MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 596 September 1980

FUNDAMENTAL SCHEME FOR TRAIN SCHEDULING
(Application of Range-Constriction Search)

Koji Fukumori

Abstract. Traditionally, the compilation of long-term timetables for high-density rail service with multiple classes of trains on the same track is a job for expert people, not computers. We propose an algorithm that uses the range-constriction search technique to schedule the timing and pass-through relations of trains smoothly and efficiently. The program determines how the timing of certain trains constrains the timing of others, finds possible time regions and pass-through relations, and then evaluates the efficiency of train movement for each pass-through relation.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

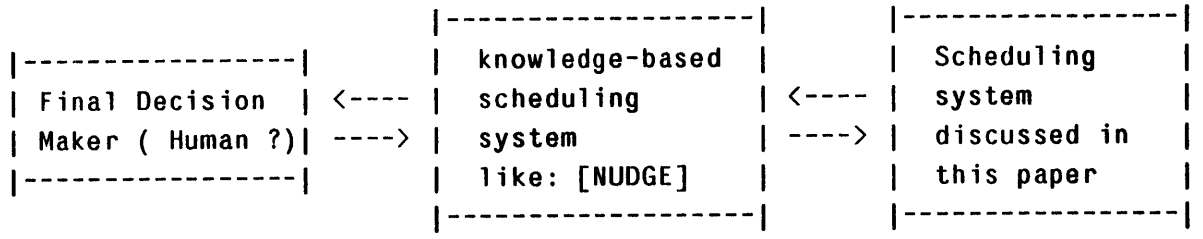
I.Introduction

Can a computer compile a timetable for a railroad that has several classes of trains on the same track? Traditionally, this question has been answered in the negative; that compilation of timetables has been one of the things done by expert people, not computers. The reasons cited are, first, that economic and social facets are basic to the problem; and second, even if the problem is limited to setting the timing of trains, the algorithm is still not clear.

In partial response to this question, we propose an algorithm, a combination of the concept of the settable time region and the propagation of constraints, which makes the so-called range-constriction search technique applicable in setting the timing and the pass-through relation of trains (hereafter we call this "setting of train movement") in terms of smooth and efficient traffic flow. By making clear how the trains constrain the settable timing of each other under a certain order relation, the computer can find possible pass-through relations between them and their settable time regions and can evaluate the efficiency of the trains' movements for each pass-through relation. These two concepts make the search space of possible combinations small, and also make the settable time region of each train small.

As mentioned above, it is true that the compilation of the railroad time-table includes economical and social facets which control the fundamental outline of a time-table -- the direction, the speed and the service interval -- in order to conform, as far as possible, to the traffic demand along its route, to the technical installations, to the local conditions, and to the economic requirements, and usually involves ill-defined, possibly incosistaint requirements. When we think about this fact, it may be said that the setting of train movement is only one sub-system of the compilation. We also feel the necessity of another weapon, such as a knowledge-based scheduling program [NUDGE: Goldstein & Roberts], besides such a "power-based" scheduling program, for the total time-table compiling system.

However,it is also true that the setting of train movement is still an essential part of the compilation and has been considered a difficult problem for computers even after being given the service interval of each train class in several time-zones, the running speed (or the running time), and the least necessary stopping time for loading/unloading.



In this paper we focus on the setting of train movement (the system in the rightmost box) for the total compilation of a long-term time table for high density traffic service with multiple classes of trains.

II. A train diagram and manual scheduling method

When a timetable is compiled manually, a train diagram is used (see Fig. 1).

☐ **Representation on the train diagram**

Train movements take place both in time and in distance. It is conventional that time is represented on the horizontal axis and distance on the vertical axis. Fig. 1(a) shows the train movements from station "A" at time "t_A" to station "B" at time "t_B" and then to station "C" at time "t_C". Note that we assume the time of acceleration or braking and the variation of running velocity are small enough so that train movement in a section[1] can be drawn as a straight line. The pass-through relation is also diagrammed as is between train "TR_a" (pass-through) and train "TR_b" (passed-through) at station "B", and the opposite direction train is diagrammed like train "TR_c". Fig. 1 (b) shows a part of an actual train diagram.

At present, train scheduling is a trial-and-error process, done manually by humans using a preliminary train diagram. A human uses a visual representation to interpret the relations between trains and stations (orders and positions in time and distance). The basic relations are represented by the three links listed below.

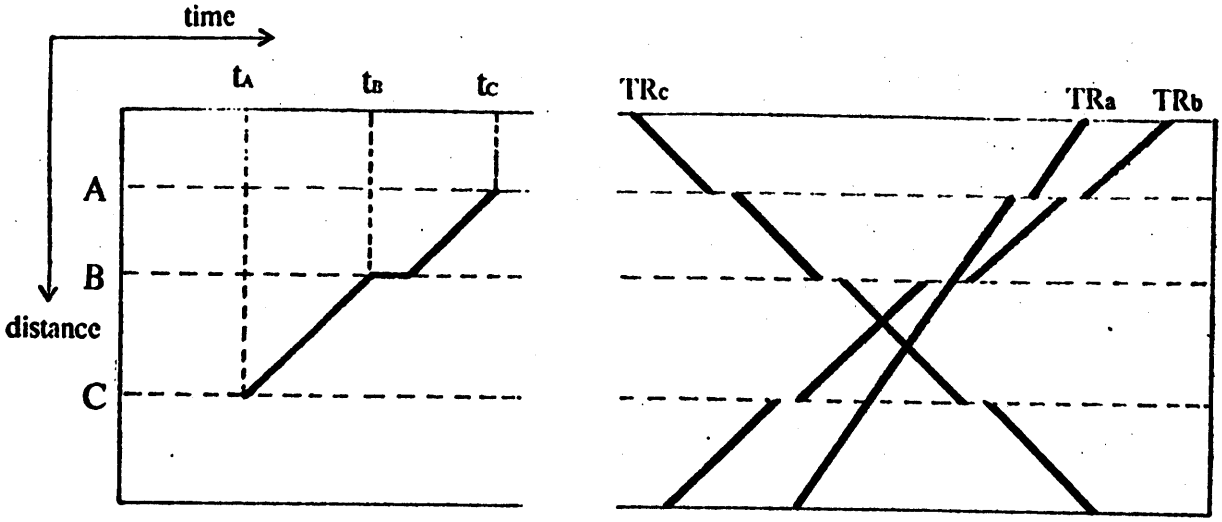
☐ **Three links which are used to access a train diagram**

horizontal line ------ Which train arrives before which train at this station.
vertical line ------ Which train is running before which train on track at this time.
train-line[2] ------ Which train arrives at which station at what time.

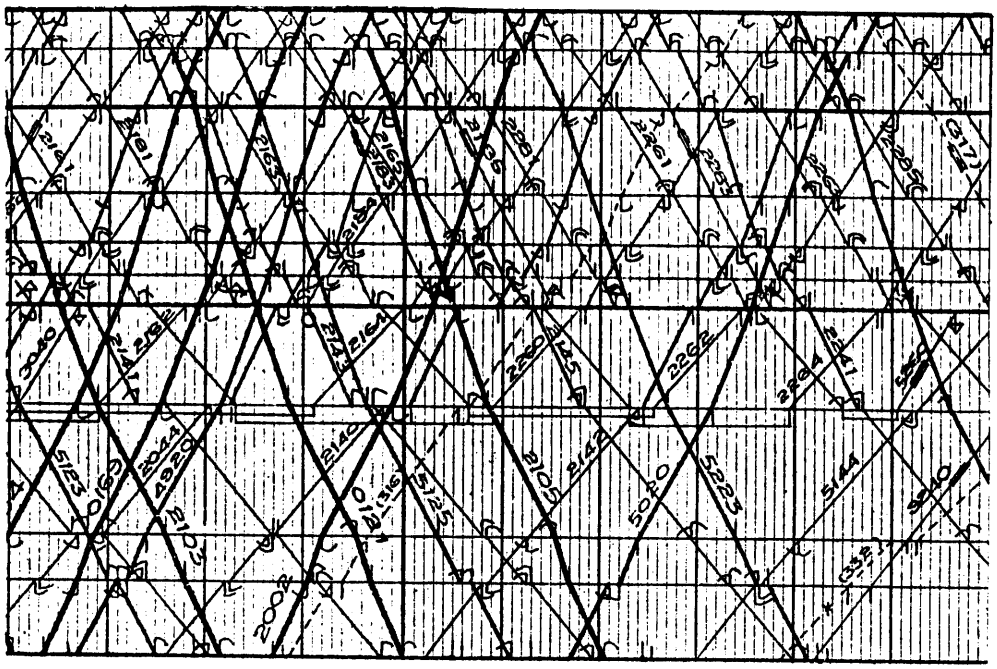
This means that people can see the train movement and relations without calculation, say, for express "TR_E" to have a pass-through relation with a normal "TR_N" at station "P", "TR_E" should be running around section "R" at this time, or should depart station "Q" around this time. This helps the scheduler to discard infeasible combinations by visualizing train movements, usually without actual drawing.

---

1. Track between two adjacent stations
2. For convenience, we use the conventional word "train-line" to express a trace of a movement of a train in the time-distance plane.



(a) Train movement represented on a train diagram



(b) A part of an actual train diagram[3]

Fig. 1 Train movements and train diagrams

3. For convenience, hereafter we assume as below in any figure which is represented in geometrical-form,
time axis = horizontal line
distance axis = vertical line.

III.How to implement trial-and-error and how to represent train movements in the computer

There are some alternative methods for maintaining a train movements data base. They are based on the access links mentioned previously.

☐ Train-line: For each train, we maintain an ordered list of the arrival and departure time set at each station. Each entry is of the form

```
( arrival-time  departure-time )
```

The horizontal line (order relation of trains) can be obtained from train-lines directly, because for all train-lines, each entry is ordered by the station order that never changes.

☐ Horizontal line: For each station, we maintain two ordered lists, one is of the departure time and another is of the arrival time at the station, with the train identification. Each entry for each list is of the form

```
( time  train-id )
```

The train-line can be obtained from horizontal lines theoretically, but the train identification is also needed, because the train order can change at each station.

☐ Vertical line: For each small time unit, we maintain an ordered list of the positions on a track with the train identification. Each entry is of the form

```
( position  train-id )
```

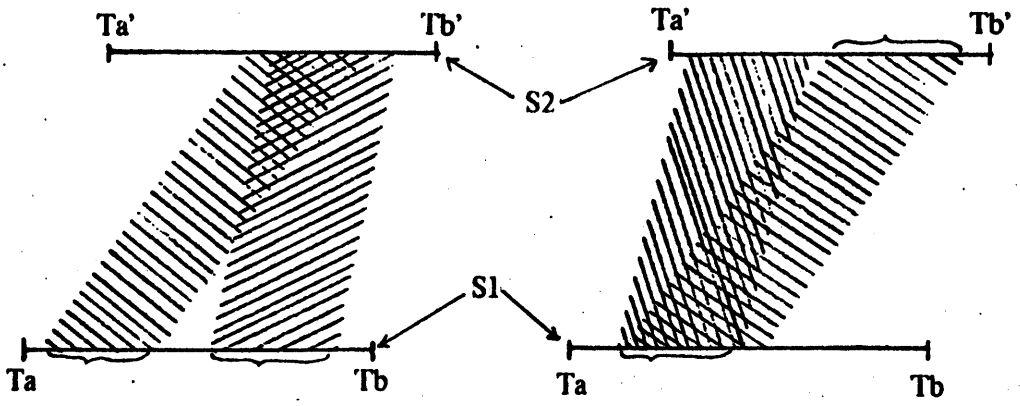
The train-line and the horizontal line can be obtained from vertical lines theoretically, but the vertical line is usually used for a traffic control or a traffic simulation as a short time data.

The representation of trains' movements should depend on how often, how easy, and how fast the data access is done from the procedure. Occasionally, a multiple-linked database may be better than single-linked database.

3-1 Implementation of trial-and-error

For trial-and-error problems, SEARCH-METHODs are well-known, and the characteristics of each of them is clear. However, whether a particular search-method is applicable and how to implement the problem into a search-tree structure depends on the characteristics of the problem itself, such as what state is an alternative, or what movement is a path. The final aim of our impementation is to reduce the number of alternatives and to prune the tree efficiently, for the determination of the timing and the order of each train, at each station, or in each section.

If we let a node in the tree represent the timing of each train, it would not be necessary to explicitly keep the order of trains. However we would have a hopelessly large number of alternatives because time is continuous. On the other hand, if we let a node represent a possible order relation in a section, we have a smaller number of alternatives. However, we must solve the problems of how to represent the timing of each train that has the vagueness and the freedom of timing, as shown in Fig. 2, and the problem of how to compress them.



S1 and S2 : stations.
Ta and Tb : the times between which trains must depart from S1.
Ta' and Tb': the times between which trains must arrive at S2.

Fig. 2 Vagueness and freedom of timing

Suppose we are given an unfocused or illegible picture of a train diagram. Probably we use things like the gradient of each line, the stopping time, the possibility of pass-through, or the departure time of a certain train. But in the case of a new compilation, we have no clues.

Therefore, our scheme will be as follows: in order to approximate unfocused or partial train diagrams, we use the search-tree to obtain the possible order relations accompanied by something like lines stretched in width, and, then, in order to focus them, some constraints. Considering this process, the necessary data link will be both the order relation in each section (horizontal line) and the sequential departure/arrival time of each train (train-line).

3-2 Representation of train movement

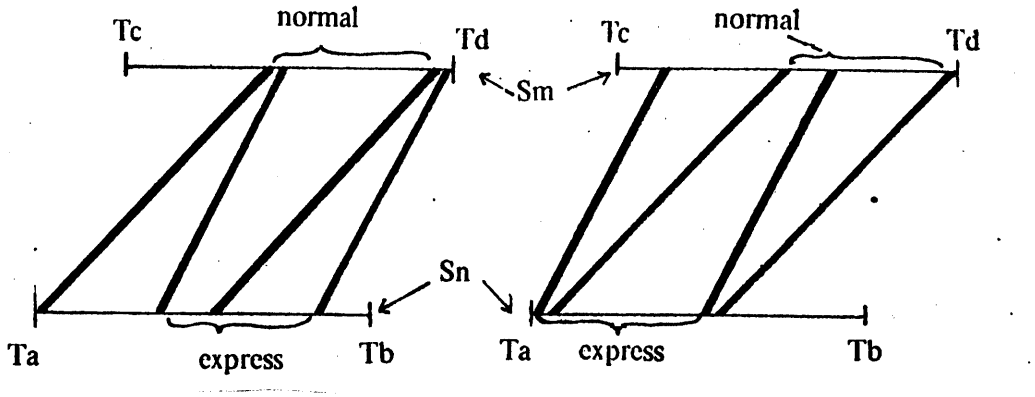
(A train line can be represented as a belt rather than as a line)

Instead of fixing the timing as one instance at each distance, we will let it be some time-length at each distance. By allowing the freedom of timing to remain, we can represent a train-line as a belt in a section, as shown in Fig. 3. The two boundaries of each belt correspond with two train-lines: one is set as early as possible and another is set as late as possible by whatever means. This representation makes it possible to handle a number of alternatives (in terms of timing) as one alternative --- this is analogous to a person's trial-and-error method of setting in the early stage of the compilation --- but leaving the ambiguity of overlapped[4] regions.

---

4. Once one of the train-lines forming the overlapped region is fixed into a line by some means, the degree of overlap decreases one: that train-line must now exit outside of the new overlapped region.

The order of trains which constrains the timing can be represented as a permutation list for each section. A running order in one section corresponds to an arrival order of one station and to a departure order of another station, because no pass-through can happen in a section.



Sn and Sm : stations
Ta and Tb : the times between which trains must depart from Sn
Tc and Td : the times between which trains must arrive at Sm

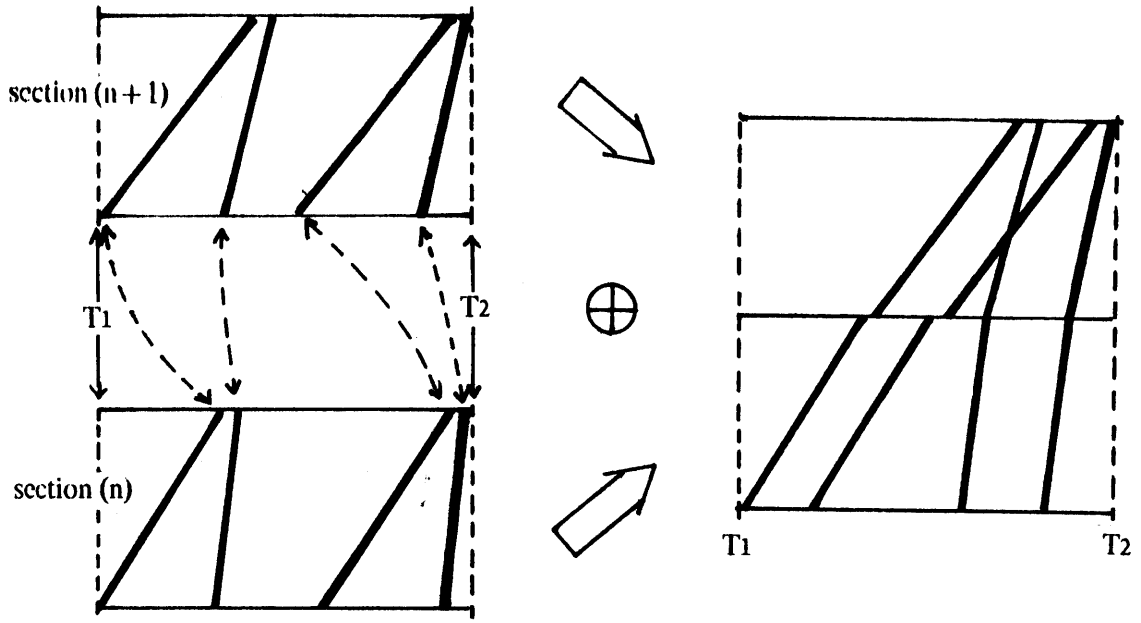
Fig. 3 Train-lines represented as belts.

3-3 Shrinking of train-line (or belt)

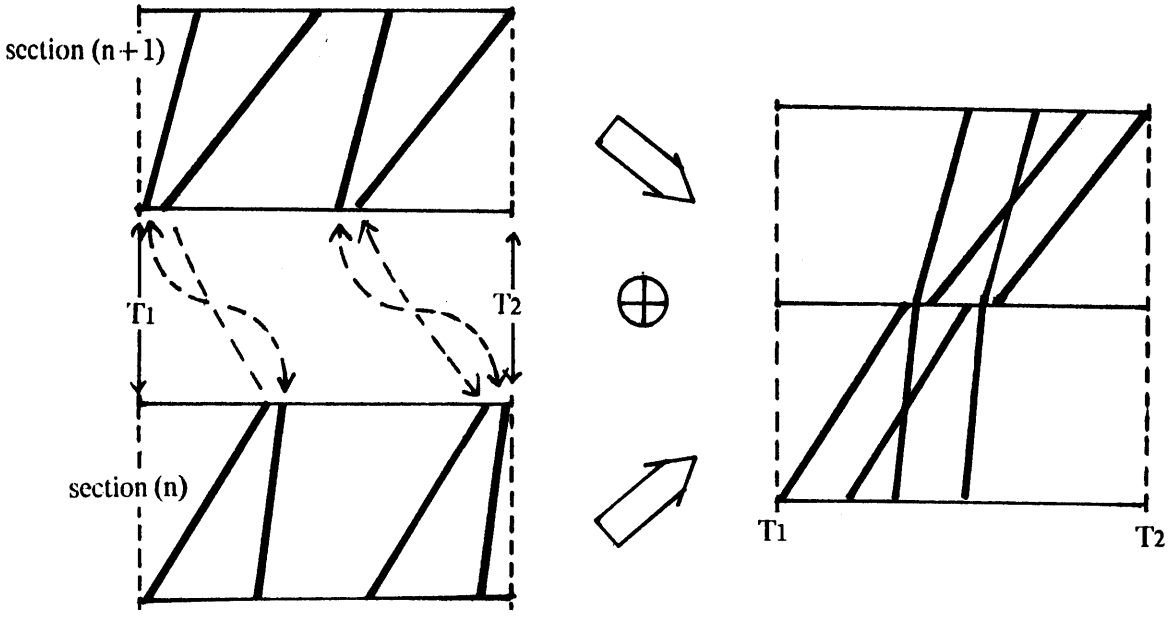
We know that each section is connected sequentially (in distance) and also each leg of the routing of each train is sequential (in time), continuous including its least stopping time. This means that in order to have some order relation in one section within some time region, the timing in the previous (or next) section must be set near some particular time, and vice versa. The series of order relations, obtained from successive sections, has less freedom of time setting than a single order relation, considered by itself. This is analogous to the shape detection algorithm in blocks world, utilizing the propagation of constraints [Waltz].

Therefore, as the number of the previously explored sections increases, the belts, each of which represents the settable region of each train, are shrunk by the propagation of constraints from other sections. Fig. 4 shows a simple example of shrinking of train-lines.

We can also consider this shrinking process as the selective cut-off of some number of alternatives from a larger group of alternatives, which comprises a group of train-lines.



(a) Order relation is unchanged



(b) Order relation is changed

Fig. 4 A simple example of the process of shrinking

IV.The constraints on time

When we compile a timetable, we have at least two kinds of constraints on time: one is physical, such as the maximum velocity of cars or the necessary time for switching; another is qualitative, such as the allowable longest stopping time, the preferable service interval or the preferable terminal departure time. Concerning pass-through, we have some additional restrictions that are mentioned later.

Let us consider a few extremes of compilation for more concrete explanation.

☐ If we set only one train in a time-zone that is wide enough to ignore the existence of other trains, say with a frequency of one train each day, the only physical constraints are the minimum running time (maximum velocity) of each section and the least stopping time at each station. One more necessary constraint is the terminal departure time (or the arrival time of another terminal) in order to fix the train-line.

☐ If we set some number of same class trains in a time-zone wide enough to ignore the conflicts or near-miss between trains and also narrow enough in time for regular service, the necessary constraints are the same physical ones as above, plus the qualitative constraints of terminal departure time of one train and the number of trains (or the interval between them).

☐ If we set two or more different classes of trains in a time-zone that is not wide enough to avoid a pass-through, we need other physical constraints which are the switching times between them at the pass-through station. In addition the capacity for pass-through must be known for each station.

(A) Constraints come from physical conditions
(These may correspond to "requirements" in [NUDGE])

(1) The basic train movements are "Running" and "Stopping", and each constraint on train x is represented as follows.

$$T_r(x, m) \geq T_{ro}(k, m)$$

$$T_s(x, n) \geq T_{so}(k, n)$$

where,
Tr: running time
Tro: shortest running time
Ts: stopping time
Tso: least necessary stopping time
k: class of train x
n: station number
m: section number

(2) The basic operation of the station is the switching among consecutive events (arrivals, departures and passing), and each switching requires a certain time length for actual shifting and redundancy for safety. The constraint between two consecutive events *y1, y2* is represented as follows.

$$T_l(y_1, y_2, n, l) > T_{sw}(e_1, e_2, n, l)$$

where,
Tl: time length between two consecutive events
Tsw: switching time and redundancy
e1, e2: kinds of two events, y1 and y2
l : switch number
n : station number

(B) Constraints come from service quality
(These may correspond to "preferences" in [NUDGE])

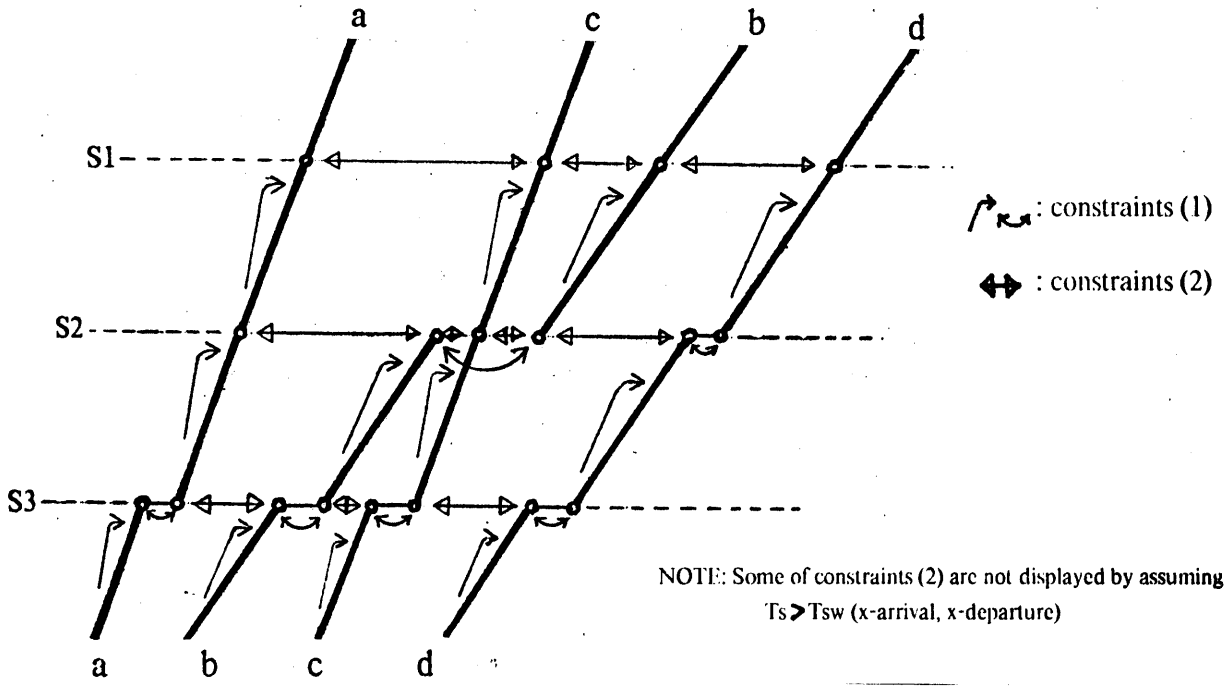
This kind of constraint does not cause collisions or disruptions of operation, but in an actual sense we should properly distribute both unavoidable delay that is caused by high density train scheduling (pass-through) and service itself. One more constraint is the relation to the absolute time.

Some examples are:

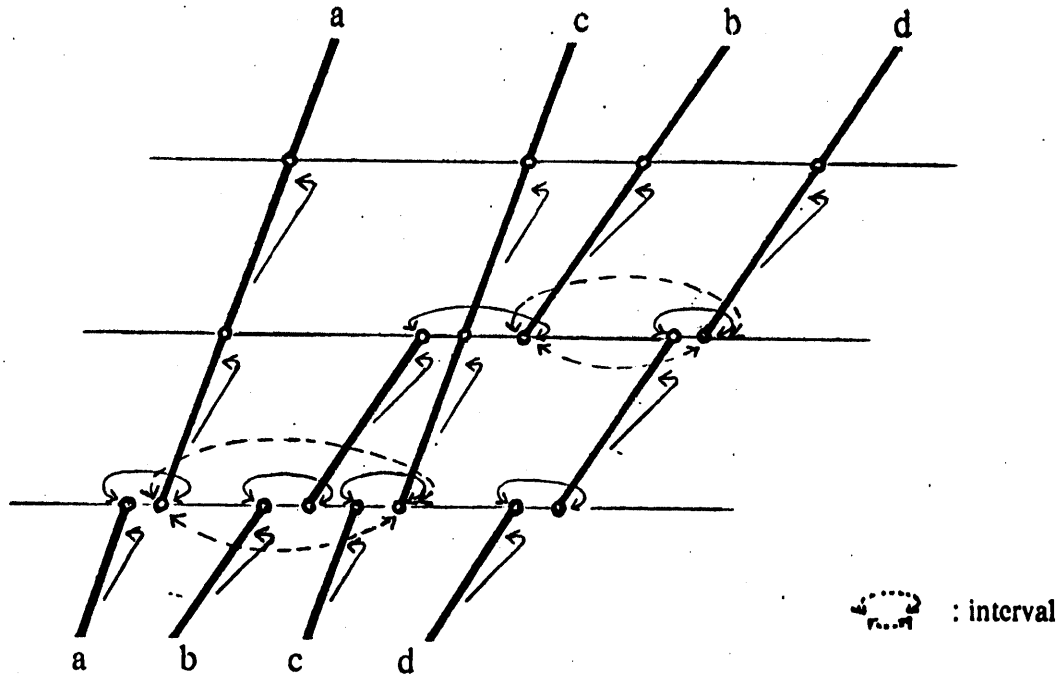
* maximum allowable stopping time
* maximum allowable slow down
* maximum displacement from preferable interval
* time of the first (last) service
* departure time of special trains

Fig. 5 shows the concept of the above constraints. Every constraint must be checked in the current section and at the current station, but some of them are unnecessary in the previously explored sections and at the previously explored stations, and can be reduced to the relations listed in Table 1. These relations can be represented as

```
( ( relation-name   related-train   station-number )
  ( ..... ) ( ..... ) ..... )
```



(a) Constraints from physical condition.



(b) Constraints from service quality.

Fig. 5 Example of constraints.

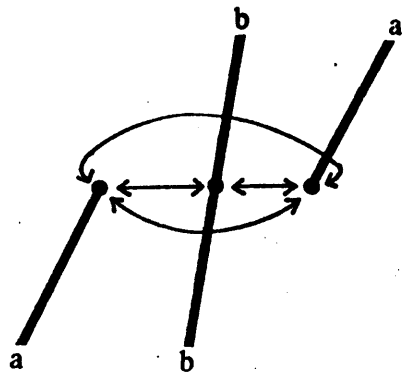
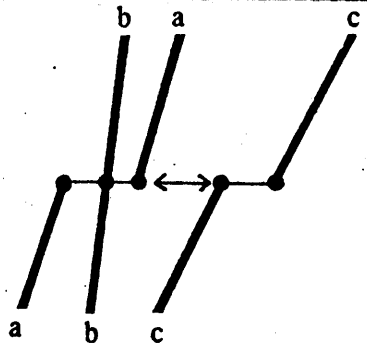
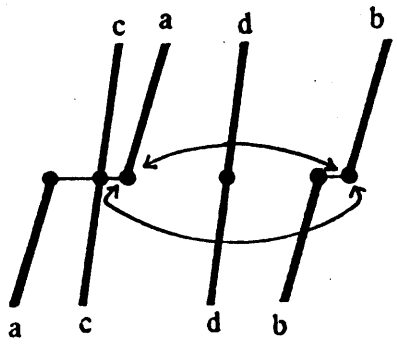
PASS-THROUGH		* Min. Ti (*a-arrival, b-pass*) * Min. Ti (*b-pass, a-departure*) * Min. Ts (*a*) * Max. Ts (*a*)
BEFORE & BEHIND		* Min. Ti (*a-departure, c-arrival*)
INTERVAL		* Max. Ti (*a-departure, b-departure*) * Min. Ti (*a-departure, b-departure*) Ti: time length between two consecutive same class trains.

Table 1. The reduced relations and the constraints that should be checked.

NOTE:

The constraints mentioned in (A) should not be compromised, on the other hand the constraints mentioned in (B) might be compromised under some conditions, but we assume they are given from another system like a knowledge-based system or a human.

V.Constraints for the pass-through and a minor-tree

(How to generate possible departure orders from an arrival order)

In order to change a departing order, an arriving order or a running order, the pass-throughs must be performed at the stations. We have some rules that constrain an arbitrary pass-through as follows:

□ **Constraints on pass-through**

(1) Passing priority derived from train class:
A train can not pass-through a train of the same or higher class.

(2) Stopping or not:
A train can not be passed-through at the station where that train does not stop

(3) Configuration of station (sidings and main track):
A train which has already arrived can depart before any other trains' arrivals, and also a train can arrive, as far as the station capacity allows, before any other trains' departure.

NOTE: A station capacity means not only the number of available sidings but also the length and the existence of platform. Fig. 7. shows some examples of the station configurations.

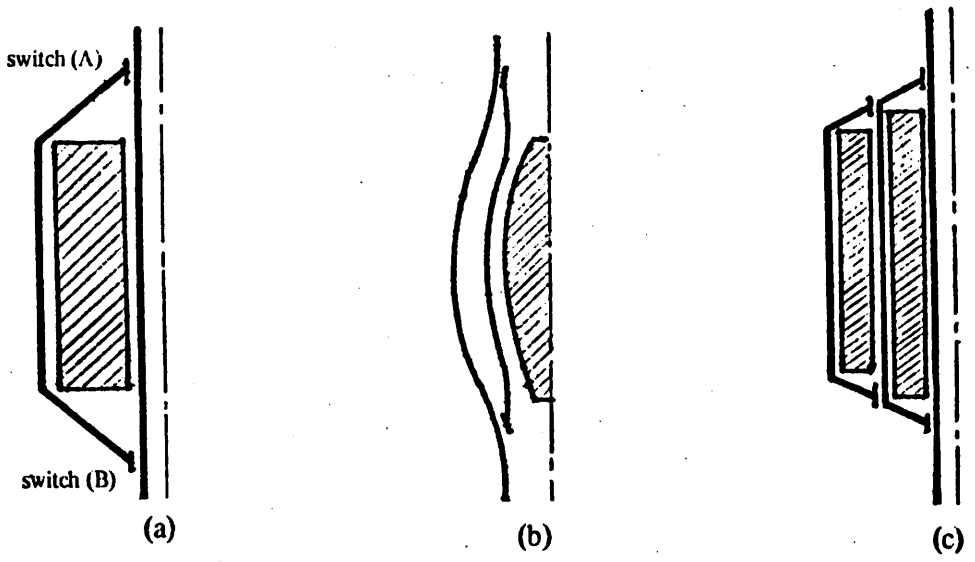
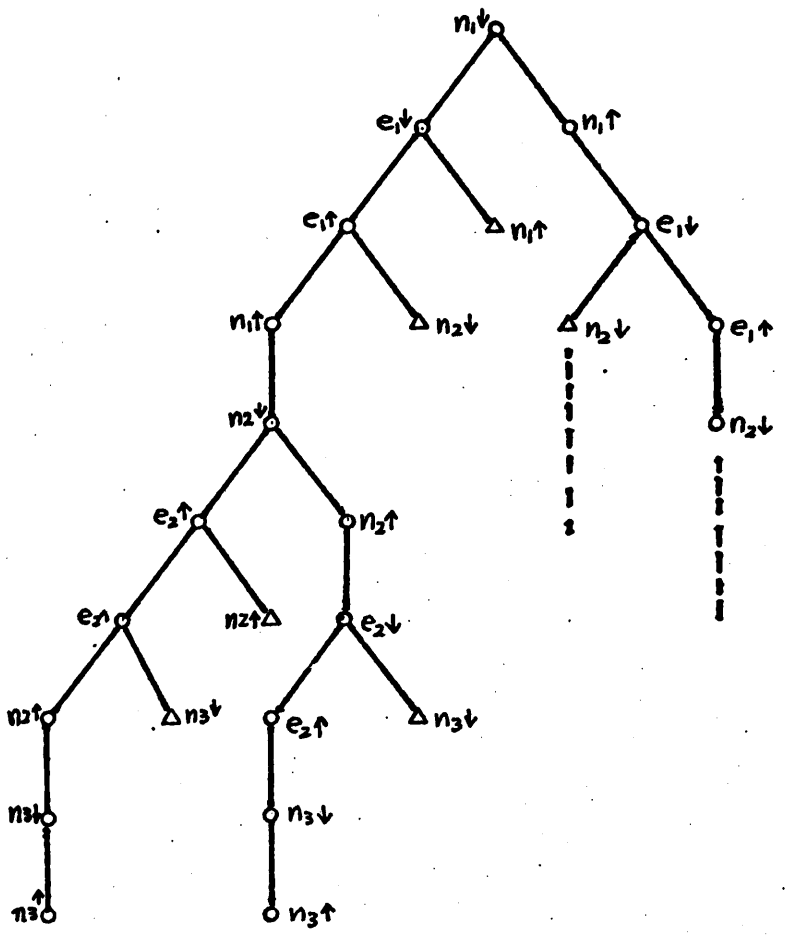


Fig. 7 Example of Configuration of the station

The generation of all possible departure orders from one arrival order is also implemented using the tree structure (which we call "minor-tree") and each terminal node corresponds to an alternative on a major-tree. Fig. 8 shows an example of a minor-tree.



An arriving order is [n1 e1 n2 e2 n3].
The priority of e1, e2 is higher than of n1,n2,n3.
For example, e1, e2 : express
and n1,n2,n3 : normal

The station configuration is Fig. 7 (a).
The switchs (A),(B) move simultaneously for the sake of safety. This means while one train is arriving (departing), another train can not depart (arrive).

: arrival (enter)
: departure (exit)
: not impossible but usually uncommon, so the expansions from this kind of nodes are not shown.

Fig. 8 The structure of Minor-tree

VI.What search method is suitable

In this chapter, we show examples of setting train movements, and using these examples we discuss the search method and the evaluation of various movement settings.

(A) A simple example of setting train movements

We introduce "pattern-diagram" as a simple example of train movements. "Pattern-diagram" is one cycle of the periodically set part of a timetable for a certain time zone, and the creation of this is usually one of the essential stages in the current manual compilation of a timetable.

> NOTE: For the sake of explanation, in this chapter we will assume that a train may stop longer than necessary only when it is passed-through at the station, and can not slow down from its shortest running time, and we will return to consider the slow down of trains. This means that if a train is not passed-through, the boundary moves like glued solid sticks when constraints are propagated, but if it is passed-through, it is discontinuous at the station (except for the one-directional link that constrains the minimum and maximum stopping time).

Fig. 9 shows the process of shrinking at each stage, and Fig. 10 shows one of the final results of shrinking and two stages in process of it, under the condition listed below.

One-way setting in a double tracks line, from one terminal to the other.
Three kinds of trains run on the track which contains seven stations besides two terminals.

On Fig. 9
Limited express runs every T1 (minutes), and stops only at both terminals.
Express runs every T1 (minutes), and stops at both terminals, No.4, 6, and 7 station.
Normal runs every T1 (minutes), and stops at both terminals and every station.

On Fig. 10
Limited express "L" ("L'") runs every T2 (minutes), and stops at only both terminals.
Two expresses "E1" and "E2" run every T2 (minutes), and stop at both terminals, No.4,6, and 7 station.
Two normals "N1" and "N2" run every T2 (minutes), and stop at both terminals and every station.

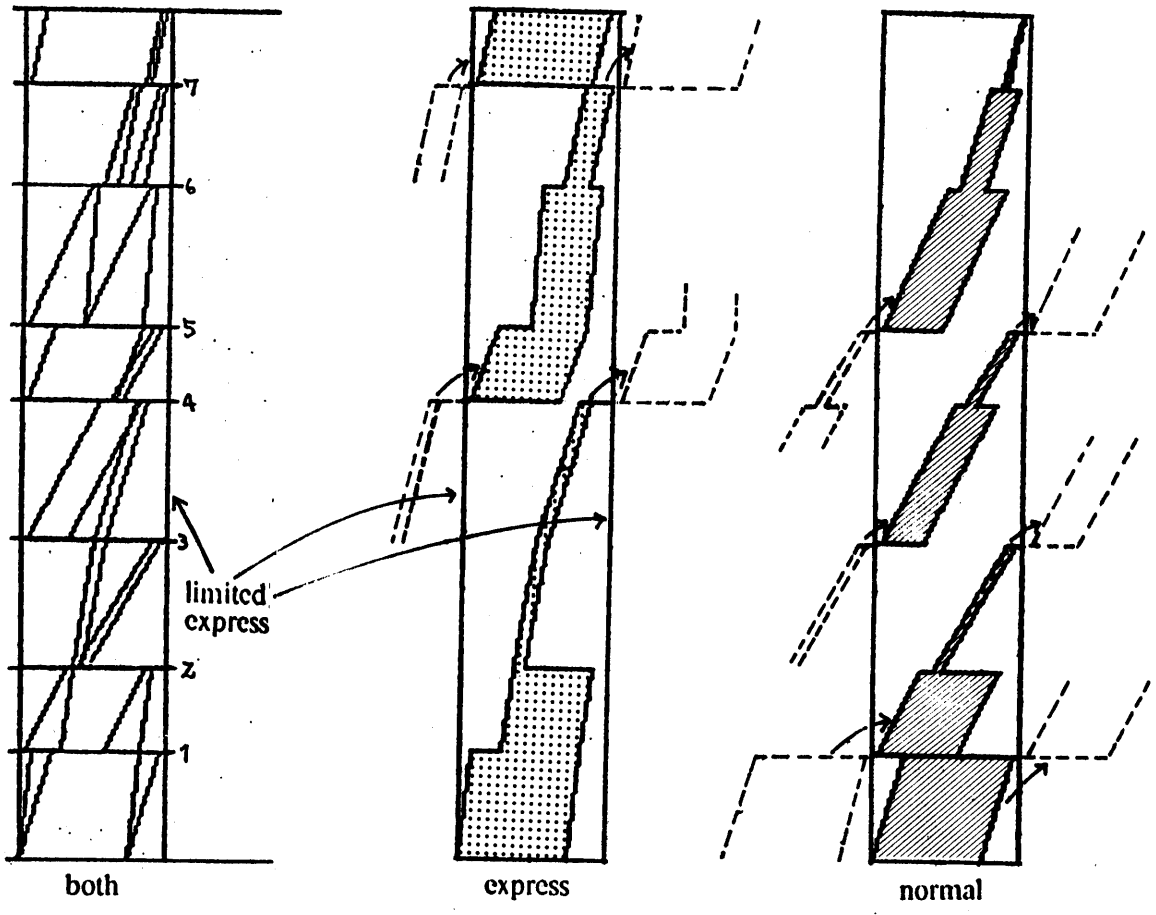


Fig. 9 Shrinking process on each stage

In this example, the limited express is represented as if it runs with the infinite speed, for convenience sake, because it is used as a sort of boundary wall, as usually done. The actual computation is also done in the same way.

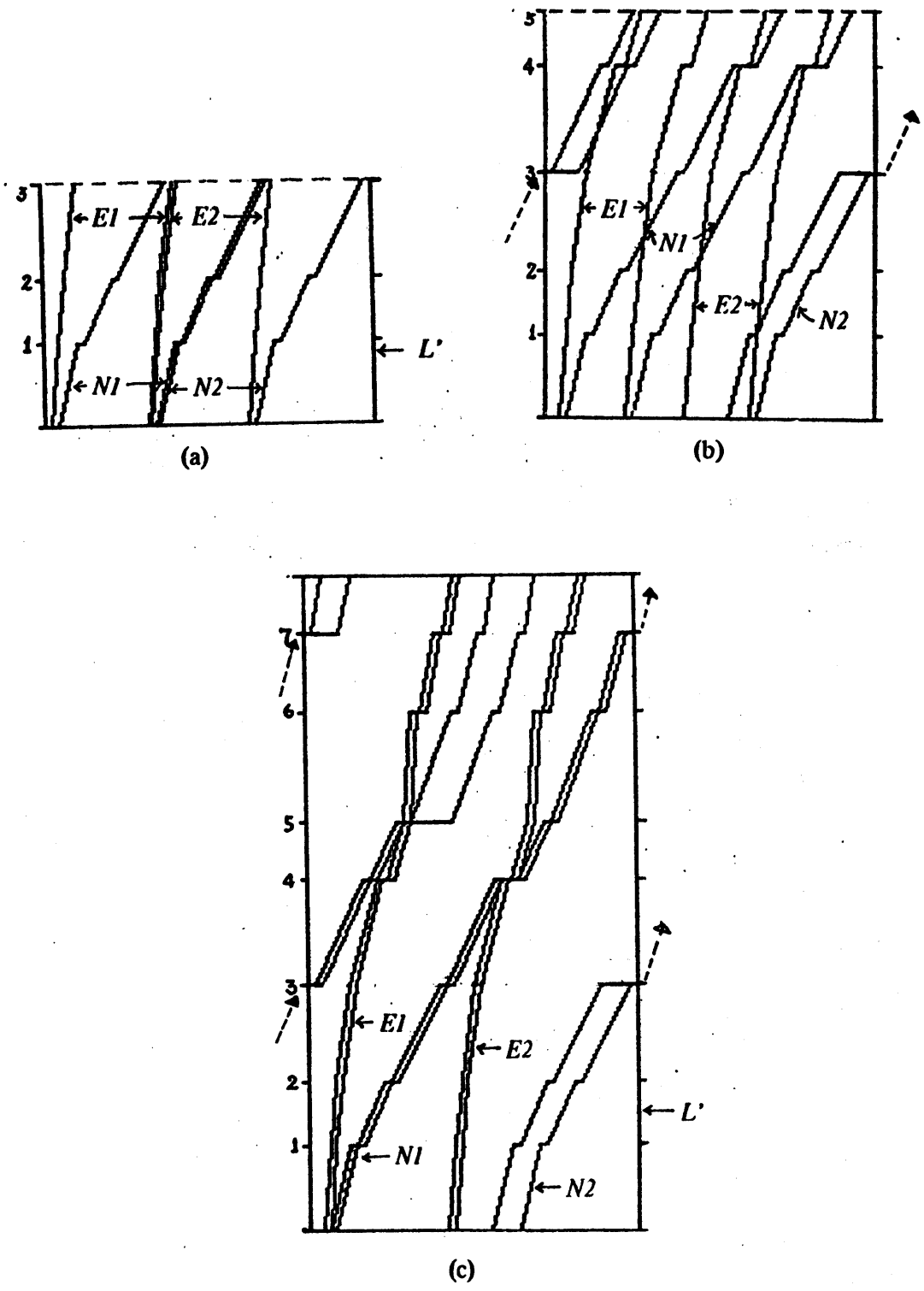


Fig. 10 Final result of shrinking and two stages in process of it

In Fig. 9 we can see that both trains, which depart from the terminal at mostly free timing, begin to shrink section by section. Occasionally they stretch when they are passed-through, and again begin to shrink.

Using Fig. 10, we explain simply how the constraints work.

❑ Fig. 10 (a) : From terminal departure to arrival at No. 3-station, *L*, *E1*, *N1*, *E2*, *N2* and *L'* do not change their order, and so the earliest boundary (hereafter we call simply *earliest*) of *E1* and *N1* do not shift but *earliest* of *E2* shifts right through the constraint relation (link) between both arrivals of *N1*, that shifts right, and *E2* at each station, *earliest* of *N2* also shifts right through the link between both departures of *E2* and *N2* at departure terminal. Concerning the latest boundaries (hereafter we call *latest*), *latest* of *N2* shifts left through the constraint between both arrivals of *L'* and *E2* at each station and then *latest* of *E2* shifts left through the link between both departures of *N2* and *E2* at the terminal and so on so forth.

❑ Fig. 10 (b) : On arrival at No. 5-station, *earliest* of *N2* pushes *earliest* of *E1* right and then it propagates on *earliest* of *E1* downward, to *earliest* of *N1* at the terminal, on *earliest* of *N1* upward till the arrival at No. 5-station, at the same time at No. 4-station, to *earliest* of *E2* through pass-through link, on *earliest* of *E2* upward till the arrival at No. 5-station, also on *earliest* of *E2* downward, to *earliest* of *N2* at the terminal and on *earliest* of *N2* upward till No. 3-station. Of course, if one of the above links is wide enough to absorb the propagating shift, the propagation stops at that link.

❑ Fig. 10 (c) : On departure at No. 5-station, the constraint "maximum stopping time" holds true clearly between the arrival and the departure of *latest* of *N2*.

❑ In every expansion, we check that the *earliests* are more left than their own *latests*, and then generate the alternatives for the next expansion.

(B) How to evaluate

We consider a simple method of assigning a penalty "P" to each possible series of order relation. P can be represented as a linear combination of the unwilling delay of each class train (so far, the stopping time), which is caused by pass-through.

$$P = \Sigma\, W(k) \times |\{T_{ea} - T_{cd}\} - T_n(k)|$$

where

W: weight(depends on the class of the train)
Tn: necessary time to arrive at some station from the departure terminal without the passing-through (also depends on the class of the train)
k : class of train
n : station number
Tea: earliest arrival time at some station
Tld: latest terminal departure time

(C) How to search

We believe "depth-first search" is the best by the reasons listed below.

□ As seen in the example, one expansion of the setting usually changes the departure and arrival time not only of the current station (or section) but also of the previously explored stations so if we had one set of train-lines and tried to expand another alternative, we would need a backward calculation in order to get the previous train-lines that we had already got before.

□ The search tree ("major-tree" in the former chapter) has the structure listed below, so whichever path is expanded, its maximum depth is limited to the number of sections on the line.

1-ply = 1-section (1-station)
1-alternative = 1- order relation in a section.

□ Sometimes, there is something that ceases expanding path in the middle. That is to say, by detecting the over-shrunk(the minus value of the width of the belt), we can know this order relation is infeasible.

□ The evaluation of the penalty in the middle is not useful to order the alternatives because the unwilling delay increases discontinuously at some stations, as shown in Fig. 11, and at this point we have no under estimation method (for A* algorithm).

Of course, once we get the first feasible (not necessarily optimal) series of order relations, we may avoid expansions of some alternatives in the middle by comparing the current P with the least final penalty obtained.

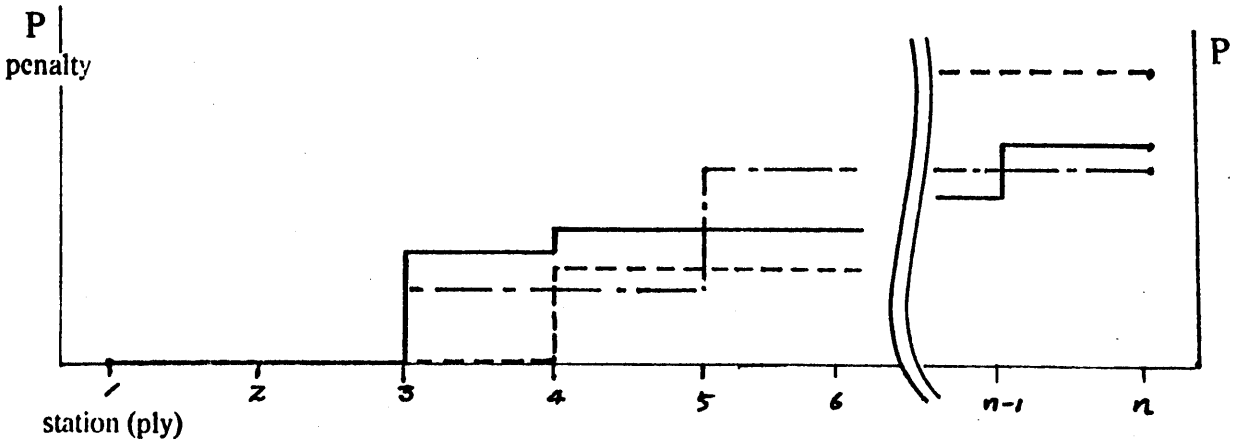


Fig. 11 Increase of penalty

NOTE: Under the condition applied on Fig. 10, we show one example of how constraints reduce the number of possible order relations.
If there are no constraints, the number of possible order relations is 4,586,471,424.
If there are only the constraints on pass-through, the number is 588.
If there are both constraints (on time and on pass-through), the highest number is 76. Of course, this number depends on the period T and the time constraints (B).

VII.Bending of the train-line (Letting speed down)

Hitherto, we have assumed that the stopping time can be expanded to the maximum allowance, but the running time can not be changed from the shortest one.

However, an increase of the running time is equivalent to an increase of the stopping time in terms of unwilling-delay (penalty). This means that, as shown in Fig. 12, the unwilling stopping time of the lower class train could be reduced by changing the pass-through station and slowing down the higher class train (unwilling increase of the running time for this train). The value of new unwilling delay could be small enough to decrease the total penalty from its former value. This method is called "Bending of the train-line" and is used by humans.

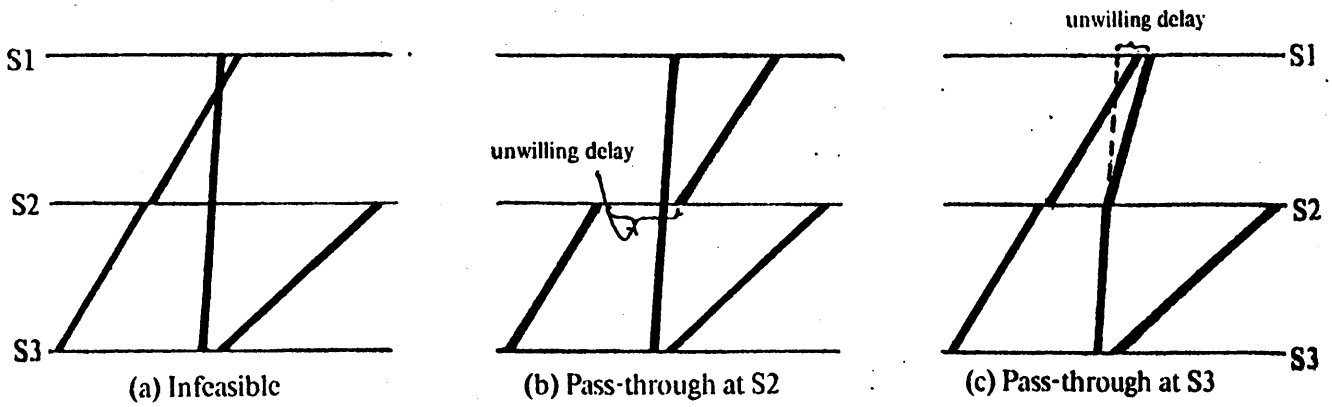


Fig. 12 Basic cocept of bending of the train-line

However this method is very complicated. Suppose there is the setting of the train shown in Fig. 13 (a), then we have at least three choices: bend the *express*, bend the *limited* or bend both. In this example, it is clear that the best choice is the one in Fig. 13 (b) considering the penalty, but the situation is usually more complicated.

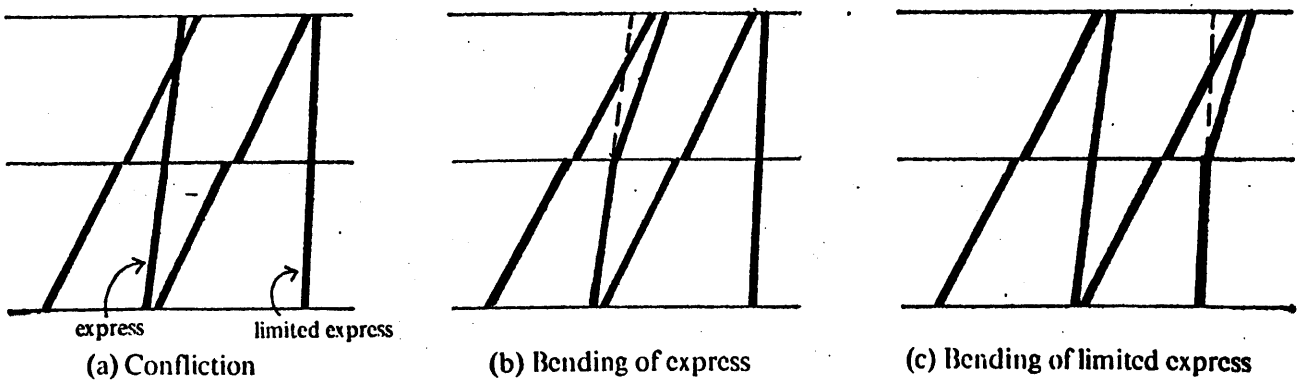


Fig. 13 Choices of bending of the train-line

In bending the train-line, it seems to us that a human uses the backward correction (bending) after detecting the infeasibilty or hopelessness of a setting of train movement with the minimum running time, but we implement this process as follows:

For bending of the train, one more set of train-lines (the *b-line*), which is not used unless the set of train-lines set with the minimum running time (the *a-line*) becomes infeasible, is set simultaneously. The *b-line* is set in a way similar to the *a-line*, but following the rules below.

1. The earliest boundary of *b-line* (*b-earliest*) is set decreasing the shift width as much as the maximum running time constraint allows -- `bending` only when the constraint propagates on it toward the departure terminal.

2. Corresponding to the bending process of *b-earliest*, some legs of the latest boundaries of *b-line* are bent up to the same running time as *b-earliest* on the same section.

3. As a result of this process, the latest boundaries of *b-line* (*b-latest*) are set using the same running time as *b-earliest* in the corresponding section -- `bending`. But this process has the possibility of overshifting of the *b-line* beyond the boundary, so it is set decreasing the shift width as much as the minimum running time constraint allows -- `raising` toward the shrinking side, only when the constraint propagates on it toward the departure terminal.

4. Corresponding with the raising process of *b-latest*, some legs of *b-earliest* are raised up to the same running time as *b-latest* -- `raising` toward the shrinking side, and its shift width propagates.

5.The limited express also has its *b-earliest* and *b-latest*: the former is used for bending and the latter is used for a sort of boundary of its bending.

By this procedure, *b-earliest* is the train-line that is packed as tight as possible toward the earliest side and *b-latest* is the one packed as tight as possible toward the latest side, considering the bending of the train-line. This means that each train line of *b-line* is given the maximum freedom of the setting. **Fig. 14** shows a example of a *b-line*.

When an overshrunk *a-line* is detected, *b-line* is also checked. If *b-line* is not overshrunk, *a-line* is substituted by *b-line*. After having set up to the arrival terminal, the following shrinking procedure is processed because the *b-line* includes the freedom of running time.

1. The *b-latest* of the limited express is shifted up to its *b-earliest*.

2. For each express, in each section from the arrival terminal to the departure terminal, the train-line with the minimum running time is tried from the earliest arrival time: as far as the corresponding departure time or the consecutive arrival time is settable inside of its *b-line*, it is set as its new *earliest*. If not, the time of *b-latest* is selected as its new *earliest*.

Both processes should be accompanied with the propagation of constraints.

This may be one setting of several variations, but at least it is enough for the evaluation of its penalty. Fig. 15 shows the result of process 1 and the result of process 2, applied to Fig. 14.

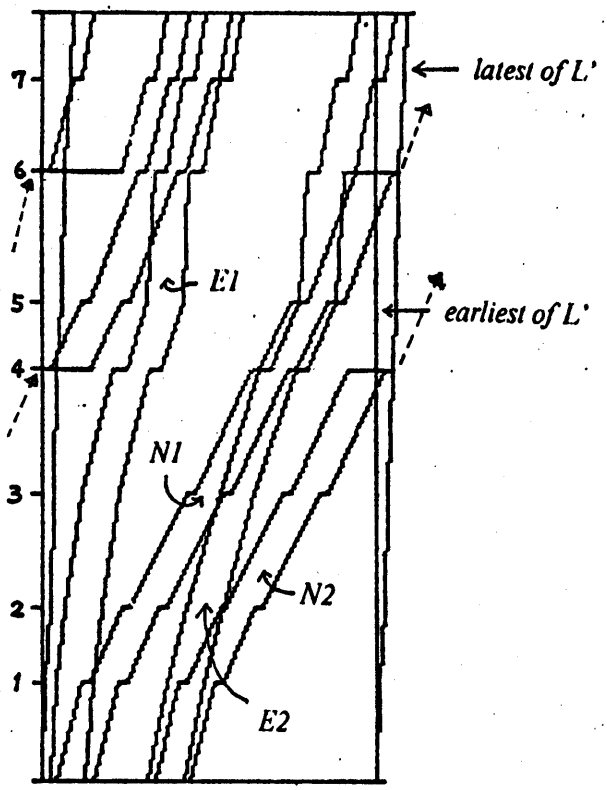
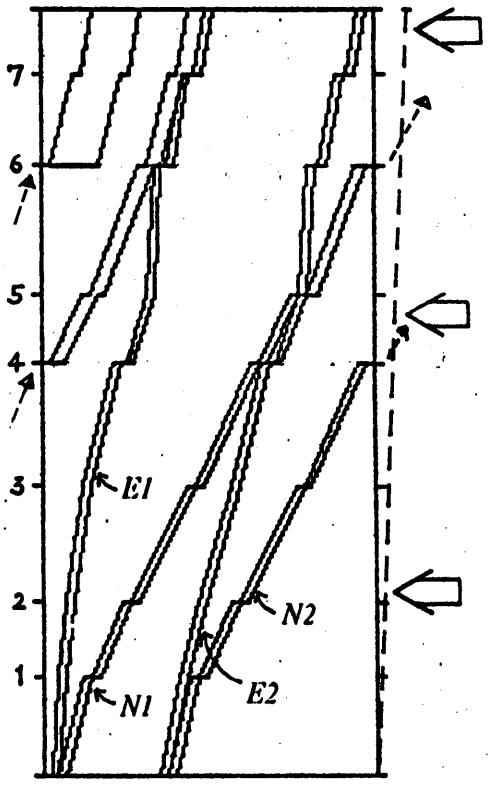
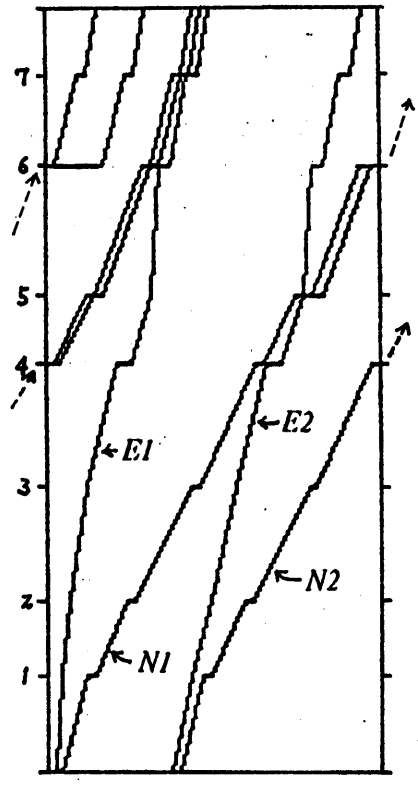


Fig. 14 An example of *b-line*



(a) Result of process 1



(b) Result of process 2

Fig. 15 Raising from *b-line*

VIII.Conclusion

We proposed an application of the range-constricted search algorithm for the compilation of a time-table for a railroad.

- ☐ The concept of the settable time region is used to represent a number of the possiblities of time setting.

- ☐ The propagation of constraints is used to cut off the impossible settings selectively.

The propagation of constraints is well-known as a powerful concept in vision, where some knowledge of physically realizable relations is used to constrain so-called neighbors. Similarly, in the problem of setting of train movement, we can find some properties -- the continuity of train movement, the difference of running time and so forth -- and then, an application of propagation of constraints shows that clarification of the relationship between trains and stations makes it possible to reduce the ambiguity or freedom of the train movement. On the other hand, the concept of settable time region makes it possible to handle a train that has ambiguity or freedom.

We did not discuss "final shrinking", the process for literally shrinking to a train-line. One minimum penalty answer for the final shrinking is found by setting the terminal departure time to the latest time, in the order of normal and express, having no guarantee of the so-called best setting. This will be our future work. However, the preliminary shrinking has the effect of reducing the human work even if the final shrinking is done by a human.

Finally, we mention that this scheme may be applied for setting of train movement on a single track line.

IX.Acknowledgement

I would like to thank Prof. Patrick H. Winston for providing invaluable suggestions and assistance during my visit, and Dr. Robin B. Stanton for his encouragement. Discussions with Dr. R. Bruce Roberts were helpful. Many thanks go to Matthew T. Mason for helping to improve the English of this manuscript and for his good suggestions. Many other members of the Artificial Intelligence Laboratory, too numerous to mention, also provided useful suggestions along the way.

This research could not be completed so rapidly without LISP and TV terminal of the AI Laboratory.

Bibliography

Goldstein, I. P. 1975 "Bargaining Between Goals", IJCAI IV. Tiblisi, U.S.S.R.

Goldstein, I. P. & Robert, R. B. 1977 "NUDGE" A KNOWLEDGE-BASED SCHEDULING PROGRAM, MIT AI Memo 405

Waltz, D. 1975 "Undestanding Line Drawings of Scenes with Shadows", in P. H. Winston (Ed.), The Psychology of Computer Vision, NY: McGraw-Hill