station in a rail network is modelled as a hold resource with the following attributes:

1. Name of resource
2. Number of platforms (accessible from different line resources)

LR Line resource (Track segment) A track segment (between stations) is modelled as a line resource, with the following attributes:

1. Name of resource
2. Hold resource immediately preceding the resource
3. Hold resource immediately succeeding the resource
4. Time of usage by an activity
5. Forks from this resource (name of fork)

Fk Fork (diverging point) A fork describes a diverging point in the network, where there are two or more successors to a track segment. It is represented as:

1. Line resources involved in the fork

2. Successor (hold) resource for every line resource involved

Jn Join (converging point) A join represents a converging point in the network, where there are two predecessors to a track segment. It is represented as:

1. Line resources involved in the join
2. Predecessor (hold) resource for every line resource involved

The primary constructs are the hold and line resources (which are described in detail in Sections 4.1.1 and 4.1.2). The fork and join resources are listed for convenience. They can be consistently generated from the attributes of the relevant line and hold resources. We model only those forks and joins which are used by services regularly.
Example: *In the Bombay suburban network for Western Railway, there are 28 stations or hold resources. There are a large number of line resources, forks and joins. Illustrations of a hold resource and a line resource from this network is presented in tables 1 and 2.*

2.3 Model for activities (Patterns)

For operational purposes, a train service on the rail network is fully described by its orignating station, destination station, halts at intermediate stations, times at all stations (arrival/departure or both, as relevant), platform assignation at all stations, and track allocation between all stations. In terms of the network representation above, a service is represented by a time of departure from a hold resource, and a sequence of line/hold resources, with times of usage.

We simplify the activity model by assuming that services will run according to certain *patterns*. A pattern is a predefined seqeunce of n hold/line resources, which is like a template on which services can run, at some specified time. A pattern is described as follows:

1. Hold resource (Origin)
2. Line resource
   ...

n-1 Line resource

n Destination (Destination)

A *service* is a pattern with some specified time of departure from the originating station. More generally, one could allow an interval of operation (say specified by a time interval at the originating point).
Example: *A typical pattern on the Bombay suburban system is as follows.*
Churchgate-Borivali fast (CCG-BVI-fast)

- *HR: CCG (platform 1,2,3,4) - Origin*
- *LR: CCG-MEL (track 3)*

- *HR: MEL (platform 3)*

  ...

  ...

- *LR: KILE-BVI (track 1)*

- *HR: BVI (platform 2,3) - Destination*

A typical service is CCG-BVI-fast, departing CCG at 17.45.

The structure of a pattern is described more precisely in Section 4.1.3 below and an example of the data structure is given in Table 3..

2.4 The (partial) time table

The Valid Partial Time Table (`vptt`) is our main structure for the time table itself. This is described in section 4.2 below. The `vptt` contains a full or partial specification of all desired services which satisfy the major operational constraints of line and station occupancy, continuity of timings and rake consistency. The `vptt` will eventually refer to the entire timetable satsifying all relevant constraints, and at intermediate points in the timetabling exercise, will refer to a partial list of desired services, perhaps with a desired interval of start times at the origin of each service. Each service listed in the `vptt` will be identified with a particular pattern and when the timings of that service are fixed at the origin, the route of the service and the times at various stations en route will be fixed satisfying the relevant hold resource and line resource specifcations.

The `vptt` also contains the rake link information that is required for provisioning services with physical rakes.

3 Strategic and operational constraints in timetabling

This section takes up some practical considerations which arise while constructing time tables. Some of these refer to the modelling of constraints, while some refer to the actual process of timetabling, in practice. This is important because validation of constraints is sometimes possible or relevant only in the specific context of the procedure for constructing timetables. Some global constraints (such as headway and others to do with safety) are relevant in all formulations and solution procedures. An one extreme, a procedure could require a one time specification of all constraints and then invoke a global search procedure to suggest a valid timetable satisfying all constraints. Another extreme would be to constructively generate partial timetables and successively post constraints as they become relevant. The procedure currently used in the Indian Railways is largely manual (with some support for display and report generation) and is of the latter, constructive type. Our procedure, in principle, allows the user to choose from an entire range of procedures, ranging from a one time specification and generation of timetables and rake links, to an interactive, incremental one where the user can specify partial timetables and build up a desired timetable.

3.1 Strategic considerations

We begin with two strategic considerations in timetabling. One, the sequence or hierarchy of major decisions in the timetabling exercise and two, network level changes.

3.1.1 Sequence of decisions

A major difference in suburban timetabling as opposed to long distance timetabling is the following. In long distance time tabling, there are specific requirements of readying rakes for journeys, both from commercial and maintenance points of view. The turnaround time for rakes could be anything from 30 minutes to a day or more. In suburban operations, the demand is such that rakes are required to turn around in a matter of minutes (often less than 10). Suburban stations and termini are often in places where space is scarce and there are a limited number of platforms. Rakes do not have to be maintained or cleaned after every trip (often lasting only minutes or a few hours). Rakes need to be utilised effectively to reduce investments in rolling stock.

All this means that rake linking is an integral part of suburban timetabling, unlike in long distance timetabling, where it is a contributing but not necessarily integral part of the planning exercise. For a given set of services, the "optimal" rake linking decision may sometimes be straightforward. [Airline scheduling problems and other applications have also shown versions of this problem to be a minimum cost flow problem]. But as a part of a procedure for constructing timetables it poses a significant constraint.

In our procedure, rake linking and construction of schedules for services are treated at the same level of hierarchy and in fact deal with the same data structure, the valid partial time table. This allows the user to construct timetables in a variety of ways, for example, to retain old service patterns (perhaps with slightly changed times) and rake links and add new services, or to specify services and accept new rake links, or options in between.

3.1.2 Network level changes

Constraints referring to the timetable need to be modelled needs at two levels. At one level, we would like to allow operational constraints to be accurately modelled, verified and used in generating schedules. At another level (which we may call the design level), network constraints may require re-definition of the conditions representing the resources and demand constraints may require re-definition of the activities themselves.

In our language, these network level changes will be effected by modifying the structures Hold Resources (section 2.1), Line Resources (section 2.2) and Patterns (section 2.3) above. The changes could be of the following types:

- Introducing new hold resources – for example, by introducing stations in between existing ones,
- Introducing new line resources – for example, by increasing the number of tracks in between two stations, or
- Introducing new service patterns – for example, by allowing services to use a different combination of resources.

3.2 Major operational considerations

Operational considerations in timetabling refer to changes in a timetable for a fixed network. In our terms, they refer to the changes in the `vptt` (section 2.4). There are three major operational considerations that must be satisfied by any timetable. In addition, there are several commercial considerations that drive the timetabling process. These and some secondary operational concerns are also briefly discussed below.

The major operational constraints are three in number, one each to do with static resources (track) and moving resources (rakes) and one to do with the continuity of activities (services). They are all modeled using the constructs of hold and line resources (to represent track resources), patterns and specific instances of patterns (to represent services), and a specific activity graph on which rake linking is represented.

- **Timetable row constraints (track resources):** These refer to the constraints that a given unit of resource (basically track sergment and station platform, can be used by one activity at an instant of time. These constraints are modeled in Section 5.2. In terms of the physical time table, they refer to the fact that, for example that,
  - no more than 4 trains can occupy Grant Road station (Hold Resource) at any point of time, and that
  - no more than one train can occupy, at any time, any 3 minute long segment of the Line Resource between Grant Road and Mumbai Central. Here, 3 minutes is the permissible headway between trains on that track.
- **Timetable column constraints (services):** This constraint links the arrival at one resource (e.g. station) with the departure at the preceding resource (e.g. track segment). These are modeled in Section 5.3.
- **Rake linking constraints (rolling stock resources):** This set of constraints ensures that a service is provided with a rake of the appropriate type. This is initiated in Section 5.3 and the various options of specifying this constraint are detailed in Section 6.2

3.3 Other operational constraints

We now list several other operational considerations which must be satisified during timetabling.

3.3.1 Terminal constraints and turnaround

Suburban services operate to and from business and residential areas of high population density and space is often a constraint at terminal stations. Also, for better utilisation of rakes (rolling stock), it is necessary to have as little time spent waiting at terminals as possible. Further, since individual trip times are small, there is no need to perform detailed maintenance activities on coaches after every single trip. On the other hand, rakes are required at low levels of service during the night hours, unlike long distance services.

All this leads to the constraints that terminal constraints are significant. The number of platforms at termini and the time required for turnaround are major constraints. For example, on the Bombay suburban network, Churchgate station has four terminating platforms. Minimum turnaround time there could be as low as two minutes (governed by time required for passengers to alight and board), because it is a crew change point, and a fresh crew of motorman and guard can be made available, if required, for every outgoing service. The time for crew to physically move to the other end of the rake is therefore saved. At other terminals, similar constraints apply.

3.3.2 Stoppages at stations

In our framework, scheduled stoppages are only at stations. This means that trains are not scheduled to stop anywhere in between stations. In fact, on most track segments, train speeds are determined and uniform for all types of trains using that track segment. This corresponds to the fact that, by design, trains are kept moving, except at stations, where there are platforms and provisions for passengers to alight and board. The stop duration on a platform at a station may be variable, within some pre-specified limits.

3.3.3 Platform accessibility

The other constraint that can quite easily be modelled is on the number of platforms available for certain activities. In suburban operations, at most stations, there is a single designated platform for most types of services, but at some stations (especially termini), there are multiple platforms that can accommodate a certain type of service.

3.3.4 Headway on track segments

Safety requires that trains on the same track segment have to be a minimum time apart. Ideally, this safe distance between pairs of trains is determined by the braking distance of the train following, and the positional guarantee provided by signalling, regarding the leading train. This is simplified and homogenised in suburban operations, since there are only a few types of trains, and streamlining of operations by providing common operating conditions is itself a major objective. What this amounts to is that each track segment can be thought of a permitting trains at safe intervals of k minutes.

3.3.5 Merging traffic

Where traffic from one or more possible streams merges to form a common stream, a safe headway condition is imposed on the common track segments that form the join (to use the terminology defined earlier). In practice, since merging traffic will use a common track segment at some point, a safe headway condition on that track segment can also provide for safe operation, but we prefer to propogate this safe headway backwards to the previous station (halt resource) so that there are no stoppages on lines.

3.3.6 Diverging traffic

Where traffic diverges to form two or more streams, the headway constraint imposed at the common line resource is enough to ensure feasible running of trains.

3.3.7 Resource blocks

Resources, either at stations, or on tracks, could be blocked for a variety of reasons. Such blocks could be external to the timetabling procedure, as listed below, or could be blocked during the timetabling procedure, in case we decide to freeze the times of certain services. The impact is in terms of headway constraints on trains following them, and in terms of platform unavailability during some time interval. These could arise because of any of the following reasons.

- **Long distance trains:** Long distance trains may use resources that are also used by suburban services. Their timetable is independently determined (often subservient to the suburban timetable), but once fixed, it would not normally be distrubed during incremental changes in the suburban timetable.

- **Level crossings:** Scheduled openings of level crossings would block line and sometimes hold resources (if level crossings are in the precincts of a station), for some length of time.

- **Platform unavailability:** Platforms at stations may not available for some length of time, for commercial or other reasons.

- **Maintenance blocks:** Platforms at stations or track segments are taken up for integrated maintenance and traffic is diverted during some interval of time. This is often done on Sundays or other holidays.

Resource blocks are handled by maintaining a consolidated list of time intervals when line or hold resources are unavailable. These can be incorporated into the row constraints mentioned in Section 3.2.

We now turn to a precise description of the timetable, with appropriate data structures for the resources, patterns and the timetable itself. The main operational constraints in this section are modeled in the next two sections (4 and 5). We show how to model the other operational constraints in the subsequent sections.

4 System Architecture

This section describes the basic software architecture of our timetabling tool. As in the picture below, the core of the tool are two independent modules, the `scheduler`, for detailing and operationalising an input timetable, and the `rake-liner`, for rake provisioning. Both these modules use as input, a data-structure called the *valid partial time table*, or the `vptt`, and output into the same format. Thus the two modules may be used iteratively to arrive at a desirable time-table.

The other inputs used by the two modules are the (i) network data describing the line and hold resources (stations and tracks), their geometry and operational capacities, and (ii) the pattern of operations, various turn-around options, and so on.

The two modules are surrounded by 'gluing' code which translates user requirements into specific data structures and constarints to be used by the two modules. Besides this, the code also manipulates the user interface.

4.1 The Network and Pattern of Operations

As mentioned before, the basic building blocks of the network are the *hold-resource* and the *line-resource*. These model, resp., a station and a length of track. The basic data structures for both appear below.

4.1.1 The Hold Resource

hold-resource-id	No. of Platforms	Stable Size	Turnaround
`Bandra`	6	4	6

Table 1: A typical Hold Resource Entry

The entry above indicates to the system that there is a hold resource called `Bandra` (which happens to be a station on WR), and which has 6 platforms. Every service which passes through a hold-resource, defines an activity on it, corresponding exactly to the halt interval of the service at that station. The system ensures that in a time-table produced by it, at no point are there more than 6 trains at `Bandra`. All platforms in a hold-resource are assumed to be equally accesible to every line-resource terminating there. If this is a serious modelling inaccuracy, it is possible to 'split' `Bandra` into sub-groups[1] of platforms which are equally accesible. However, the line-resources terminating or starting at `Bandra` also need to be 'split'. The stable-size indicates the stabling capacity at that station. Rakes may be pulled out of service and stables at these locations. The turnaround time (in minutes) denotes the minimum time which must elapse between two services operated by the same rake. This is typically a function of the station.

4.1.2 The Line Resource

The line resource is the second building block, which represents a physical **unidirectional** track segment of the network. Two typical line-resources are shown below. The start and the end stations (hold-resources) are indicated. The length of the track specifies (in minutes) a feasible range for a train service to cover the distance. The headway indicates that two successive trains entering the line-resource must be separated by the headway. A service utilising a certain line-resource defines an activity on the resource corresponding to the interval $[t - headway, t + headway]$ where t is the time of entry of the service into the line-resource. The fork-list and the join-list indicates if there are other line-resources with which this line-resource shares, either its initial segment (a fork) or its final segment (a join). An activity on a line resource forces corresponding ones on entries in the fork-list and the join-list. More on this, later.

It may be that the chosen track segment has several intermediate stations, however they do not play a role in the time-tabling process. A line resource then is a useful device to 'contract' the network, as we have done with WR.

[1]It may be that the subgroups intersect, e.g., `Bandra-12car` and `Bandra-North`. More on this, later.

line-resource-id	
BAN-DDR-Local	
Start	Bandra
End	Dadar
Length	[10,11]
Headway	3
Forklist	-
Joinlist	-

line-resource-id	
ADH-BVI-Cross	
Start	Andheri
End	Borivili
Length	[21,24]
Headway	4
Forklist	ADH-BVI-Thru
Joinlist	ADH-BVI-Local

Table 2: Two typical Line Resource Entries

4.1.3 The Pattern

The pattern denotes a sequence of movement of a particular service. This is a list alternating in hold-resource and line-resource, begining with the starting station (hold-resource), and ending with the terminating station. For each station, an interval of allowed stoppages is specified in minutes. Thus for example, the `9-CCG-BVI-Ft` pattern begins at `Churchgate` and terminates at `Borivili`. At `Andheri`, for example, it is allowed to stop between 1 and 2 minutes.

pattern-id		
9-CCG-BVI-Ft		
Station	Stoppage	Line
Churchgate	-	CCG-BCT-Thru
BombayCT	[1,1]	BCT-DDR-Thru
Dadar	[1,2]	DDR-BAN-Thru
Bandra	[1,1]	BAN-KHR-Thru
Khar	[0,0]	KHR-ADH-Thru
Andheri	[1,2]	ADH-BVI-Cross
Borivili	-	-

Table 3: A typical Pattern

4.2 The Valid Partial Time-Table

This is the common input and output format, for both the `scheduler` and the `rake-linker`. The basic formal consists of a table with a variable number of columns and a fixed number of rows, which depend on the network and operational input, as above. A sample `vptt` is shown below:

Each column represents either a desired, proposed or scheduled service. The first two rows indicate the service number (identifier) and its pattern of operation. The *start-time* indicates the interval within which the service *must* start from its origin point. The rake-link is a service number, and is optional. This number indicates, *e.g.*, that the rake which does `BVI-647` follows it

service-id pattern-id	BVI-647 9-CCG-BVI-Ft		PROP-3 9-BVI-CCG-Su		ADH-751 9-CCG-ADH-Sw		BVI-616 12-BVI-DDR-Ft	
start-time	18:20	18:26	19:24	19:30	20:19	20:24	17:50	17:50
rake-link	PROP-3		ADH-751					
Churchgate	*	18:25			*	20:19	-	-
BombayCT	18:34	18:35			20:28	20:29	-	-
Dadar	18:41	18:42			20:39	20:40	18:29	*
Bandra	18:47	18:48			20:48	20:49	18:20	18:21
CST	-	-			-	-	-	-
Bandra-H	-	-			-	-	-	-
Khar	18:50	18:50			20:51	20:52	18:19	18:19
Andheri	18:55	18:56			21:04	*	18:10	18:11
Borivili	19:18	*			-	-	*	17:50
Virar	-	-			-	-	-	-

Table 4: A Section of a `vptt`

with `PROP-3`. The succesive rows are again optional, and represent (respectively), the arrival and departure times of the service at the specified stations, as dictated by its pattern of operation. It is assumed that for a service, either all arrival-departure timings have been specified (in which case, we call it as *scheduled*), or none.

The data in a `vptt` must be *consistent*. This may be enforced by checking the following conditions:

1. For every scheduled service, the timings specified respect the apttern of operation specified for that service. In other words:
   - The halt at every station is within the interval specified by the pattern. Further, the service only visits those stations specified in the pattern.
   - The line resource used by the service from one station to another are assumed to follow the pattern. Under this assumption, the transit times respect the transit times as specified for that line-resource.
2. Every entry apearing as a rake-link for some service, must itself have an entry in the `vptt`. If service s_2 follows service s_1 in the rake-link, and s_1 is scheduled, then the start interval for s_2 must respect the turnaround time at the common station at which the linking is taking place.
3. The collection of scheduled services together define activities of every line resource and hold resource. These activities must be feasible at each resource.
   - The number of platforms at every hold-resource exceeds at all moments, the number of trains halting there, at that moment. In the calculation of the activities on the hold

resources, besides the scheduled services, those which correspond to the occupation of the platform during rake-links, are also included.

- Every service defines activities on every line resource that it uses, and on entries in its fork-list and join-list. All these activities must be mutually disjoint.

4.3 Resource Blocks

This is a list of time-intervals during which a resource may not be used. These arise from causes which are not within the time-tabling regime, such as long-distance train movements, platform blockages for maintenance, and so on. This data is made available to the scheduler as two tables, *viz.*, the line-resource-block, and the hold-resource-block. A typical set of line resource blocks is shown below. The first entry indicates that the entry to the `BCT-DDR-Thru` line resource is blocked from 12:47 to 12:53. The reason is the movement of the long-distance train service No. 1905.

Line-Resource-Block			
Line-Resource	Block-start	Block-end	Remarks
`BCT-DDR-Thru`	12:47	12:53	1905
`DDR-BAN-Thru`	12:57	13:03	1905
`BAN-KHR-Thru`	13:06	13:13	1905
`KHR-ADH-Thru`	13:07	13:13	1905

Table 5: A typical list of blocks created by the movement of the Vapi-Express, No. 1905

5 The Core Scheduler Definition

The scheduler takes as input a *vptt*, say `vptt1`, and outputs another *vptt*, say `vptt2`. The basic objective of the scheduler is to assign detailed timings to every service which is as yet unscheduled in `vptt1`. Thus `vptt2` will have the same rows as `vptt1`, except that it will, optmistically, have all its services as scheduled. The scheduler problem is formulated as a search for a feasible set of arrival and departure timings which are subject to various linear constraints, and certain global contstraints which arise from the modelling of utilization of tracks and platforms as activities on appropriate resources.

5.1 The basic functions

To begin, with `vptt1`, with every service s in `vptt1`, we have the data (i) $pattern(s)$, the pattern of service s, (ii) *start-time*(s), which is the interval within which s must start, (iii) *rake-line*(s), the service which follows s, if any, (iv) $init(pattern(s))$, the initial station (hold-resource) of s, and (v) $term(pattern(s))$, the terminal station of s. All this data is either explicitly available in `vptt1` or is deducible from the pattern descriptions.

Given a hold-resource `hr` (or a line-resource `lr`), we say that `hr` $\in pattern(s)$ (or `lr` $\in pattern(s)$) if the pattern $pattern(s)$ utilizes that resource. For a hold-resource `hr`, we have the associated functions $platforms(\texttt{hr})$ and $turnaround(\texttt{hr})$, which associate, respectively, the number of platforms (as an integer) and the turn-around time (in minutes), as assigned in the definition of hr. One similarly has functions $start(\texttt{lr})$, $end(\texttt{lr})$, $length(\texttt{lr})$, $headway(\texttt{lr})$, $forklist(\texttt{lr})$ and $joinlist(\texttt{lr})$ for each line-resource lr. All these functions return an appropriate data-structure, *e.g.*, $forklist(\texttt{lr})$, a list of line-resources. For a pattern, say pattern, we have already seen the function $init(\texttt{pattern})$ which is the hold-resource marking the initial station of the pattern. Besides this, we have $term(\texttt{pattern})$, the terminal station of the pattern, and $stoppage(\texttt{hr}, \texttt{pattern})$, the allowed stoppage for the pattern at the hold-resource hr (provided, of course, we have $\texttt{hr} \in \texttt{pattern}$). Note that all these functions associated with resources and patterns, are defined purely from the data associated with the network and scheme of operations, and does not depend on the input vpttl.

The set $Time$ will denote the discrete set of instants of interest and will be equipped with a total order $\leq$. For any two instants t_1 and t_2 in $Time$, we say that t_1 is before t_2 if $t_1 \leq t_2$. For instance, in WR, $Time = \{hh : mm | hh : mm \neq 02 : 00\}$, and the total order $t_1 \leq t_2$ is defined if $[t_1, t_2]$ does not contain the point $02 : 00$.

5.2 Variables and Resources

The scheduler proceeds to create variables $s(\texttt{hr}, a)$ and $s(\texttt{hr}, d)$, for each hr in $pattern(s)$. These variables correspond to the arrival and departure times of the service s at the hold-resource hr. Of special importance are the variables $s(init(pattern(s)), d)$, the departure time of s at its initial station, and $s(term(pattern(s)), a)$, the arrival time of s at its terminal station. These collection of variables will be collectively denoted by $\mathcal{S}$.

Next, for every hold-resource hr, a collection of activities $\mathcal{A}(\texttt{hr})$ is defined. Each activity is represented as a time interval parametrised by the variables $\mathcal{S}$ as specified below. For every s such that $\texttt{hr} \in pattern(s)$ but is not the initial or terminal station of $pattern(s)$, we include the activity $[s(\texttt{hr}, a), s(\texttt{hr}, d)]$ in $\mathcal{A}(\texttt{hr})$. Besides these, if $rakelink(s) = s'$, and $term(pattern(s)) = init(pattern(s')) = \texttt{hr}$, then we add the interval $[s(\texttt{hr}, a), s'(\texttt{hr}, d)]$ to $\mathcal{A}(\texttt{hr})$. Further, each entry of the Hold-Resource-Block table adds an entry to an appropriate $\mathcal{A}(\texttt{hr})$. For an assigment $T : \mathcal{S} \to Time$, we say that T satisfies $\mathcal{A}(\texttt{hr})$ if for every $t \in Time$, the number of intervals $I \in \mathcal{A}(\texttt{hr})$ such that $t \in I$, do not exceed $platforms(\texttt{hr})$[2].

The activities on each line-resource lr are a bit more intricate. Firstly, clearly, if $\texttt{lr} \in pattern(s)$, so are its start and begining hold-resources, $start(\texttt{lr})$ and $end(\texttt{lr})$. For every s such that $\texttt{lr} \in pattern(s)$, we add the interval $[s(start(\texttt{lr}), d), s(start(\texttt{lr}), d) + headway(\texttt{lr})]$ to $\mathcal{A}(\texttt{lr})$. Thus, effectively, the entry point of the line-resource is modelled as a resource. If, a service enters the line-resource at time t, then it blocks the entry for an interval $[t, t + headway(\texttt{lr})]$. As before, each entry of the Line-Resource-Block table adds an entry to an appropriate $\mathcal{A}(\texttt{lr})$.

The modelling of forks and joins is as follows: for every fork-list or join-list, we have an *owner* which is a chosen element of the list.

Thus, if $\texttt{lr} = owner(forklist(\texttt{lr}'))$ and $\texttt{lr}' \in pattern(s)$, then we add to $\mathcal{A}(\texttt{lr})$ the interval

[2]There is the weaker possibility of define activities as 'left-closed' intervals, instead of the 'both-closed' ones that we have chosen.

$$[s(start(\texttt{lr}'), d), s(start(\texttt{lr}'), d) + headway(\texttt{lr})]$$

This represents the fact that an entry into `lr'` is also an entry into `lr` since `lr` and `lr'` share the same initial segment. Finally, if $\texttt{lr} = owner(joinlist(\texttt{lr}'))$ and $\texttt{lr}' \in pattern(s)$, then we let r and R stand for the expressions

$$r = s(start(\texttt{lr}'), d) + min(length(\texttt{lr}')) - max(length(\texttt{lr})) \text{ and } R \text{ for:}$$
$$R = s(start(\texttt{lr}'), d) + max(length(\texttt{lr}')) - min(length(\texttt{lr})) + headway(\texttt{lr})].$$

We add the interval $[r, R]$ to $\mathcal{A}(\texttt{lr})$. This ensures[3] that no service enters `lr` which would conflict with the entry of s into `lr'`. The selection of owners for each list enables the translation of constraints on any element of the list onto the owner, and thus ensuring that no two of the resources of the list are in simultaneous use. The selection of the owner resource is usually routine. In our example network (WR), forks and joins arise when a cross-over track either leaves a main track or joins another. In either case, the owner is usually the main track.

As before, for an assigment $T : \mathcal{S} \to Time$, we say that T satisfies $\mathcal{A}(\texttt{lr})$ if for every $t \in Time$, there is at most one interval $I \in \mathcal{A}(\texttt{lr})$ such that $t \in I$.

5.3 The Pattern and Rake-link constraints

The pattern constraints enforce the operation of the service s along $pattern(s)$. For every $\texttt{lr}, \texttt{hr} \in pattern(s)$, we have:

$$s(\texttt{hr}, d) - s(\texttt{hr}, a) \in stoppage(\texttt{hr}, pattern(s))$$
$$s(end(\texttt{lr}), a) - s(start(\texttt{lr}), d) \in length(\texttt{lr})$$

In addition to this, if s has been rake-linked, and $rakelink(s) = s'$, then we have:

$$s'(init(pattern(s')), d) \geq s(term(pattern(s)), a) + turnaround(term(pattern(s)))$$

The basic definition of the scheduler is, on input `vptt1`, to create an instance of the problem, as described above, and to provide a solution $T : \mathcal{S} \to Time$ which (i) satisfies all the constraints above, and (ii) which does not alter the timings of those services which were already scheduled in `vptt1`. This solution T is output as `vptt2`.

At this point, we should note that there are many possible implementational choices for the core scheduler. Our own implementation is based on a commercial **constraint solver** call **CHIP C++**. Our scheduler, first hunts for a solution satisfying both (i) and (ii) above, and if that fails, one satisfying only (i).

Besides a *vptt*, there are frequently other inputs to the scheduler, further constraining the system. Typically, in the WR situation, these arise as scheduled maintenance activities on various resources, and, more commonly, scheduled movement of long-distance goods and passenger train services in the network. This movement has much more variety than that of suburban services, and is not easily amenable to 'patterns'. Our modelling of such movement is by making a $blocklist(\texttt{lr})$ and $blocklist(\texttt{hr})$, which are lists of intervals for each resource. Thus, the long-distance train movement is converted into scheduled activities on the resources. These activities are included in the sets $\mathcal{A}(\texttt{lr})$ and $\mathcal{A}(\texttt{hr})$.

[3] This is very conservative and some other weaker formulations are possible.

6 The Core Rake-Linker

The input and output of the rake-linker are again, *vptt*'s. On input `vptt1`, the task of the rake-linker is assign rake-links to as many services of `vptt1` as possible. In the process, it may propose some new services to 'link-up' the services specified in `vptt1`. A typical situation when this is useful, is specify the peak-direction services in `vptt1`, and link these up using the rake-linker. These proposed services, and the computed rake-links are output as `vptt2`.

6.1 Chain Decompositions in Directed Acyclic Graphs

The rake-linker problem and its variations have been addressed by many authors. In our typical situation (*e.g.*, with WR), the rake-linking problem reduces to the following graph theoretic problem: Let $G(V, E)$ be a directed acyclic graph and let $c : E \rightarrow \Re^{+}$ be a non-negative cost function. A *path* π in G is a sequence $v_0, \ldots, v_k$ of vertices in V, such that $(v_i, v_{i+1}) \in E$ for all $0 \leq i \leq k-1$. The cost of a path π is denoted by $c(\pi)$ and equals $\sum_{i=0}^{k-1} c(v_i, v_{i+1})$. A chain decomposition of G is a collection $\Pi = (\pi_i)_{i=1}^{r}$ of paths such that every vertex in V appears in exactly one path in the collection Π. The cost of a chain decomposition is denoted by $c(\Pi)$ and equals $\sum_{i=1}^{r} c(\pi_i)$.

Various computational problems associated with chain-decompositions are:
(i) Compute a miminum cardinality chain decomposition.
(ii) Compute a minimum cost chain decomposition.
(iii) For a fixed r, compute a minimum cost chain decomposition with r chains.
All these problems have straight-forward **min-cost flow** formulations and are solvable in polynomial time.

6.2 The Service Graph

The rake-linking problem is converted into computational chain-decompositions problems on the *Service Graph*. The service graph $G(V, E_p)$ consists of directed acyclic graphs on the same vertex set $V = \{s | s \in \texttt{vptt1}\}$, the services specified in `vptt1`. The edges E is a binary predicate $p : V \times V \rightarrow \{\texttt{true}, \texttt{false}\}$. Typical predicates which are useful in defining p are :

1. `after`(s, s') is `true` when s ends at the station where s' starts, and the departure time of s' from this station is after the arrival time of s by atleast the headway at that station. When either of s or s' is unscheduled, `after`(s, s') may be calculated by estimating the arrival/departure times from the pattern definitions, and checking if a rake-link is indeed possible between s and s'.

2. Let *upperlimit*(`hr`) be a sequence of intervals, one for each hold-resource. The predicate `upper-limit`(s, s') holds if `after`(s, s') holds, and further, the gap between the departure of s' and the arrival of s does not exceed *upperlimit*(*init*(s')). This formulation ensures that rakes are pressed into service rather quickly. This is especially important at stations where either stables are unavailable, or stabling is operationally expensive, such as `Churchgate` in WR during peak hours.

3. Frequently, each pattern has a *type* associated with it, and the predicate $\texttt{sametype}(s, s')$ holds if $type(pattern(s)) = type(pattern(s'))$. WR, for example, operates two types of rakes, *viz.*, the 9-car and the 12-car rake. Each pattern specifies the rake-type which must run that pattern. Clearly, therefore, s may be rake-linked to s' if both have the same type.

4. An important predicate is the $\texttt{one-hop}(s, s')$ which allows for the introduction of new services. Let $term(pattern(s) = \texttt{hr}$ be the terminal station of s and $init(pattern(s')) = \texttt{hr}'$ be the starting station of s'. We say that $\texttt{one-hop}(s, s')$ is true if there is a fictitious service s'' such that $s \to s'' \to s'$ would make a rake-link. The specification of this s'' is easy enough. Firstly, there must be a `pattern` such that $init(\texttt{pattern}) = \texttt{hr}$ and $term(\texttt{pattern}) = \texttt{hr}'$. Next, the gap between the arrival of s at `hr` and the departure of s' from `hr'`, must allow for the turn-around at `hr`, plying of the service s'' with `pattern`, and the turn-around at `hr'`. Finally, the interval of starting s'' at `hr` to effect the two rake-links is computed from the above data.

5. Finally, we have $\texttt{raked}(s, s')$, which is `true`, if either s does not have a rake-link in `vptt1`, or $rakelink(s) = s'$ in `vptt1`. Thus, this predicate will preserve the rake-links of the input `vptt1`.

The costing of various edges in the service graph above, is a matter of taste. We have costed edges which arise from `one-hop` at three times those which arise otherwise. Another typical costing is to penalise platform usage during rake-links, with a larger penalty if the rake-link is so long that it requires stabling. The output of the rake-linker is thus another *vptt*, and is output in that format.

7 The applications of the timetabling procedure

This section relates the modelling of the previous two sections 5 and 6 with the details of the procedure of actually constructing timetables. We first list several ways in which the timetabling tool can be used. Later in this section, some further operational constraints are described, which makes the modelling more accurate.

7.1 Uses of the timetabling tool

The entire timetable is generated afresh about once a year, in most suburban operations in India. This itself is a cumbersome exercise, which takes a lot of skilled effort and manual generation of a number of scenarios. We discuss this and other incremental, possible uses of the timetabling tool.

7.1.1 Introduction of new services

The most common incremental application of the timetabling procedure is to introduce new services to (or perhaps withdraw some services from) an existing timetable. Our structure of a valid partial timetable (`vptt` in Section 4.2) will allow for addition of services while keeping existing services fixed (or essentially fixed) very easily. The interval for the starting time of every service in the `vptt` makes it easy to fix some times if required, and permit minor changes if allowable.

7.1.2 Changing rake links

If desired, the rake link for a particular service can be changed, either by specifying the new link in the `vptt` or more generally, by redefining the edges in the service graph (Section 6.2). For example, if an exisiting 9 car rake service is to be provisioned by a 12 Car rake, the new service graph can be quickly constructed. See the subsection below on rake integrity for how the new service graph is created in this case.

7.1.3 New timetables

The core scheduler and rake linker equally well allow the construction of a new time table (starting from scratch). The inputs would be the network description and the desired timetable in terms of patterns and the desired intervals of departure (in other words the `vptt` with the first three rows filled in as an initial suggestion. The output would be (perhaps after some iterations) the completed timetable (column) for each scheduled service, some new services (as proposed by the rake linker) and a set of consistent rake links.

7.1.4 Complete timetable

One of the advantages of our scheme is that networks can be suitably compacted and major timetabling decisions can be taken on the reduced network. While manipulating the timetable and posting constraints, it is often enough to do so on this reduced network. Exploiting the standard nature of patterns, and largely standard usages of resources, it is then quite convenient to expand the timetable to get detailed timings at all stations. In our terms this means the implementation of the column constraints (Section 3.2 and 5.2).

7.1.5 Sectional views and abstract timetables

This is the reverse of the above requirement, where the user may simply need to see some parts of the timetable. This could refer to a collection of services (e.g. all Virar fast locals) or all services at a collection of stations. This just needs indexing on some subset of columns and rows of the `vptt`.

7.1.6 Combined suburban and long distance timetables

If long distance services share parts of the suburban network, they may need to be timetabled as well. Since there are relatively few long distance services, and their usage of resources by these services is non-standard, we find it most convenient to specify their usage in terms of resource blocks (Section 3.3.7).

7.1.7 Maintenance block timetables

This is another aspect of suburban timetabling that deserves attention. Since suburban operations are often quite intensive, maintenance of the resources is a difficult task, sometimes. The network in Mumbai relies on integrated maintenance blocks, wherever multiple routings are available, to provide uninterrupted time (4-5 hours) while still maintaining some services. In the case of Mumbai, one challenging task is to prepare temporary timetables for these periods. The major tasks are two: one is to redefine the network and therefore the patterns which might use some of the affected resources; the other is to propose a restricted set of services that are to be run. Our framework allows for a precise answer to the first task, and to support the second task by autmatically checking feasibility and permitting quick modifications of proposed services (including rake links).

7.2 Further operational considerations

Here, we discuss some further details of suburban rail operations, apart from the major ones outlined in sections 3.1 and 3.2 and modeled in Sections 4-6.

7.2.1 Terminal constraints and turnaround

Suburban services operate to and from business and residential areas of high population density and space is often a constraint at terminal stations. Also, for better utilisation of rakes (rolling stock), it is necessary to have as little time spent waiting at terminals as possible. Further, since individual trip times are small, there is no need to perform detailed maintenance activities on coaches after every single trip. On the other hand, rakes are required at low levels of service during the night hours, unlike long distance services.

Turaround time is an attribute of a hold resource (station). In terms of the `vptt` and the rake links, platform availability imposes a global constraint on the hold resources at the station and the turnaround time imposes a linear inequality constraint on arrival times of services and departue times of the linking service.

7.2.2 Stabling

Rakes need to be withdrawn from service when not in use. This happens typically at the end of the day, for almost all rakes. Some rakes are also withdrawn during the mid day slack period. During both of these times, some rakes are taken to the car shed (where lines are assumed to be available in sufficient quantities) and some rakes are parked on lines just outside stations. The number of such lines at each station that are available for stabling is limited, and is specified as an attribute of the relevant hold resource.

For the timetable, stabling imposes certain end constraints on a valid time table. [*This has not been explicitly modeled so far.*]

7.2.3 Rake integrity

To meet growing and varying types of service requirements, there are different types of physical services that could be offered in a time table. For example, in Mumbai, both 9 car rake and 12 car rake services are offered. For demand reasons (or for operating reasons such as available length of platforms), certain services may have to be restricted to be of a certain type. Such constraints are easily imposed while construcing the rake linking service graph.

7.2.4 Conflicting traffic

In the suburban sections that we have considered, implicitly, we have assumed that one track is reserved for traffic in one direction. [4] Wherever change of tracks is involved, there is a chance that conflicting traffic (in the other direction) is affected. To forbid such conflicting movements, in practice, foolproof interlocked signalling is used. In the current practice of modelling the timetable, there is no direct scheduling of such cross movements (they are few in number) and they are accounted for by a traffic allowance of a minute or two, in the running time just before the terminal station (which is where the maximum conflicting traffic movements take place).

In our case, the preliminary model ignores this minor correction. A more detailed model will require a separate line resource to be used at the end of each terminating platform.

7.3 Allied decisions, reports and statistics

Although we have considered the working timetable and the rakelink set as the two major outcomes of the timetabling exercise, there are other decisions that are linked to timetabling. These decisions may be carried out subsequently, but the timetabling process should allow for reports and statistics that enable these decisions to be effectively made. Clearly, the most effective way of implementing this is by querying an appropriately designed database which stores the relevent `vptt` and the master tables to do with hold resources, line resources and pattern descriptions.

7.3.1 Scheduling of level crossing gates

Using the mechanism of resource blocks, the opening of level crossing gates can be modelled. The issues of interest here are (i) whether a scheduled opening of a level crossing at some times of the day (e.g. for 2 minutes in every half hour) will still permit a feasible timetable (`vptt`) and (ii) what time intervals for level crossings are permitted by a given `vptt`.

7.3.2 Rake link analysis

Rakes assigned to services are required for their physical operation. Apart from a cost based objective of minimising the number of rakes for a given set of services, there could be objectives such as the overall balanced usage of rakes. This translates into a balance condition on the cycles corresponding to each rake (section 6.1 and 6.2). In some constraint representation tools, this is possible to specify.

7.3.3 Rake maintenance and rotation

Rakes are to be routinely taken to car sheds for maintenance, and rakes of the same vintage should ideally be rotated in service so that maintenance schedules are more balanced. For timetabling, the implication is that the end of day services have to be planned so that (i) an appropriate number of rakes are at various car sheds or maintenance centres and (ii) rake cycles should originate and terminate at different locations and it should be possible to define an overall circuit covering different

[4] Single track services can also be modelled in our framework, but is not discussed here.

rake cycles. This has not been modeled so far. The rostering problem in crew scheduling is a more complicated version of this problem.

7.3.4 Inputs as frequency of service

The timetabler may like to provide a list of desired services, not in terms of intervals of departure for each service, but as a frequency for each type of service. A separate procedure will have to determine the order of departure at various stations in order to respect headway constraints and meet the desired frequency as best as possible.

7.3.5 Cyclical timetables

This is an extension of the above concept, where we could attempt to fulfill all demands by creating a compact repeatable schedule. This may not always be possible, but our tool allows the analysis (not generation) of such schedules as well.

7.3.6 Crew scheduling

In addition to the modelling described in the previous sections, we need some efficient ways of managing the data involved in the whole process. This and some other implementation aspects are taken up in the subsequent, final section.

8 Database Design Functionalities

8.1 Scheduling Operations

For our purposes, the Time-Table consists of the following two tables *vptt* and *dt*, of formats `Vptt` and `DetailTime`. The pair $vptt, dt$ is called *valid* if it satisfies all the scheduling and rake-linking constraints mentioned earlier. In other words:

1. Each row in *vptt* corresponds to a desired service, which may or may not have entries for it in *dt*. When entries are indeed present in the *dt*, then they respect the starting-time interval and the pattern description as mentioned in the *vptt*.

2. Rake-links, if specified, are respected. In other words, if service s mentions s' as its rakelink in *vptt*, then depending on whether either/both of them have entries in *dt*, the start-intervals, or detailed timings should such that the rake link is feasible.

3. The detail-timings of those services which have such entries, must utilize the line and hold resources correctly. The forks and joins, and resource blocks must be included in this analysis.

We now describe desirable utilities operating on a pair $(vptt, dt)$.

- $check(vptt, dt)$: returns an empty list if $(vptt, dt)$ is valid. If the pair is invalid, then it returns a structure listing out the points of invalidity.

- $merge((vptt, dt)(vptt', dt'))$ inputs two valid pairs and merges the tables, whenever possible.

- $postpone(s, T, vptt, dt)$: the input is a service s in a valid pair $(vptt, dt)$ which appears in dt, and a positive integer T. The function produces a valid pair $(vptt', dt')$ wherein s is moved ahead in time by T minutes. This function, as a side-effect, may postpone several other services. The output should be lists of services affected.

- $prepone(s, T, vptt, dt)$: The variation of *postpone* for negative values of T.

- $force(s, vptt, dt)$: The input is a pair $(vptt, dt)$ such that s has entries in dt, and thus has detailed times. Further $(vptt, dt)$ is such that *minus* the service s, the pair is valid. $force$ outputs a valid pair $(vptt', dt')$ which does not change the detailed timings for s. The function may call *prepone* or *postpone*.

- $insert(s, vptt, dt)$: The input is a pair $(vptt, dt)$ such that s has entries no dt. The function inserts detailed timings for s into dt following the prescribed pattern. Note that if $(vptt, dt)$ were valid, then the output may not be so, however, now $force$ may be applied.

8.2 Display Functions

This collection of functions relate to the effective display of various tables.

- $hold\text{-}time\text{-}table(vptt, dt, S, H)$: On input a pair $(vptt, dt)$, a list of services S, and a list of hold-resources H, produces a time-table for the services in S at the stations H displayed in the order of the two lists.

- $display\text{-}rake\text{-}link(vptt, dt)$: outputs the services in the pair arranged by rake links.

- $summary(vptt, dt)$: produces a summary report of the input pair. This may contain pattern-wise movement summary, rake-link summary and other aggregate quantities.

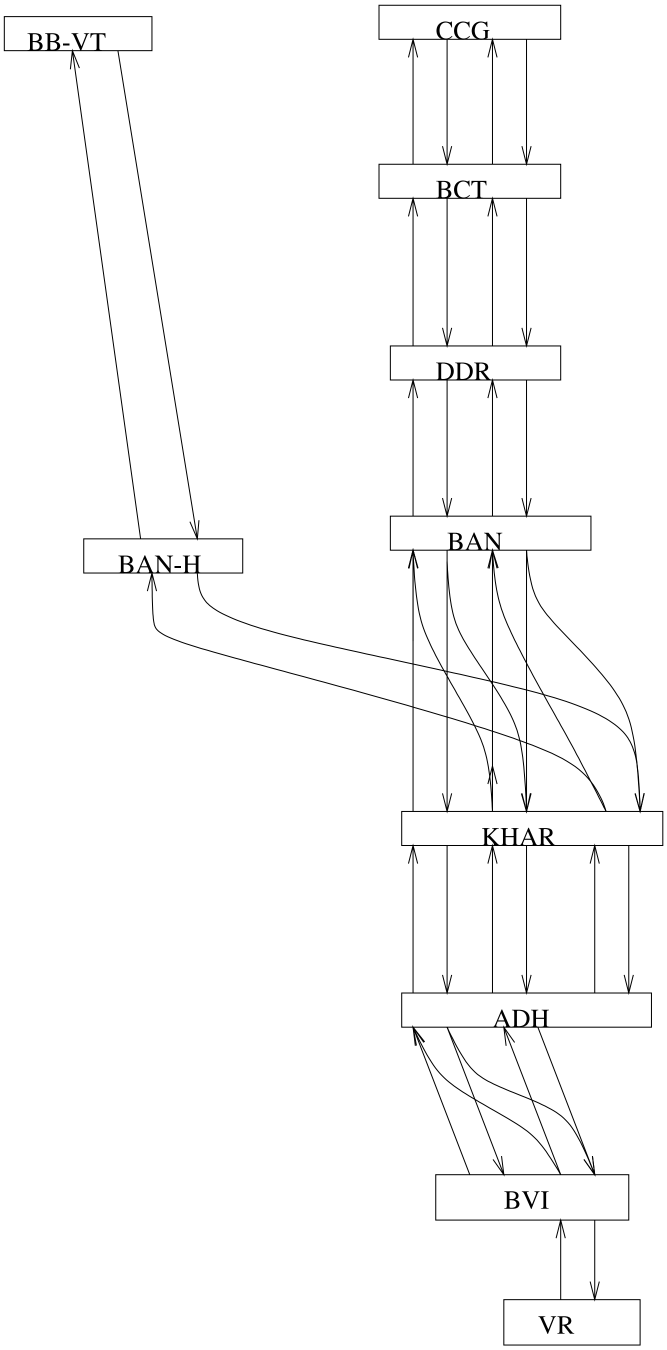

Figure 1: The Mumbai Western Railway (WR) Network