UNIVERSITY OF PENNSYLVANIA
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

COMPUTER SIMULATION IN REAL TIME RAIL TRAFFIC CONTROL

Edwin Reese Kraft

Philadelphia, Pennsylvania

December 1983

A thesis presented to the Faculty of Engineering and Applied Science of the University of Pennsylvania in partial fulfillment of the requirements for the degree of Master of Science in Engineering for graduate work in Civil Engineering.

Edward K. Morlok

Frederic Roll

ABSTRACT

Computer algorithms for assisting railroad train dispatchers to find minimum delay dispatching strategies on single track rail lines are developed. Train point to point running time prediction methods are evaluated and the variability in actual running times is measured from data. Uncertainty in train running times expressed as a percentage of train running times is explicitly considered in the planning algorithm. FORTRAN computer code implementing the algorithm is presented and documented in the Appendix.

ACKNOWLEDGEMENTS

I wish to acknowledge the patient assistance of Dr. E. K. Morlok, my thesis advisor, during the preparation of this document.

Also, Mr. W. M. Hart, F. L. Wiegmann, L. C. Davis and L. R. Townsend at Chessie each reviewed various parts of the document during several stages of its development and provided valuable criticisms.

I enjoyed the opportunity to meet with Mr. Chris Hausler at General Railway Signal and tour their facility in Rochester, New York. Bob Lusby at the Seaboard System Railroad arranged my visit to their Corbin, KY system installed by WABCO. Stu Burgess and his staff in Richmond, Virginia have been very helpful during my several visits to their system installed by S. A. B. Harmon. Transcontrol does not yet have a major Centralized Train Dispatching installation in this country, but Mr. Robert Jahn provided me with information on European installations installed by Transcontrol's parent company, Siemens.

CONTENTS

ILLUSTRATIONS

1. Introduction

1.1. Statement of Purpose and Outline

Computerized dispatcher assist features in centralized traffic control centers can improve the efficiency of rail line operations. Dispatcher assist systems improve capital productivity by increasing rail line capacity and reducing train delays to a minimum. By making better use of existing facilities, investments in additional facilities can be reduced or eliminated.

Several commercial train dispatching computer systems have already been produced. The problems of train tracking, maintenance of records, and exercise of control over switches and signals have been solved. The emphasis here is on train meet planning on heavily used, single track main lines. Decision making rules will be derived and dispatcher assist computer code developed. The performance of alternative train meet planning algorithms will be evaluated.

This thesis is divided into five sections, plus summary and appendices. Each section is described in the following paragraphs.

Introduction

The introduction includes the statement of purpose and outline of the thesis. It also includes sections providing historical perspective, a

discussion of the problems of past installations of computer assist systems, and a set of evaluation criteria or goals for design of dispatcher assist systems.

Prediction of Train Performance

Prediction of train performance means prediction of the time trains will require to run from point to point, exclusive of any delays. Of interest are both the mean and variance in train running times. The mean running time is the optimal predictor for train performance in the sense that it minimizes the mean square prediction error. The variance is also important, since it introduces uncertainty into the planning process.

Train performance predictions, along with the present system state, are the inputs to the meet planning procedures. The quality of the output meet/pass plan depends upon the quality of these basic inputs. Therefore, it is worthwhile to devote effort to the development of accurate models for performance prediction.

Local Optimization

The "first come, first served" dispatching rule is the most common procedure now used in computer programs for simulation of train dispatching. This rule states that the first train to arrive at the

entrance to a single track section will be dispatched across the section first. When a train arrives at the entrance to single track, if that track is not occupied, then the arriving train is cleared onto the single track.

The first come, first served rule resolves each conflict with minimum delay, considering only the two trains involved in the conflict. However, this procedure does not guarantee that delay is minimized "globally". In fact it might be better to resolve a particular conflict in a locally suboptimal way, if greater delay can be saved in a subsequent train meet. The advantage of this procedure is its simplicity.

Since the first come, first served rule does minimize local delay, it is an optimization procedure. Because of these optimizing qualities, it can be useful in normative (real time control) as well as in predictive (simulation) applications. The rule can be called local optimization.

Only simple modifications are required to account for differences in train priorities, running times, and for differences in the time penalties for using various possible meet/pass locations. The formulation of these required modifications can be derived from probability theory.

Meet Planning With Uncertain Run Times

Explicit consideration of the uncertainty in train running times adds complexity to the train dispatching problem. Because of run time prediction errors, the actual amount of delay resulting from a given sequence of decisions cannot be determined in advance. Only the distribution of delay can be estimated. The practical limitations of meet planning will be examined.

Global Optimization

A procedure for systematically exploring all possible meet/pass strategies is described. This procedure is based on branch-and-bound integer programming techniques. The minimum delay strategy for resolving all pending conflicts on a single track rail line is found.

1.2. Historical Perspective

A centralized traffic control system enables remote control of all switches and signals on a rail line from a single location, often hundreds of miles away. Trains move on signal indication, eliminating the need for train orders. The dispatcher can follow the progress of trains along the line by observing lighted indicator lamps or computer graphics displays. Centralized traffic control was developed by the General Railway Signal Company. The first system was

installed in 1927 on the New York Central, between Stanley and Berwick Ohio, a distance of 40 miles[1].

While total track and route milage has been on the decline, the number of miles under central control has been steadily increasing. In 1973, railroads in the U.S. had Traffic Control Systems (TCS) on 39,830 route miles and 47,060 track miles. By 1982, they had TCS on 44,821 route miles and 53,594 track miles. At the same time, they reduced track mileage from 368,625 in 1973 to 290,000 in 1980[2]. As railroads consolidate more traffic on fewer routes, there will be a continued increase in the number of miles under direct central control.

The same ten year period saw the first introduction of computer assisted traffic control systems. Prior to 1966, TCS systems used only electromechanical relays in their logic and coding circuits. During 1966, Westinghouse Air Brake Company (WABCO) built a computerized interlocking control system for the Union Railroad in Pittsburgh, Pa. In 1971 WABCO installed a computer system for the Canadian National in Kamloops, British Columbia controlling 500 miles of railroad. In 1973, Harmon Electronics installed an 850 mile control system for the Denver & Rio Grande. General Railway Signal also supplies computer-assisted TCS systems[3].

Therefore, while the total mileage under TCS has increased during the past ten years, the growth of computer aided dispatching systems has been even more dramatic, starting from practically zero only ten years ago. Computers are being installed to upgrade old TCS systems, as well as on new milage. Indeed, some of the old computer systems are now being replaced with newer ones.

Computer systems have generally been custom designed for each location. Depending on when the system was installed and the philosophy of the railroad management, the systems have had widely varying capabilities. Some installations boast complete computer control, where the computer plans train meets, exercises control over the field switches and signals, and maintains the trainsheet, the record of all train movements. Other systems have less capability.

More recent systems generally have had less capability than the older ones. WABCO's initial installations provided full automatic features; now WABCO has developed the "Trak Commander", a microprocessor based system which offers neither automatic dispatching nor automatic record-keeping. It seems many of the features of the previous sophisticated systems either have not performed

satisfactorily, or are impractical to use.

1.3. Problems of Dispatcher Assist Systems

One common problem has been an overworked central computer leading to unacceptably slow response time. Some systems require several minutes simply to clear a train out of a siding. On the L&N's Corbin system, it has been faster to maintain manual records of the operation rather than key the information into the computer[4].

It is clear that computer systems have not been used in the way their designers intended. Dispatchers use tricks to manipulate the computers. For example, in Richmond Virginia, Chessie dispatchers control the autoclear function by using "track blocks" intended to provide safety for maintenance of way personnel. Programs intended to perform these functions have been ignored [5].

In spite of these problems, computer assist systems will continue to be installed. The problems of record maintenance, keeping track of train location (tracking), and exercising physical control over switches and signals have been solved. With the declining cost of computer hardware, the problem of inadequate computing power should no longer constrain system capabilities. In the past, many railroads ordered systems having meet planning abilities. There

seems to be no fundamental lack of demand or need for such systems. Rather the disappointing performance of the systems has forced a retrenchment.

The relatively poor performance of the real-time meet planning procedures is not surprising, because the meet planning problem is not trivial. During the past ten years, numerous rail line simulation models have been developed. These simulation models have been conceived and executed by operations research specialists. The use of sophisticated logic to simulate rail line operations, running under the close supervision of highly skilled transportation analysts, has been inadequate. It has proven impossible to obtain satisfactory "hands off" simulation of realistic situations including train work, mechanical failures, and maintenance of way track outs. Success has been restricted to highly simplified dispatching situations.

Yet, successful simulation is possible if an effective means of manually interacting with the model are provided. On this future system, the computer would propose a solution, the analyst would review the meet pattern, then modify it, until the pattern is finally satisfactory. This approach is attractive only if:

- the computer's decisions are accurate a high

proportion of the time, otherwise the analyst ends up doing all the work himself , and

- the means provided to interact with the program are convenient and effective.

Dispatcher assist systems have tended to suffer deficiencies in both areas. First, the decisions made with highly simplified algorithms have been incorrect an unacceptable percentage of the time. Second, the procedures allowing communication between the computer and the dispatcher are poor: Often the dispatcher must guess what the computer will do; there is no capability to have a "discussion" in advance. Once the dispatcher knows the computer solution, it may be too late to change it. Communications between the dispatcher and the computer are equally awkward. Rather than waiting for the computer to err, dispatchers:

a) turn automatic dispatching off, or

b) bypass automatic dispatching by entering commands to activate specific switches and signals.

Overriding occurs because dispatchers have learned it is easier to perform actions directly, under manual control, than to allow the computer to perform the actions. This is understandable since present meet "prediction" systems really do not provide any additional information to the dispatcher:

he knows where the next meet is likely to occur without the computer having to tell him. Use of the meet prediction programs would only cause him extra work.

1.4. Goals for Dispatcher Assist Systems

Train dispatching computer systems should be designed with at least the following in mind:

1. Is it economical? The programs should be designed within the capabilities of a reasonably priced and generally available computer system. The system must be priced below the benefits the system will generate in use.

2. Is it safe? Although fail-safe field interlockings will theoretically prevent collisions resulting from computer error, a system must not tempt fate by attempting to issue unsafe directives.

3. Does it function as intended? The system must of course be free of programming bugs. The programs should be well designed. A poorly designed system will never function well, even if its implementation is perfect. The system should be designed so that in the event of total computer failure, the railroad can still be operated.

4. Is it accurate? If the system is designed to make meet decisions or assist the dispatcher, two more criterion apply. First, a high proportion of the

decisions the system makes or proposes must be "correct." A correct decision is defined as a decision that is satisfactory to the dispatcher. If the decisions are not correct, the decision making logic will simply be turned off and the whole effort is wasted. Second, even under worst case circumstances, the system should not "lock up" the railroad. A lock up occurs when two or more trains are facing each other on single track, with no moves possible until one or more trains backs out of the way. If the logic is not sophisticated enough to prevent a lock up, the potential must at least be recognized promptly, and the dispatcher alerted in time to prevent it.

5. Is it generalized? Preferably the programming should be designed for application to any rail line, without special coding for any particular location. Such a program would be simpler to install and maintain, as it would only be necessary to understand and support a single well-documented program, versus having to maintain a multiplicity of special purpose programs unique to each installation. A generalized program enables standardization.

A generalized program would most likely be consistent with the design of modern train dispatching computer simulation models. These models provide an

ordered and efficient way to analyze data. They provide a basis for systematically operating a rail line.

6. Is it in an upper level language? Assembly language programs are extremely difficult to understand or change. Further, they tend to be operable only on specific computers. Software should be as hardware independant as possible.

7. Is it modular? Effective use of subroutines is a good practice. A modular system can be installed one section at a time and upgraded later. Modular design tends to isolate any programming bugs which might exist. At least five levels of implementation are possible:

a. Routines to throw specific switches and clear specific signals (or perhaps combinations of switches and signals through an interlocking) on command.

b. Meet/Pass commands entered by a dispatcher prior to a conflict. The system executes these commands, but has no decision making capability.

c. Autoclear. The system automatically clears certain paths when a train approaches, provided they are not physically occupied.

d. Local optimization. Includes train priorities and prevention of "lock up" conditions. A

single alternative, based on the "first come, first served" dispatching rule is generated by the system and displayed.

e. Global optimization. Multiple meet alternatives are automatically generated and internally compared, and the best alternatives are displayed. The dispatcher has a choice of any alternative or he can modify any alternative as desired.

8. Is it modifiable? Railways need the flexibility to adapt to changing business conditions. If they are unable to change except at great expense they are placed at a competitive disadvantage. The computer system should be adaptable to change in at least the following areas: operating plan, operating rules, track layout, signal system layout, and government regulations.

9. Is it adaptable to advances? The system should anticipate technological or institutional change. For example, it should be capable of accepting input from wayside wheel detectors as well as from track circuits. It should be capable of operation of wayside signals as a replacement for field relay installations.

A successful dispatcher assist system must present possible solutions well in advance, and allow

ready interaction, to permit the plans to be changed. It should be capable of simulating far enough in advance that it will help the dispatcher anticipate congestion prior to its occurrance.

1.5. Factors Affecting the Choice of Meet Location

The train dispatching problem can be viewed in the classic mathematical programming framework as a constrained optimization problem. Train delay is to be minimized, subject to the following constraints:

1. All trains meet their schedules.
2. No crew exceeds 12 hours of service.
3. All on-line industries are served.

A feasible solution does not exist for every dispatching problem. For example, it may be impossible for a local switcher to serve all industries and still make it back to the yard within the 12-hours of service limitation. Or, the dispatcher may be faced with a choice between delaying a merchandise train, causing it to fall behind schedule, or delaying a coal train, causing it to exceed its 12 hours of service limitation.

Rail yard capacity limitations change the optimal train meet decision, because one component of train delay is time loss waiting to be received at a destination rail yard. If two or more trains follow each other with insufficient spacing, a destination

yard may not be able to receive all the trains as they arrive. Dispatchers, aware of the capacity limitations of yard facilities, try to space train arrivals into destination yards.

Some rail lines act as "conveyor belts" between two rail yards: all trains originate and terminate at the same yards. These lines are strongly impacted by destination yard capacity. Traffic flow must be uniform with little train fleeting on such lines. Management tries to dispatch trains at uniform intervals, and the dispatcher should choose a meet strategy which breaks up any fleets which may develop.

Other lines handle trains originating and terminating at many different yards. These lines are less impacted by destination yard capacity, but may experience higher peak loadings due to less coordinated train origination times. Fleets of three or more trains may be developed on such lines.

Train dispatchers must make at least three types of decisions:

1. When the constraints are nonbinding, the dispatching problem is a delay minimization problem. Some trains have a higher value of time than others. The problem may be characterized as the minimization of the weighted sum of train delays.

2. When the constraints are binding, but a

feasible solution exists, the dispatcher is not seeking a strictly minimum delay solution. He also considers external factors including published train schedules. Passenger train priorities are best explained and modeled in this way.

3. When no solution exists which meets all the constraints, the choice is a fundamental management decision which cannot be made by a computer.

It is common to confuse the impacts of constraints with those of priorities. Priorities, as defined here, arise out of the different values of time of various trains. A longer train, or one with more locomotives, should receive a heavier weighting. Weightings may also appear if the traffic carried by one train is more time sensitive than that carried by another. These weightings apply if neither train is scheduled or constrained in any way, or if both trains are expected to arrive ahead of schedule.

Constraints can be modeled as priorities, but this approach is strictly heuristic and not always accurate. It is better to solve the constrained optimization problem directly, excluding any alternatives which do not meet all the constraints. Simple procedures such as local optimization do not include constraints. The branching algorithm includes constraints and can solve the generalized problem

directly. The branching algorithm can also anticipate delay incurred while trains are being held out of destination yards.

Certain types of decisions are appropriate for automation. A computer system may now be capable of making most Type #1 decisions and a few Type #2 decisions. The computer is excellent at unconstrained local delay minimization. The branching algorithm enables global delay minimization, and adequate handling of schedule constraints.

Work item handling has not traditionally been considered appropriate for automation. An error in a work item decision may cause massive congestion. Many crews which perform work will approach 12 hours of service, so the decision often involves this constraint. To solve this problem, the computer may need to project 12 hours in advance, far beyond the practical planning horizon. Automation of work decisions is not recommended; manual action should be required. However, for the dispatcher's information, the capability to simulate work items may be useful.

2. Prediction of Train Performance

2.1. Run Time Prediction Methods

The first step in planning train meets is the prediction of point-to-point train running times. Either a simulation or a statistical approach can be used. If adequate, accurate data are available, the statistical approach is preferred.

The time a train requires to traverse any given segment of track is a function of many variables. These include the train's weight, locomotives, length, engineer, and the track's curvature, gradient, speed limit and physical condition. While many such variables exist, if a train's performance is observed as it traverses several segments, the aggregate impact of these variables may be summarized by the train's historical performance record.

Existing dispatcher assist systems utilize a "running time/speed table" to predict performance. This table contains standard run times or speeds over each section of the line. It may have separate entries for various train classes and for each direction. The values contained in the table may be updated to account for normal seasonal changes in train performance.

The primary advantage of the speed table is its simplicity. It is not necessary to store or process

past performance data. However, a more sophisticated predictor which utilizes historical information is more accurate.

Assume a train has traversed segments 1 through n, but has yet to cross segments n+1 through m. Then the run times on each segment n+1 through m may be estimated as a function of the observed run times of segments 1 through n. The functional form may be nonlinear and may include all previously observed run times. For the purposes of this study, the simple linear functional form with only one independent variable has been assumed.

2.2. Regression Predictor

Systematic differences between trains tend to lead to high correlations among run times over various segments. For example, if a train has only one working locomotive, its run time should be consistently slower than another train having two locomotives.

In Figure 1, a train has arrived at B and its run time crossing segment #1 has been measured. Using this run time, the run times on segments 2 and 3 can be estimated. (In actual practice, the run time A-B might be checked against a reasonable maximum. If it exceeds this maximum, the train has presumably stopped enroute, so the observation would not be used.)

Segment 2 is "adjacent" to segment 1; it is the first segment downstream. Segment 3 is the second segment downstream from segment 1.

A B C D

Observed Predicted Predicted

Seg #1 Seg #2 Seg #3

Fig. 1. Observed and Predicted Run Times

In general, regression equations can predict run time T_p on any downstream segment given the measured run time T_o on any observed segment. The form of the predictor is:

$$T_p = a + bT_o \qquad (1)$$

where the parameters a and b have been estimated from data.

The first segment of the line will have an equation for every other segment of the line. The final segment, of course, needs no equations since no segments are "downstream" from it.

An analysis of data from four Chessie rail lines is performed here. While it is not claimed the results are necessarily typical of all rail lines, the

techniques used to analyze the data may be applied to any rail line.

2.3. Data Source and Analysis

The source of the data was the Chessie centralized traffic control center in Richmond, Va. Since the data used in this study was computer generated, the data base is believed to be very accurate.

First, time distance charts or stringlines of the operation were drawn. This gave a visual picture of the operations. By inspection, delays and stops to perform switching were identified and removed from the data base. The final result was a data base of free, unimpeded point to point running times.

Using this data base, a set of regressions were performed. The correlation between run times of trains on various segments was measured, and regression coefficients and root mean square errors were estimated.

2.4. Results

The estimated mean run times are reasonable: they are within the speed limits and performance capabilities of the trains. The variance in run times is large. Among apparently identical trains, the standard deviation is 20% to 35% of the mean run time.

Train performance is not only highly variable,

but it is also inconsistent. Trains speed up or slow down unpredictably. Downstream run times are only weakly correlated with previously observed times. While the regression results are statistically significant, the use of the regression models does not result in a dramatic reduction in root mean square error. Regression run time root mean square error is the conditional standard deviation in running times, S, the amount of "spread" in the data from the regression line, while the speed table error is simply the unconditional standard deviation. The average improvement in error is only 12.4% when the two segments are adjacent. The average R value for adjacent segments is 0.41.

As the number of segments downstream increases, the correlation in running times decreases. For the second segment downstream, the R value declines to 0.29; for the third segment, R is 0.26. The improvement in error resulting from use of the regression model declines to 6.6% and 5.9%, respectively.

The physical limitations of loaded trains results in a measurably greater correlation of run times between segments, and an improved ability to predict train performance. For loaded eastbound trains, R is .44, while for empty westbounds, R is .38.

While these results are statistically significant, they suggest that only small improvements in root mean square error can be achieved through use of the regression models. If the results of this analysis are typical, the reduction in error from use of a regression model may not justify the added effort and expense. This work justifies continued use of the speed table method (as described on page 18), and provides an estimate of the root mean square error.

For the remainder of this research, the speed table approach will be used.

3. Local Optimization

3.1. The First Come, First Served Rule

One way to reach a meet decision is through use of the first come, first served rule. This rule is useful because, at least locally, it minimizes delay. Also, it is simple to implement. The rule states that the first train to arrive at the entrance to a single track segment will be dispatched across the section first. In fact, the dispatcher cannot wait until the trains arrive at the entrance to single track. He must make his decision in advance, before either train reaches the siding approach signals. However, no matter where the trains are located, he can estimate their arrival times at each end of the single track. The train having the earliest estimated arrival time would be cleared across the single track section first.

The first come, first served rule minimizes delay only under certain conditions. For example, one set of conditions is when the single track run times of the opposing trains are equal, when the acceleration time losses for each train at each location are equal, and when each train has an identical value of time in dollars per hour. If these conditions are not met, it may be desirable to clear the second train instead.

A way to determine which train will arrive first is to take the difference between the two trains'

anticipated arrival times. Subtract the arrival time of train A from the arrival time of the opposing train B. If the difference is positive, train A will arrive first; if negative, train B will arrive first.

The proper train to clear onto the single track section is chosen by comparing this difference with a threshold value. If the difference is greater than the threshold, train A is cleared. If less, train B is cleared. If the threshold is zero, then the first train to arrive will be cleared. Specification of a nonzero threshold appropriately modifies the first come first served rule for the effect of different single track run times, acceleration penalties, and time values. The threshold difference used in the comparison is called the time advantage.

Time advantage causes one train to be favored over another, because the favored train can arrive second, but still be cleared for first movement. For example, suppose Train A has a 5 minute time advantage over Train B. Then train A may be expected to arrive up to 5 minutes later than B, but A would still be cleared onto the single track.

The correct time advantage for resolution of any conflict is determined from the equation given below. This equation was derived from a probability model of

train delay. In its formulation, the probability model includes the running times of each train, the acceleration penalties at each location, and the time value weighting of each train. The dispatching policy is specified by time advantage. The total weighted delay generated is stated as a function of time advantage. By differentiating the probability formulation with respect to time advantage, the optimal time advantage formulation stated below is obtained. The complete derivation is included in following pages.

The result of the derivation is:

$$A_{OPT} = \frac{W_B(R_A + D_{min_B}) - W_A(R_B + D_{min_A})}{(W_A + W_B)}$$

where:

A_{OPT} = Optimal time advantage for train A, in minutes.

W_A = Time value weighting of train A

W_B = Time value weighting of train B

R_A = Single track run time of train A, in minutes.

R_B = Single track run time of train B, in minutes.

D_{min_A} = Acceleration Penalty to A if train A is delayed, in minutes.

D_{min_B} = Acceleration Penalty to B if train B is delayed, in minutes.

The formula is applied differently if one of the trains has already been stopped for a previous meet. Then, the acceleration time loss becomes a certainty. The anticipated time loss should be added to the train's normal single track run time. The D_{min} term for that train is zero since there is no additional acceleration penalty for meeting another train in the same location.

3.2. Example

An eastbound merchandise train valued at $600/hour is in conflict with a westbound coal train valued at $200/hour. The merchandise train will cross the single track segment in 40 minutes, while the opposing coal train requires an hour. At the west end of the single track section is high speed double track, so the merchandise train would be able to begin acceleration to full speed immediately after the meet. The merchandise train's acceleration penalty is only 3 minutes, but the coal train would have to pull out of a low speed passing siding. The coal train's penalty is 20 minutes. What is the proper time advantage to apply?

Merchandise Train A		Coal Train B	
W_A	= \$ 600	W_B	= \$ 200
R_A	= 40 min	R_B	= 60 min
D_{min_A}	= 3 min	D_{min_B}	= 20 min

$$A_{opt} = \frac{200(40+20) - 600(60+3)}{600+ 200}$$

$$= -32.25 \text{ minutes}$$

The arrival time of the merchandise train at the west end of single track is subtracted from the arrival time of the coal train at the east end. The merchandise train may be expected to arrive up to 32.25 minutes later than the coal train, but it should still be cleared across the single track section first.

Suppose the merchandise train arrives exactly 32.25 minutes later. Then the dispatcher should not care which train is cleared first: the delay costs should be equal. Let's test this. Suppose the merchandise train is cleared:

Coal train delay= 32.25 + 40 + 20

= 92.25 min

92.25 (\$200)/60 = \$307.50 cost of delay

If the coal train were cleared:

Merchandise train delay= -32.25 + 60 + 3

= 30.75 min

30.75 ($600)/60 = $307.50 cost of delay

Using the formula, the optimal time advantage can be computed to minimize delay, simultaneously considering differing values of train time, different running speeds and acceleration penalties.

3.3. Derivation of the Dispatching Rule

This local optimization rule is derived for the case of light traffic volume. The sum of delay to each train multiplied by that train's value of time is minimized. Delay is the total time loss relative to passing through an area at full speed. Delay includes both waiting time for an opposing train to clear single track and acceleration time loss after the meet has been completed.

The expected value of delay to a single train, the "train of interest", arriving at the entrance to a single track section, is determined first. Then, by summing the expected delays of each individual train, total train delay is estimated.

Here and elsewhere in the thesis, unexpected mechanical failures are not included in the analysis. Since the dispatcher does not know if or when a mechanical malfunction will occur, he can only plan

based on normal system performance. At some point in the future, the sensitivity of various plans to disruption might be tested, and this included as a measure of the "goodness" of a particular strategy. This is well beyond the scope of this thesis.

Define:

N_{opp} = Number of trains/day entering the single track segment in the direction opposite that of the train of interest.

D_{max} = The maximum time the train of interest will be held waiting for an opposing train to clear, which is the sum of the single track running time and the time advantage applied by the dispatcher. (Only in the case of unexpected mechanical failure would delay exceed this value.) The unit of time of D_{max} must be the same as used in N_{opp}; if N_{opp} is a daily volume, then D_{max} is measured in days.

D_{min} = Minimum time lost while pulling out of the passing siding and reaccelerating to full speed. Computational methods for estimating D_{min} are described in Appendix B.

P(delay)= The probability of delay to the train of interest.

Z = The expected number of opposing train arrivals during time period D_{max}.

AWAIT= Given that delay occurs, the average time

the train of interest must wait for an opposing train to clear the single track segment ahead.

ADV = Time advantage, the threshold difference in train arrival times used by the dispatcher to determine which train will be cleared across the single track segment first. This was fully described in Section 3.1.

R = Running time of the single track segment at full speed, without stops at the sidings on either end.

The expected number of opposing train arrivals during any 24-hour period is N_{opp}. If opposing trains arrivals are uniformly distributed throughout the day, the expected number of arrivals during any shorter period D_{max} is proportional to the length of the time period D_{max}, giving:

Z = Expected number of arrivals during time period D_{max}

$$= N_{opp} \times D_{max} \qquad (2)$$

If train arrivals are Poisson, the probability of exactly k arrivals during D_{max} is:

$$P(k) = \frac{e^{-Z} Z^{k}}{k!} \qquad (3)$$

Delay will occur if one or more arrivals occurs in period D_{max}. Since traffic density is assumed to be light, the probability of more than one arrival in time period D_{max} is assumed to be so small that it can be neglected. The exponential term approaches one as Z approaches zero. One factorial equals one. This leaves only the Z term, equaling $N_{opp} \times D_{max}$ which is raised to the first power.

Therefore:

$$P(\text{delay}) = Z = N_{opp} \times D_{max} \quad (4)$$

The maximum wait D_{max} at a location (for meeting one train) is the running time of the opposing train minus any time advantage. In section 3.2, the coal train had a maximum wait of 40 + 32.25 = 72.25 minutes. The merchandise train had a maximum wait of 60 - 32.25 = 27.75 minutes.

If opposing train arrivals are uniformly distributed, the average wait to trains which are delayed is half the maximum wait, plus a fixed acceleration penalty, which occurs regardless of the length of the wait for opposing traffic. Call this average wait time AWAIT.

$$\text{AWAIT} = D_{max}/2 + D_{min} \quad (5)$$

One of the initial assumptions was that the

probability of more than one arrival in D_{max} is negligible. Equation 5 holds true as long as D_{max} is not increased to the point where this initial assumption is violated.

The expected value of delay the train of interest, and also to each train arriving at the entrance to a single track section is the probability of delay to that train, times the average delay given that delay occurs:

Expected delay= P(delay) x AWAIT (6)

to each train

A simplified version of the complete formula follows, assuming first come, first served train sequencing (ADV=0), equal single track run times R and no acceleration penalties.

$$D_{max} = R \quad (ADV=0) \qquad (7)$$

and $P(delay) = R \times N_{opp}/24$ (Conversion factor of 24 allows running times to be measured in hours.) (8)

and $AWAIT = R/2 \qquad (D_{min} = 0)$ (9)

so Expected delay $= R \times N_{opp}/24 \times R/2$

to each train

$$= N_{opp} \times R^2/48$$

Total delay caused by a single track segment is the sum of delay at the east end plus delay at the west end. If the same number of trains are operated in each direction, (dropping the opp subscript), N eastbounds can be delayed, and N westbounds can be delayed, for a total of 2 x N trains. Total train delay is then given by:

$$\text{Total daily delay} = 2 \times N \times N \times R^2 / 48$$
$$= N^2 R^2 / 24 \qquad (10)$$

The following expands Equation 10 to include individual run times, acceleration penalties and weightings for each train. "R+ADV" or "R-ADV" have been substituted for the D_{max}. The factor 24 converts daily train volumes into hourly volumes.

For trains in direction A:

$$D_{max} = R_B + ADV$$
$$P(\text{delay}) = N_B (R_B + ADV) / 24$$
$$AWAIT = (R_B + ADV)/2 + D_{min_A}$$

Likewise, for trains in direction B:

$$D_{max} = R_A - ADV$$
$$P(\text{delay}) = N_A (R_A - ADV) / 24$$
$$AWAIT = (R_A - ADV)/2 + D_{min_B}$$

Using equation 6, compute the expected value of delay to each train in each direction. Each term is premultiplied by W, the cost or priority weighting factor, and by N_A and N_B, the number of trains in directions A and B respectively.

$$D_{tot} = W_A N_A \left(N_B (R_B + ADV)/24 \left((R_B + ADV)/2 + D_{min_A}\right)\right) + W_B N_B \left(N_A (R_A - ADV)/24 \left((R_A - ADV)/2 + D_{min_B}\right)\right) \quad (11)$$

Combining terms:

$$D_{tot} = N_A N_B/24 \left(W_A/2 (R_B + ADV)^2 + W_A (R_B + ADV) D_{min_A} + W_B/2 (R_A - ADV)^2 + W_B (R_A - ADV) D_{min_B}\right) \quad (12)$$

Now, to optimize, differentiate with respect to ADV and set the result equal to zero:

$$dD_{tot}/dADV = N_A N_B/24 \left(W_A (R_B + ADV) + W_A D_{min_A} - W_B (R_A - ADV) - W_B D_{min_B}\right)$$

$$0 = W_A R_B + W_A ADV + W_A D_{min_A} - W_B R_A + W_B ADV - W_B W_{min_B}$$

$$0 = W_A (R_B + D_{min_A}) - W_B (R_A + D_{min_A}) + ADV (W_A + W_B)$$

Gathering terms yields the expected result:

$$A_{OPT} = \frac{W_B (R_A + D_{min_B}) - W_A (R_B + D_{min_A})}{(W_A + W_B)} \quad (13)$$

4. Train Meet Planning With Uncertain Run Times

4.1. Definitions, Terms and Relationships

Meet planning procedures rely on accurate predictions of train running times. However, some trains will run slower than predicted; others will run faster. Therefore, when a meet is projected, the trains' exact arrival times are not known. These arrival times can vary; and can be characterized by a distribution. Train delay, which is the difference between two trains' arrival times, will also be characterized by a distribution. The mean and standard deviation can be computed if the distribution is known.

Two methods can be used to predict train meet delay. The simplest and usual method is to predict delay based solely on average running times and to compute the estimated delay as the difference in projected train arrival times. The preferred method is to compute the expected value of delay by considering the entire range of possible running times.

If two opposing trains' positions were totally unknown, and the two trains had equal priorities, delay could range from zero to the full single track running time, with equal likelihood. This is true unless there is an unanticipated mechanical failure. On the average, with uniformly distributed arrivals throughout the day, it would be one half the run time

between sidings. (In general it is half the maximum wait D_{max}, which depends on train priorities.) Once the expected delay converges to half the intersiding run time, the amount of estimated delay becomes irrelevant.

The use of expected delay in meet planning automatically establishes a "planning horizon." The planning horizon is defined as the point in the future beyond which the optimal meet strategy is unaffected by future meets. There is no need to project meets past this point since they cannot affect the current train meet decision. Since each distant future meet has the same expected delay, the sum of expected delays will cancel out, no matter what meet strategy is chosen. The use of expected delay also discounts future delay savings when they exist.

For some combinations of slow and fast trains, it may become desirable to shift the meet location. If the meet had been "locked in" at a suboptimal location, a time penalty is incurred equal to the amount of delay that could have been saved. The meet has been locked in if the switches and signals have already been lined or if train orders determining the meet location have already been issued.

The probability the proper siding will be selected is a function of the level of variance and

the number of sidings across which the meet is being projected. It is insensitive to varying distances and run times between sidings. This implies a meet can be projected farther ahead in the future if the trains run slow and if the distance between the sidings is large. Since fewer sidings are available to choose from, the probability of selecting the correct siding is improved.

Therefore, the planning horizon is measured in terms of the number of sidings across which a meet can be projected. It is not directly measured in miles or in hours. As measured in these units, it is directly proportional to the distance and running time between sidings.

The penalty due to locking in a meet is strongly related to the probability the proper siding will be selected. If the minimum delay location is always selected, then the penalty is always zero. If the probability of choosing the correct siding is less than 100%, there is an associated penalty.

The location of a projected meet can usually be determined before the amount of delay in that meet can be estimated. Many combinations of slow and fast trains could yield the same optimum meeting point, but each combination would have a different level of delay. As the amount of uncertainty in train arrival

times increases, the expected delay approaches half the intersiding run time. But, in order to affect the probability of selecting the proper meet location, the level of uncertainty must be high enough to shift the meet. Small amounts of uncertainty may have absolutely no effect on the probability of selecting the proper location, but any uncertainty will affect the expected value of delay.

The arrival time of the later of the two trains is the meet completion time. Due to variable running times, it cannot always be determined which train will actually arrive first. The expected meet completion time, taking account of variability, is usually later than the projected arrival time of the second train. This is because of the possibility the train expected to arrive first may actually arrive second.

The standard deviation in train arrival times is needed in order to compute the expected delay. The standard deviation can be estimated as a percentage of train running time from the most recently reported location. After two trains meet, they depart the meet location at the same time. As shown in the next section, the variance in this departure time distribution is less than the variance of either trains' arrival time distribution. This is one of the few cases where a probabilistic interaction has a

variance-reducing effect.

Because of the large number of terms used and defined in the previous paragraphs, these definitions are provided as review:

Estimated delay- Meet delay computed as the difference between average train arrival times.

Expected delay - Meet delay based on the entire distribution of train running and arrival times.

Locked in- A condition where a meet location has been selected in advance and cannot be changed.

Meet completion time- The time of arrival of the second train, after which both trains may proceed.

Penalty- The difference between the actual delay and the amount of delay had the meet been shifted to the optimal (minimum delay) location.

Planning horizon- the point in the future beyond which the optimal meet strategy is unaffected by future meets. It is a function of the number of sidings across which the meet is being projected.

4.2. General Procedure

At each potential meeting location, each train's potential arrival times can be characterized by a probability density function. Given this continuous distribution of potential arrival times, the problem can be solved directly. However, the direct solution is not computationally convenient.

It is more convenient to use an approximate means of solution. The density function is divided into K discrete intervals, each interval having equal probability 1/K. The solution can be made as exact as desired by selection of sufficiently small intervals.

Then K possible arrival times are generated for each train. Each arrival time represents one interval. This time can be computed as the centroid of the interval it represents, or alternately, such that half the probability in the interval is on each side. Define A_i = A's arrival times and B_j = B's arrival times; $1 < i < K$, $1 < j < K$.

A K x K matrix of train delays is then formed. Train A's times A_i are placed across the top, and train B's times B_j down the left side of the matrix. The difference between these times:

$$d_{ij} = \text{abs}(A_i - B_j)$$

is placed in each cell ij. This difference represents the delay which would result should that combination of train arrivals actually occur. Each cell has equal probability $1/K^2$.

The expected value of delay can then be computed as:

$$D_{exp} = \sum_{ij=1}^{K} d_{ij} / K^2 \qquad (14)$$

This estimate of the expected value of delay

assumes the meet will not be shifted. Therefore, it is correct if either the variance is small (the meet doesn't need to be shifted away) or if the meet has been locked in.

The meet completion time matrix is found as:

$$m_{ij} = \max(A_i , B_j).$$

From this matrix, the expected meet completion time is found as:

$$M_{exp} = \sum_{ij=1}^{K} m_{ij} / K^2 \qquad (15)$$

and the variance in train departure times after the meet as:

$$\sigma^2 = \sum_{ij=1}^{K} m_{ij}^2 / K^2 - M_{exp}^2 \qquad (16)$$

The average train arrival times, A_{avg} and B_{avg}, are:

$$A_{avg} = 1/K \sum_{i=1}^{K} A_i$$

$$B_{avg} = 1/K \sum_{j=1}^{K} B_j$$

The "estimated delay" is computed as:

$$D_{est} = abs(A_{avg} - B_{avg})$$

To allow for the possibility of shifting the meet location, delay matrices and the estimated delay must be computed for all possible meeting locations. These meeting locations are numbered 1 through L. Figure 2 and Table 1 in section 4.3 show how this delay is

computed for different ij combinations. By adding this additional subscript representing meeting location:

A_i becomes A_{iz}

B_j becomes B_{jz}

d_{ij} becomes d_{ijz}

$D_{est_z} = abs(A_{avg_z} - B_{avg_z})$

The d^*_{ij} matrix is then created by selecting the location from d_{ijz} for each combination ij which minimizes that combination's delay. This minimum delay becomes d^*_{ij}. This assumes that, although the dispatcher is planning the meet, he has not yet committed to it by lining switches and clearing signals. Therefore, he retains the flexibility to adapt his plan to changing operating conditions. For each combination of slow and fast trains, by observing the trains' progress, he can select the optimum location.

$d^*_{ij} = \min(d_{ijz}),\ 1 < z < L$

Then the expected value of delay, allowing for the meet to be shifted, is:

$$D^*_{exp} = \sum_{i,j=1}^{K} d^*_{ij} / K^2 \qquad (17)$$

It is this value of expected delay, D^*_{exp} which converges to half the intersiding run time. The maximum value of d^*_{ij} is the intersiding run time,

equal to D_{max}. Should delay at any location exceed this value, it would not be the optimal meet location and the meet would be shifted elsewhere. Lesser delays are also possible. The values of d^*_{ij} are distributed between zero and the full intersiding run time; the average converges to half the intersiding run time as the uncertainty in arrival times increases. D_{exp}, defined in equation (14), does not converge to any value: it continues to increase without bound as uncertainty in arrival times is increased.

The planned meet location N is the meet location which minimizes the estimated delay:

$$N = \arg(\min(D_{est_z})) \quad 1 < z < L$$

Then the expected penalty P_{exp} for locking in the meet is:

$$P_{exp} = \sum_{i,j=1}^{K} (d_{ijN} - D^*_{ij})/_{K^2} \qquad (18)$$

The probability the minimum delay meet location is selected is determined by comparing all d_{ijz} matrices for each ij combination. The number of times location N is chosen is summed. When divided by K^2 (the number of cells) this yields the probability the correct siding will be chosen.

Define:

$X_{ij} = 1$ if $N = \arg(\min(D_{ijz}))$ $1 < z < L$

$= 0$ otherwise

Then the probability H the correct (minimum delay) siding will be selected is:

$$H = \sum_{i,j=1}^{K} X_{ij} / K^2 \qquad (19)$$

4.3. Numerical Examples

The examples in this section are intended to illustrate the general principles stated earlier.

Figure 2 shows a rail line having five sidings A

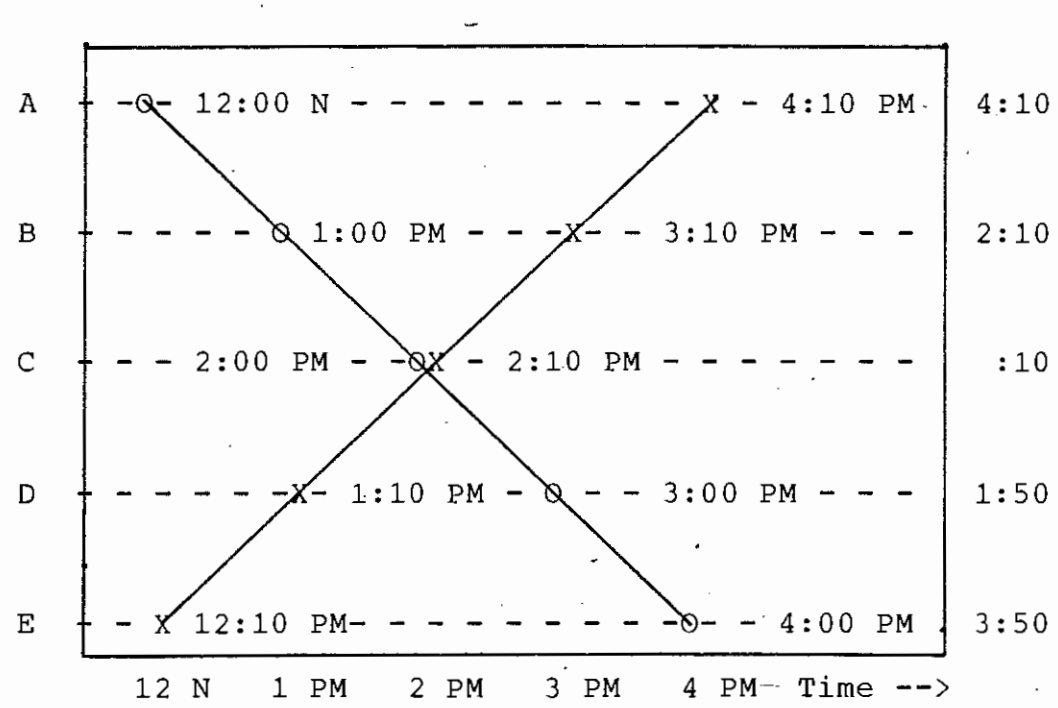


Fig 2 - Unresolved Time Distance Diagram, showing arrival times and computed delay times for a meet in each location A through E.

through E spaced evenly at one hour apart. At 12:00 noon, a southbound departs point A; a northbound departs point E at 12:10 . Based on the average one hour running time, the optimal location for the meet is point C. 10 minutes delay to the southbound is anticipated.

After the dispatcher has planned the meet for C, suppose the northbound runs fast, requiring only 40 minutes between sidings. The southbound runs slow, requiring one hour 20 minutes between each siding. Then the respective arrival times and delays would be given by Table 1.

Arrival Times and Train Delay

at Possible Train Meet Locations

Location	Arrival Times		DELAY
A	12:00 N	2:50 P	2:50
B	1:20 P	2:10 P	:50
C	2:40 P	1:30 P	1:10
D	4:00 P	12:50 P	3:10
E	5:20 P	12:10 P	5:10

Table 1- 1:20 run time south, 0:40 north.

Now the optimum meet location has been shifted from C to B. If the meet had been locked in at C, a penalty of 20 minutes= 1:10- 0:50 would be incurred.

This illustrates the cost of locking in meets and the desirability of keeping all meet/pass options open.

Lesser speed variations may not shift the meet location, but will still affect the amount of delay generated in any meet. For example, if the southbound requires 1 hour 10 minutes, and the northbound 50 minutes, the delay is given by Table 2.

Arrival Times and Train Delay

at Possible Train Meet Locations

Location	Arrival Times		DELAY
A	12:00 N	3:30 P	3:30
B	1:10 P	2:40 P	1:30
C	2:20 P	1:50 P	:30
D	3:30 P	1:00 P	2:30
E	4:40 P	12:10 P	4:30

Table 2- 1:10 run time south, 0:50 north

In this case actual delay was 30 minutes when 10 minutes had been predicted. However, C would still be the optimal location. Many combinations such as this could be formed, all having C as the optimal meet location. The meet location can be determined prior to the time when the delay can be accurately estimated.

A new example having different values of the train arrival times is presented in Table 3. Based on

the average run times, the meet described in Table 3 would be perfectly timed: it would have no delay. But the expected value of delay is positive, because of the potential variation in train arrival times.

To improve readability, in the following Tables 3, 4 and 5, the words "Fast", "Average" and "Slow" have been substituted for numerical subscripts i or j equal 1, 2, and 3, respectively. Eastbound arrival times are A_i's, while westbound arrival times are B_j's.

Arrivals at Siding C

(Each event equally likely)

	Eastbound A_i's	Westbound B_j's
Fast	2:40	2:40
Average	3:00	3:00
Slow	3:20	3:20

Matrix of d_{ij}'s

Westbound Arrivals		Eastbound Arrival Time: Fast 2:40	Average 3:00	Slow 3:20
Fast	2:40	0	20 min	40 min
Average	3:00	20 min	0	20 min
Slow	3:20	40 min	20 min	0

Table 3- Matrix of d_{ij}'s

Estimated Delay = 0 minutes= 3:00 - 3:00

Applying Equation 14, compute the expected value of delay as:

$$D_{exp} = (0+20+40+20+0+20+40+20+0)/9$$

$$= 17.8 \text{ minutes}$$

Another table can be built of the m_{ij}'s so that the meet completion time and variance can be estimated. Using the same arrivals as Table 3:

Matrix of m_{ij}'s

Eastbound Arrival Time

Westbound Arrivals		Fast 2:40	Average 3:00	Slow 3:20
Fast	2:40	2:40	3:00	3:20
Average	3:00	3:00	3:00	3:20
Slow	3:20	3:20	3:20	3:20

Table 4- Computing meet completion time and the Standard deviation after the meet

Taking the average of these times, according to equation 15, yields the expected meet completion time:

(2:40 + 3:00 + 3:20 + 3:00 + 3:00 + 3:20 + 3:20 + 3:20 + 3:20) /9 = 3:09

This is later than 3:00 which would have been predicted if uncertain run times had not been considered.

The plus or minus twenty minute time spread of both arriving trains yields a sample standard deviation before the meet of 20 minutes. The standard deviation after the meet can be estimated from Table 4. Equation 16 could be applied directly. However, for hand calculation, the more usual formula for standard deviation has been applied.

3:09- 2:40= 29 min -->		29^2 times 1
3:09- 3:00= 9 min -->		9^2 times 3
3:20- 3:09= 11 min -->	+	11^2 times 5

Sample Variance = 1689 min^2

Stand. dev = Square Root of 1689/8= 14.5 min

Note the 14.5 minute estimate implies a substantial reduction from the previous twenty minute spread.

Another quantity of interest is the probability the correct meet location will be chosen. To estimate this probability, delay matrices are required for all potential meeting locations. It is convenient now to

return to the original example of Figure 2. This is done in Table 5 for locations B, C and D. These locations correspond to numerical Z subscripts 2, 3, and 4, respectively. Locations A and E are not optimal meeting for any ij combination, so to save space, matrices for those locations have been omitted.

Train Arrival Times

	Northbound A_{iz}'s			Southbound B_{jz}'s		
	D	C	B	B	C	D
Slow	1:30	2:50	4:10	1:20	2:40	4:00
Average	1:10	2:10	3:10	1:00	2:00	3:00
Fast	12:50	1:30	2:10	12:40	1:20	2:00

Delay Matrix - B

		Northbound Arrival Time		
Southbound		Fast	Average	Slow
Arrivals		2:10	3:10	4:10
Fast	12:40	1:30	2:30	3:30
Average	1:00	1:10	2:10	3:10
Slow	1:20	:50	1:50	2:50

Delay Matrix - C

Southbound Arrivals		Northbound Arrival Time: Fast 1:30	Average 2:10	Slow 2:50
Fast	1:20	**:10**	**:50**	1:30
Average	2:00	**:30**	**:10**	**:50**
Slow	2:40	1:10	**:30**	**:10**

Delay Matrix - D

Southbound Arrivals		Northbound Arrival Time: Fast 12:50	Average 1:10	Slow 1:30
Fast	2:00	1:10	:50	**:30**
Average	3:00	2:10	1:50	1:30
Slow	4:00	3:10	2:50	2:30

Table 5 - Delay matrices for B, C and D

Using the above data, one can compute the probability the correct meeting point will be chosen. Each possible combination of slow and fast trains will have an optimal meeting location. In Table 5, the proper location for each interaction has been underlined, indicating $X_{ij}=1$ in equation 19. While most of the meet combinations ideally occur at siding

C, there are two exceptions: a fast northbound meeting a slow southbound should be shifted to point B; a slow northbound meeting a fast southbound should be shifted to point D. Therefore the probability that the correct meet location will be chosen is 7/9.

Since two meet combinations should be shifted away from C, a penalty exists for locking in the meet now. The expected penalty is ((1:10-:50)+(1:30-:30))/9 = 8.9 minutes. (For all other ij combinations, where the optimal location for the meet is siding C, d_{ijN} - D^*_{ij} equals zero in equation 18.)

The expected value of delay is increased 8.9 minutes if the flexibility to shift the meet is not retained. Two expected values of delay can be computed, as defined in equations 14 and 17 respectively:

1) If the meet is locked in at C, or

2) If the flexibility to shift the meet is retained.

The expected value of delay if the meet is locked in at C is computed the same as it was in Table 3. The procedure (equation 14) is to sum all the d_{ij}'s and divide by the number of elements:

D_{exp}= (10+50+90+30+10+50+70+30+10)/9

= 38.9 minutes.

If the meet is not locked in at C, the procedure

is modified. Instead of using only C's matrix, choose the element d^*_{ij} which is the delay at the best meeting point for that combination. The 30 minute delay at point D is preferable to the 90 minute delay at point C. The 50 minute delay at point B is preferable to the 70 minute delay at point C. Substituting these two elements (equation 17), compute:

$$D^*_{exp} = (10+50+30+30+10+50+50+30+10)/9$$

$$= 30 \text{ minutes}$$

In this example, the expected delay D^*_{exp} has already converged to half the single track run time. The penalty for locking in the meet is 38.9 min - 30 min = 8.9 minutes.

4.4. Simulation Results

The procedure defined in section 4.2 has been computerized, and several simulations were run using various passing siding spacings, levels of variance, and distances between opposing trains. The levels of expected delay, delay penalty, and probability of selecting the correct siding were computed. The simulations were performed in order to estimate an approximate planning horizon for the branching procedure under various sets of circumstances.

All simulations are based upon uniformly spaced passing sidings, with equal opposing train priorities and equal running times between sidings. The purpose of this section is to develop relationships and present a general methodology. This is best accomplished by keeping the complexity of the simulations to a minimum. Non-uniformly spaced sidings, unequal run times and priorities are subjects for future study.

When a train arrives at the approach signal to a passing siding, the decision as to whether to divert into that siding, or to proceed to the next meet location, must be made. This decision is made by determining the location of the opposing train and choosing a meet location. The number of passing sidings between the two trains can be counted. Figure 3 shows the case where a meet is being projected across two passing sidings.

It is convenient to use the number of sidings in place of miles as a unit of distance. Fractional amounts are possible. In Figure 3, Train A is exactly two sidings from Train C. Train B is 2.5 sidings from

C, as it is halfway between sidings.

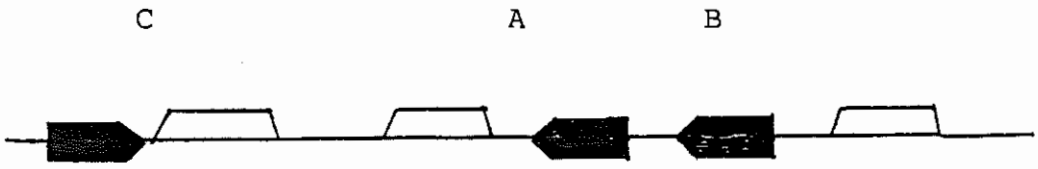


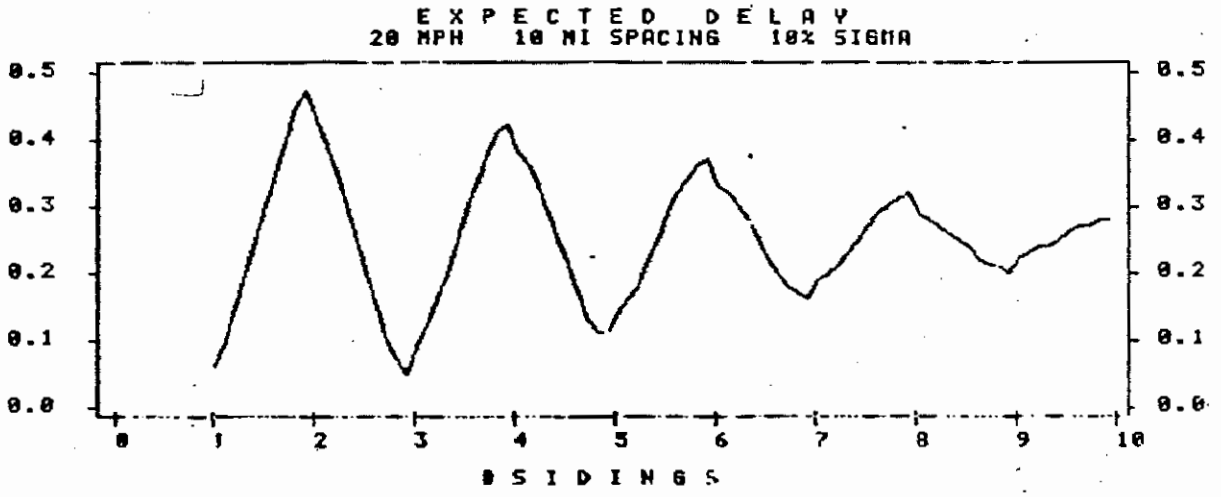
Figure 3 - Meet projected across two sidings

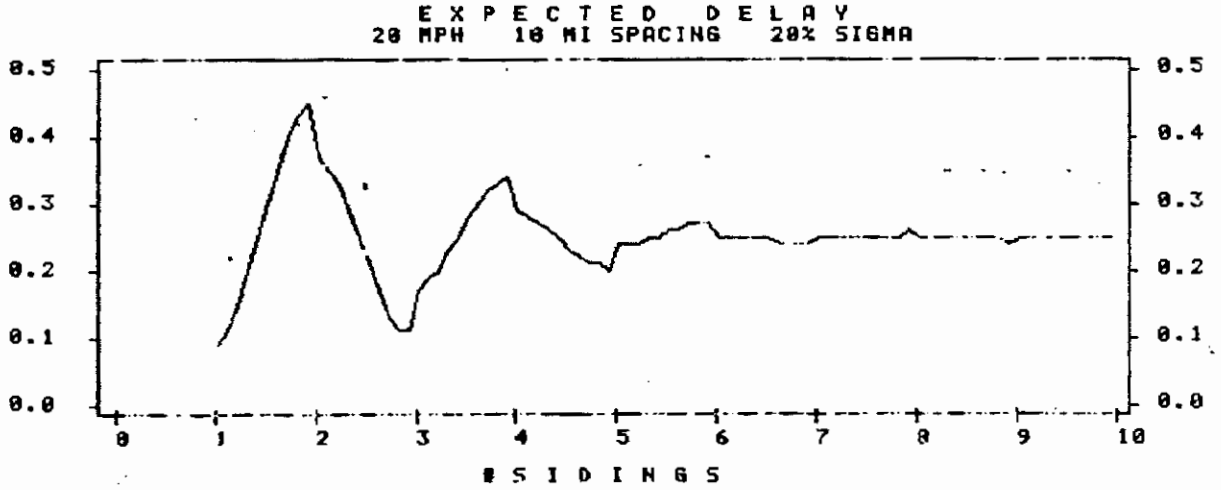
The more variance, the faster the expected value of delay converges to one half the intersiding run time. At 27% sigma, the average level of variance found in Section 2, the convergence is complete by the time the meet is projected across four sidings. The planning horizon, when measured in hours, depends on the running time between sidings. Figures 4a through 4h show the behavior of the expected value of delay through a range of sigma values, train speeds and siding spacings. The expected delay in hours is plotted as a function of the number of sidings across which the meet is projected.

When examining Figure 4, notice the general trend of convergence to half the single track running time. The more variance, the faster the convergence. But the convergence rate is unaffected by varying train speeds or intersiding spacings (Figures 4e through 4h).

The most striking feature of all the simulations

Fig 4a,4b,4c - Expected Delay in hours as a function of the number of sidings across which the meet is being projected





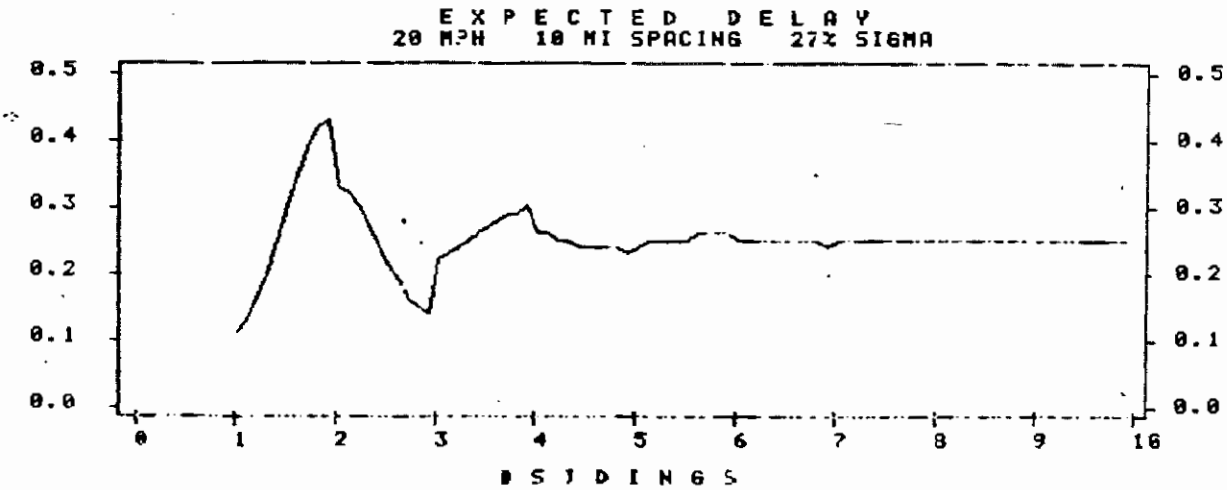
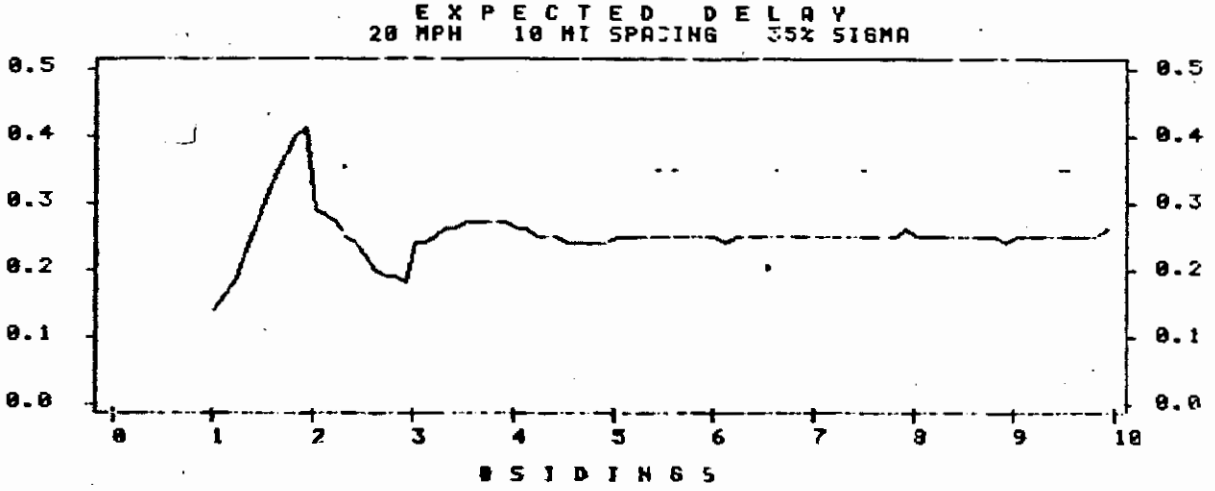
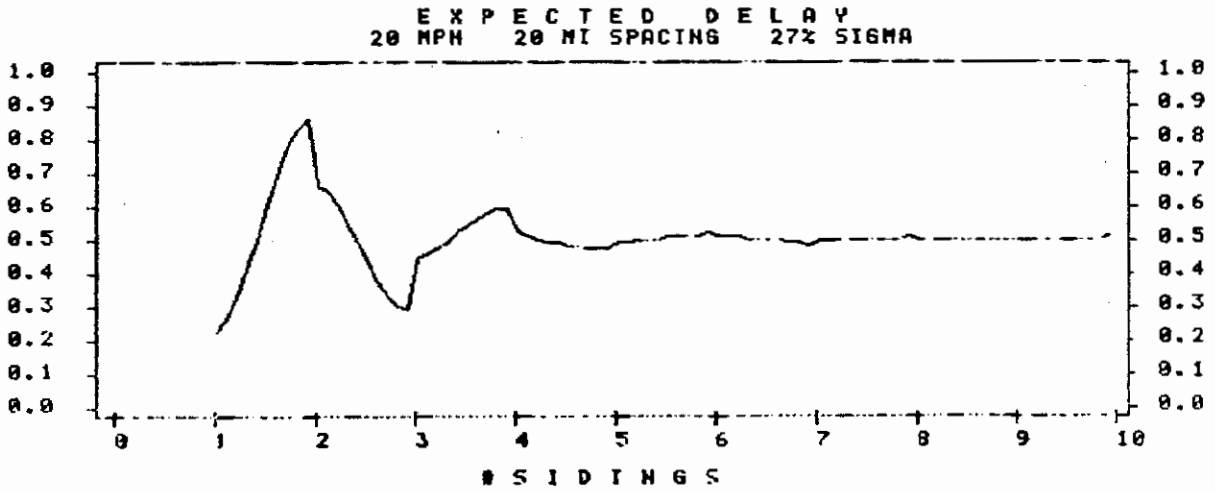


Fig 4d,4e,4f (ctd)





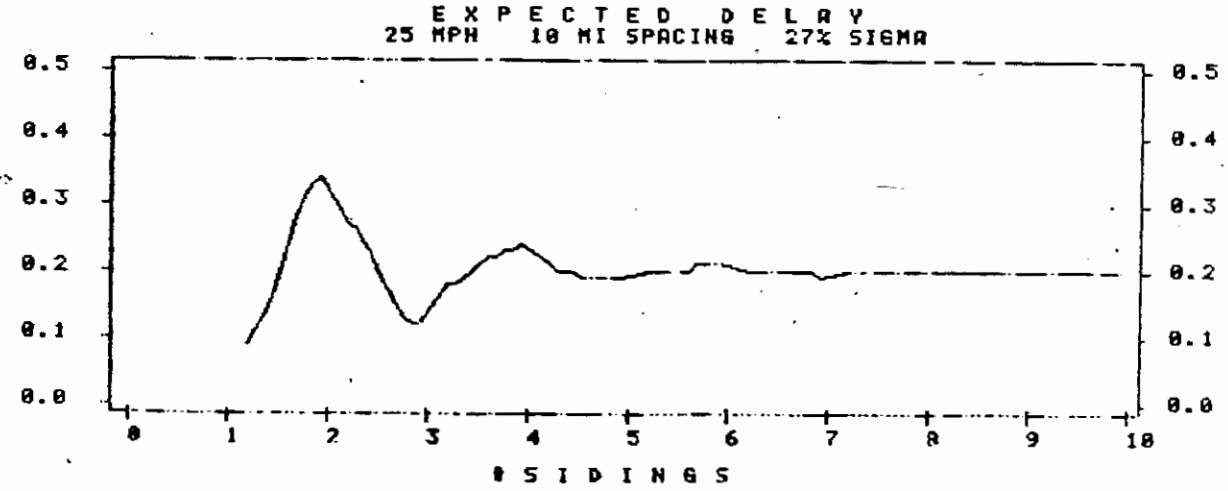
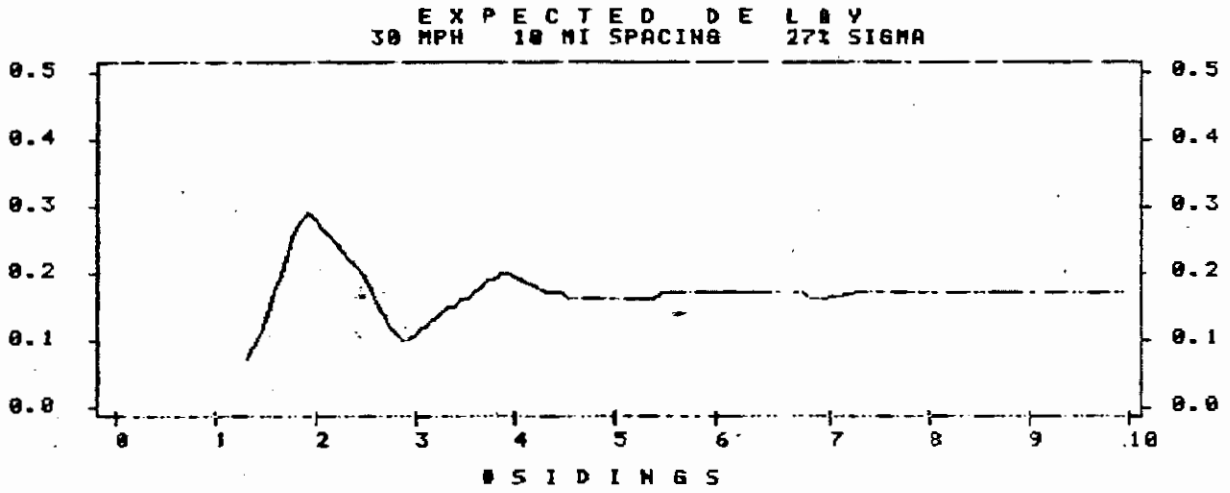
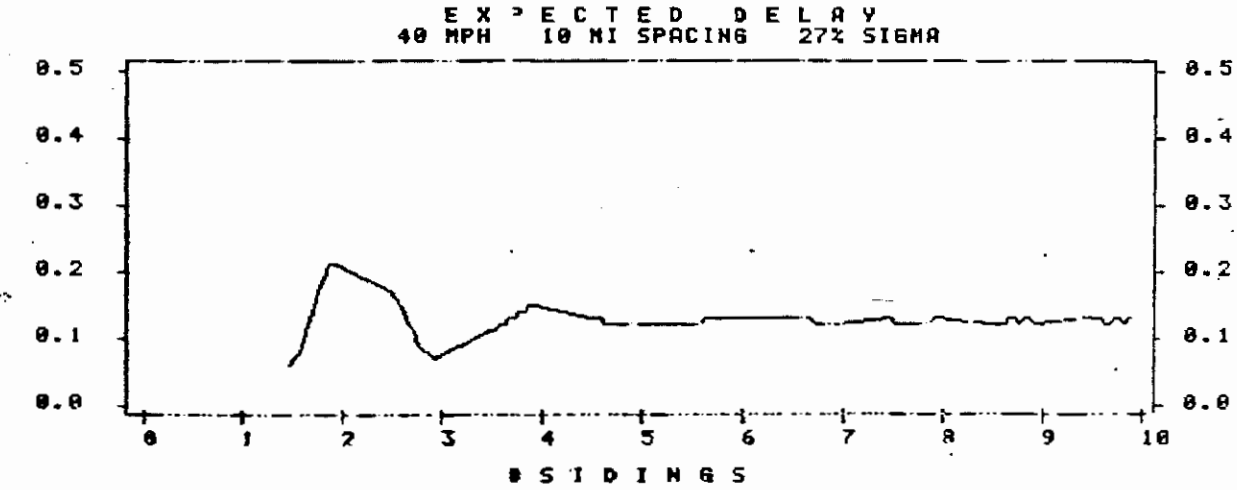


Fig 4g, 4h (ctd)


EXPECTED DELAY
30 MPH 10 MI SPACING 27% SIGMA
0.5
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10
SIDINGS


EXPECTED DELAY
40 MPH 10 MI SPACING 27% SIGMA
0.5
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10
SIDINGS

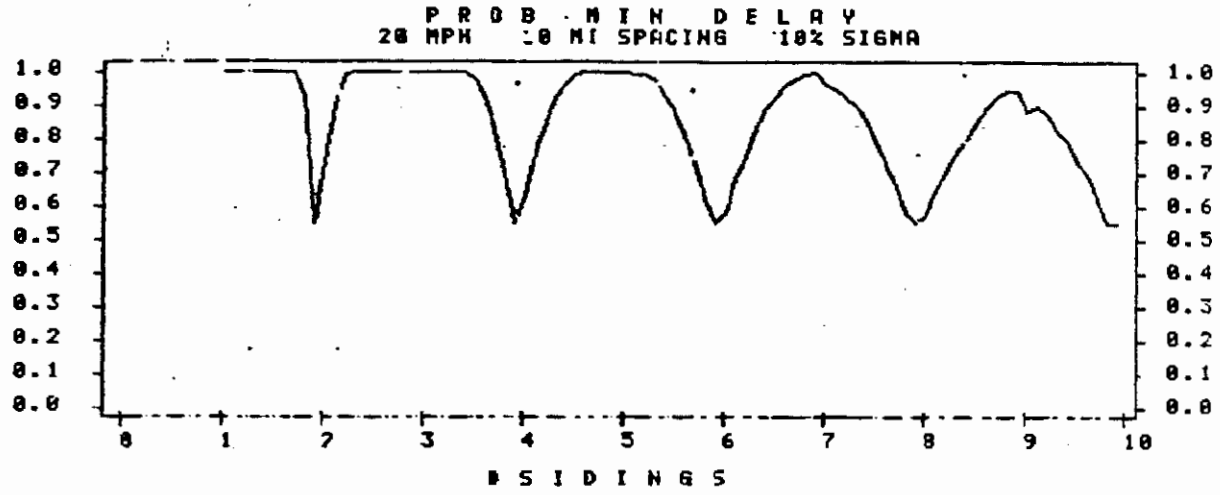
is the periodicity of the results. They tend to repeat every two sidings. This occurs for geometric reasons. With evenly spaced sidings, when an odd number of sidings are in the system, the meet is perfectly timed. Figure 5 shows the case where a meet is projected across three sidings. The meet can occur in the middle with very little delay.

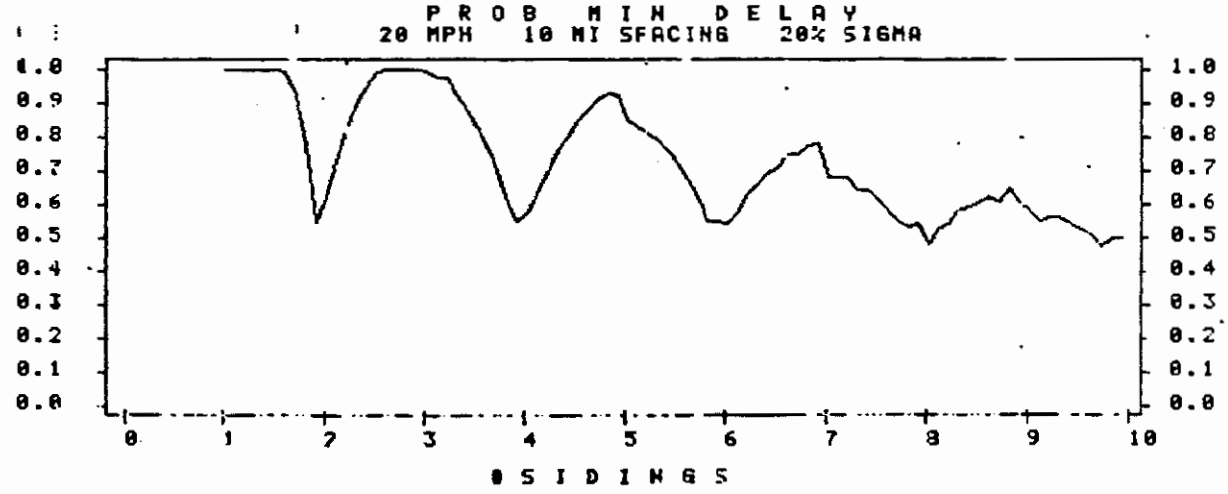
Figure 5 - Meet projected across three sidings

However, with an even number of sidings as in Figure 3, the meet can occur in either siding. In either case the delay will be the full running time of the single track segment.

This periodicity also extends to the probability of selecting the proper siding. The Probability of Minimum Delay (Figures 6a through 6h) dips to 50% at each even passing siding, then improves again. If both trains arrive at the end of single track at precisely the same moment, the probability of selecting the proper meet location is only 50%. There is no reason to favor one location over the other; each siding is equally apt to be the correct meeting point. With two segments of single track having three passing sidings,

Fig 6a,6b,6c- Probability of selecting the minimum delay meet location as a function of the number of sidings across which the meet is being projected





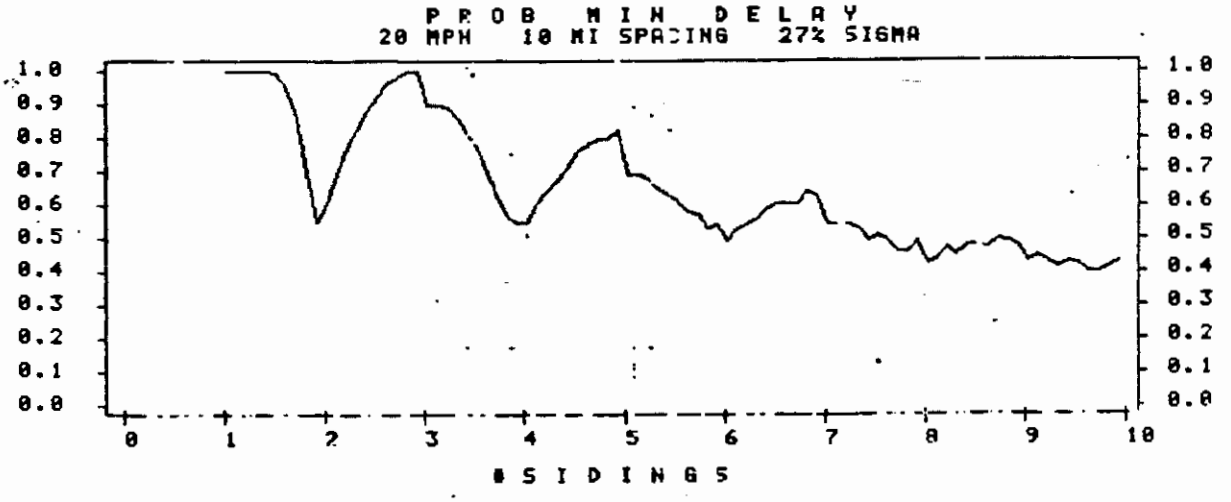
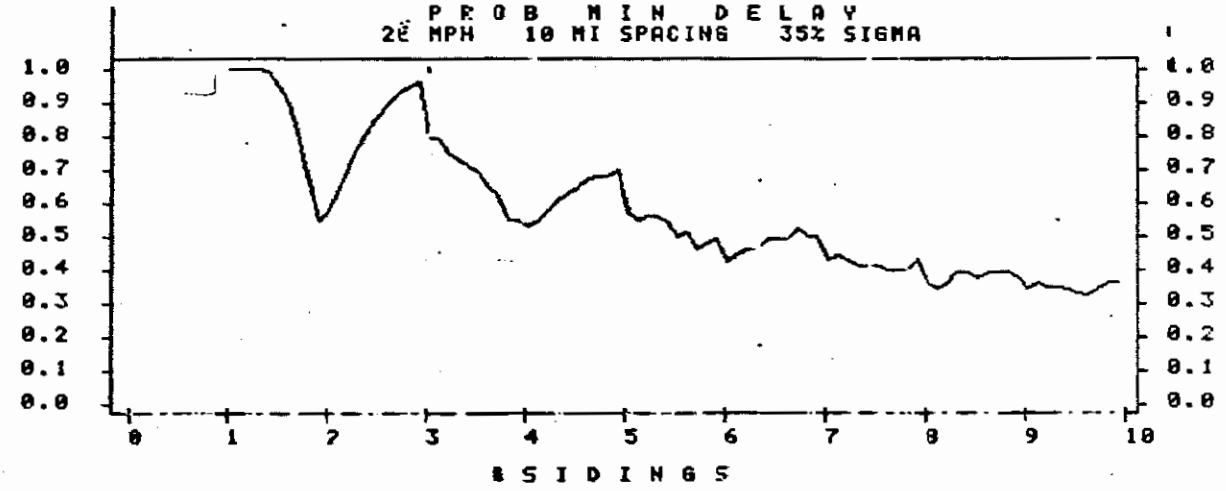
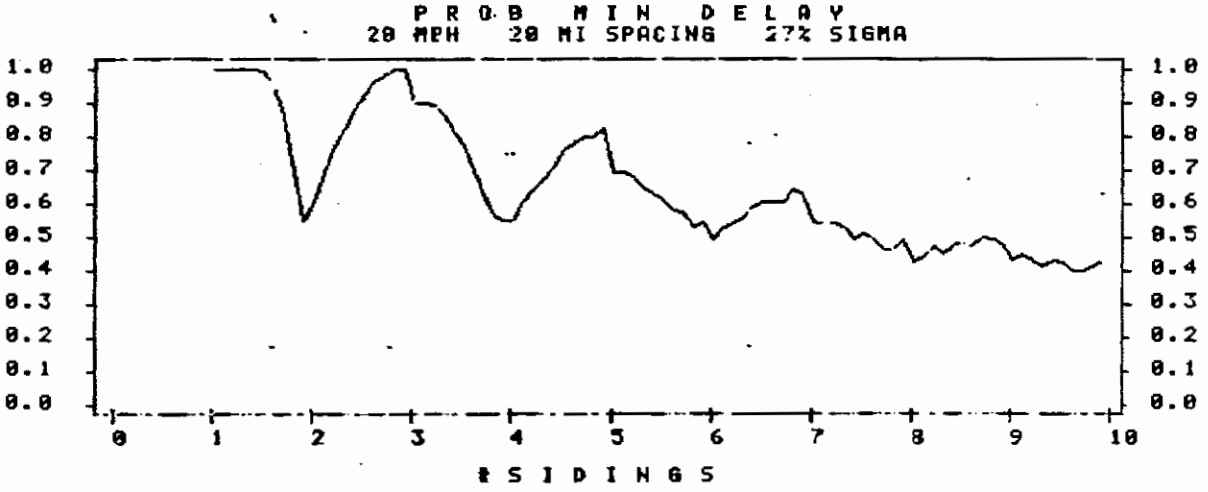


Fig 6d,6e,6f (ctd)





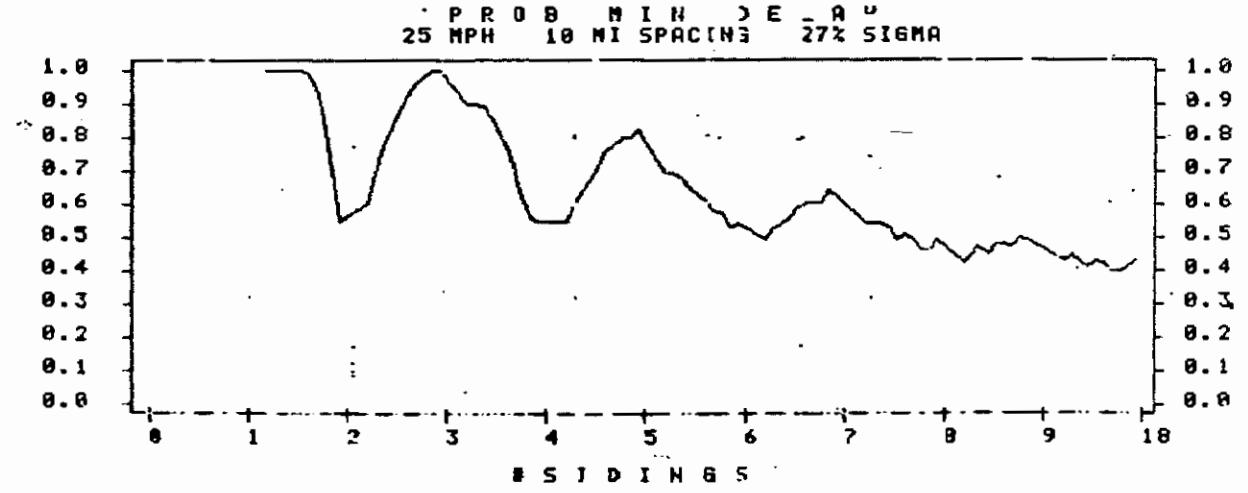
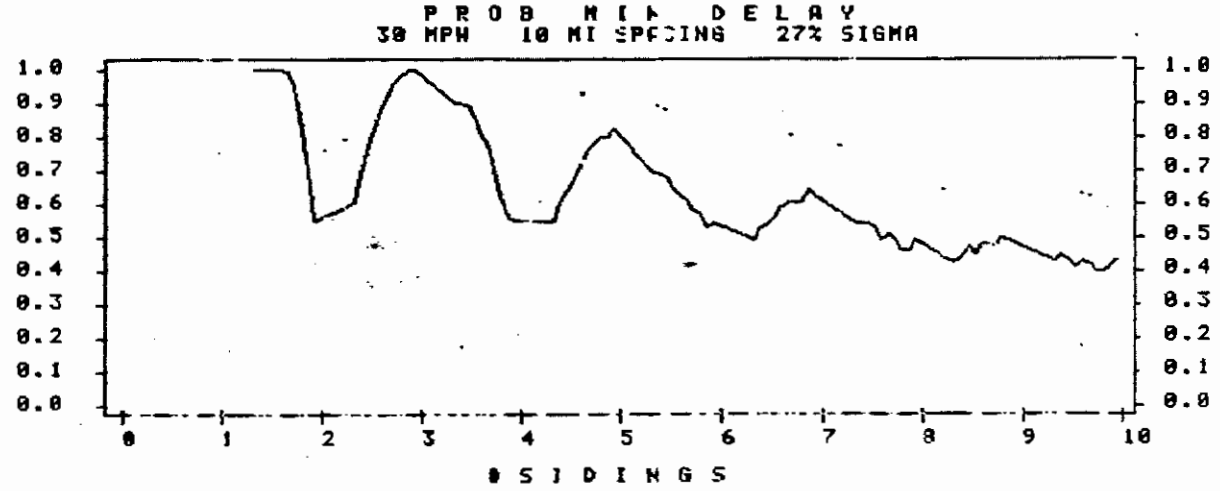
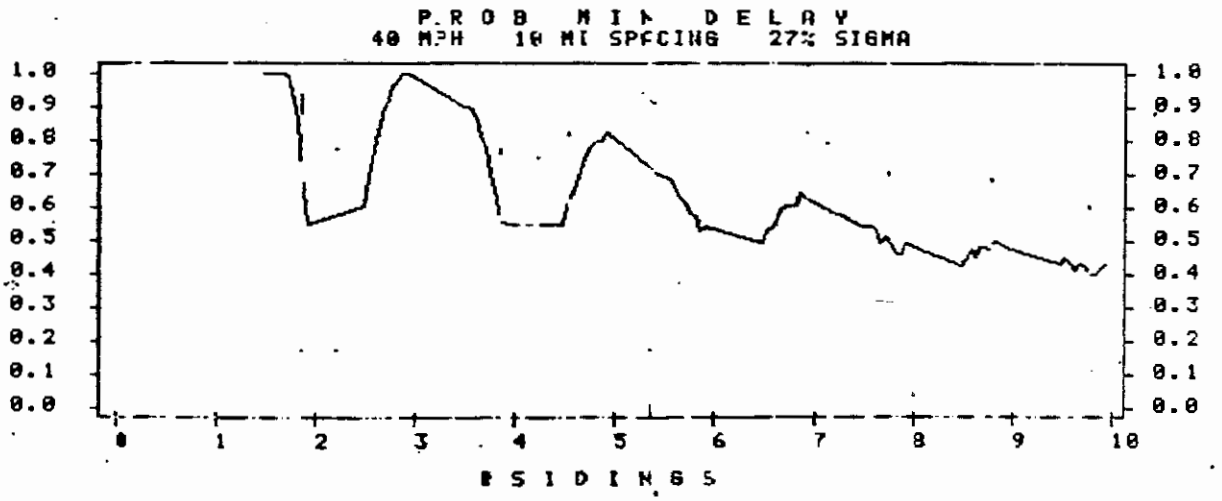


Fig 6g,6h (ctd)


PROB MIN DELAY
30 MPH 10 MI SPACING 27% SIGMA
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10
SIDINGS

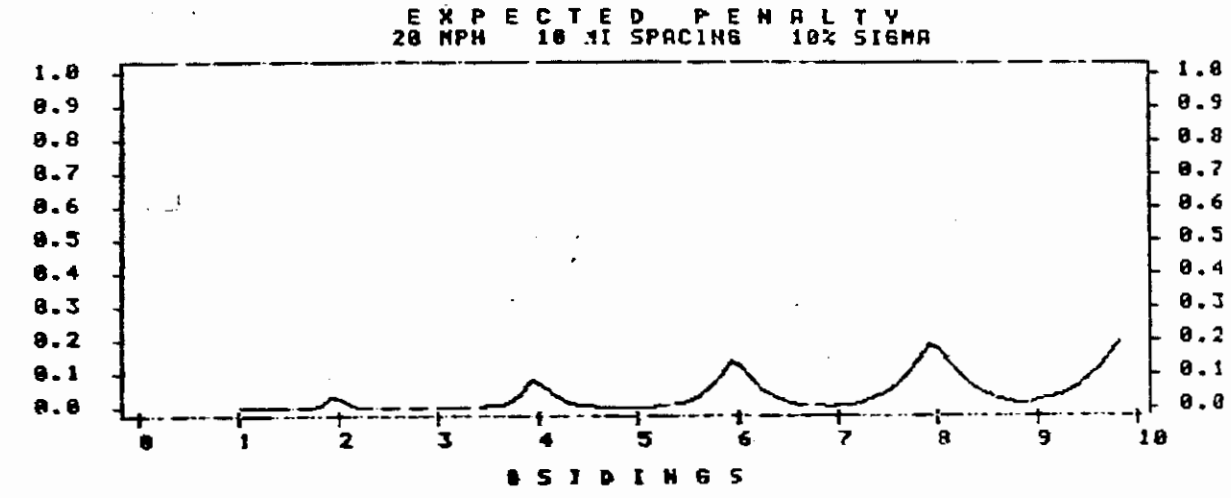

PROB MIN DELAY
40 MPH 10 MI SPACING 27% SIGMA
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10
SIDINGS

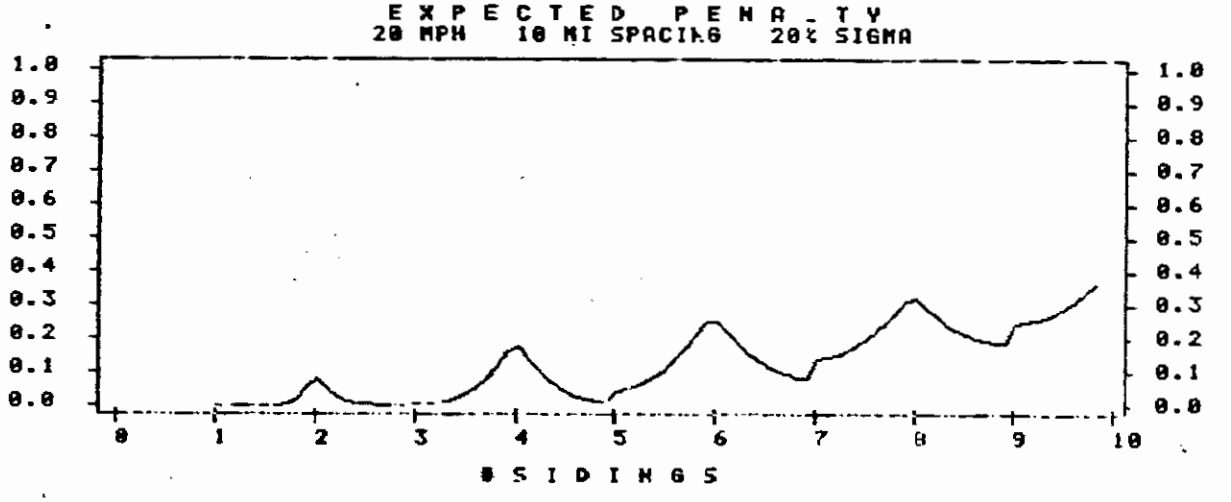
the choice is obvious: meet at the siding in the middle. For this geometric reason, predictive ability actually improves over the interval between two and three passing sidings. Overall, however, predictive ability deteriorates with increasing distance between trains. Performance is worse across five sidings than across three; and it is worse across four sidings than across two.

Because the penalty due to locking in a meet is related to the probability of selecting the proper siding, it too is periodic. The amount of penalty in hours is plotted as a function of the number of sidings in Figures 7a through 7h. The magnitude of the penalty is proportional to the running time between sidings. However, this running time does not affect the "shape" of the curve. Only the level of variance affects this shape.

The performance of meet planning procedures is affected by geometric factors. The deterioriation in their performance is not smooth but is a periodic function, which repeats every other siding.

If a typical rail line has a 27% sigma, 10 mile siding spacing and 20 mph speed, from Figure 4c, the expected delay has converged when the meet is projected across four sidings. Four sidings imply three sections of single track. The running time





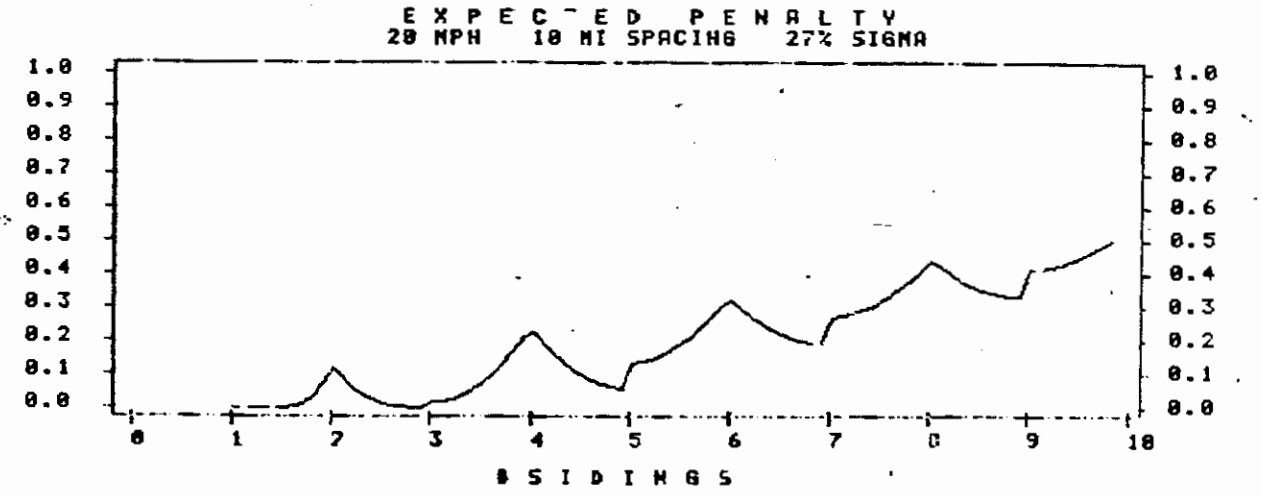
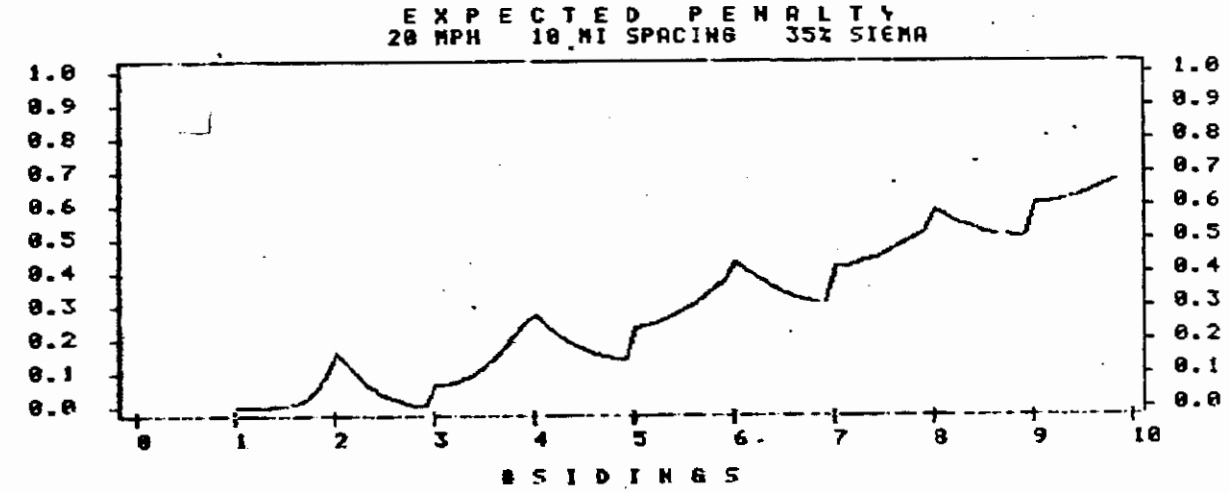
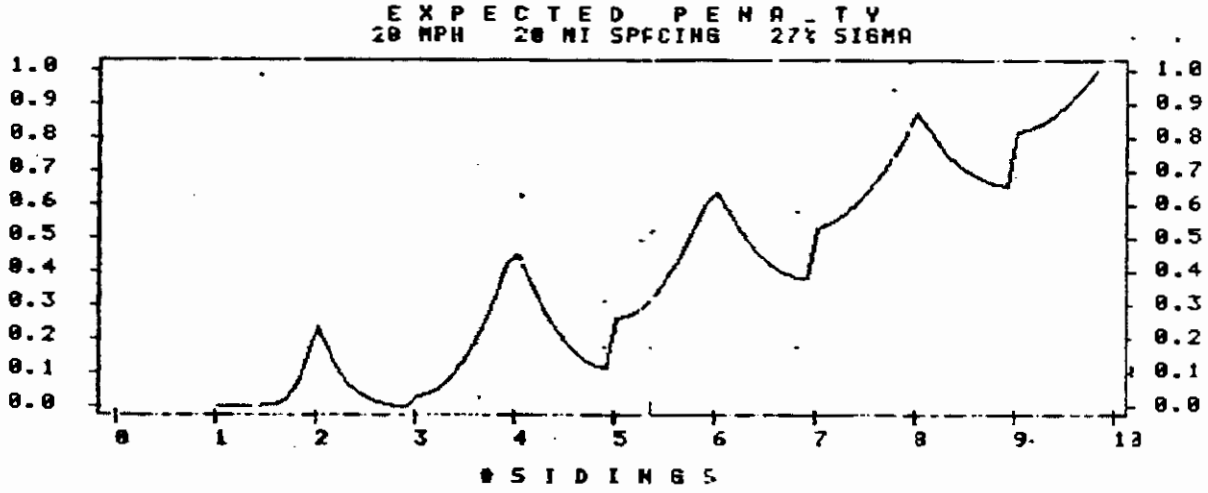


Fig 7a,7b,7c - Expected penalty in hours for locking in a meet as a function of number of sidings

Fig 7d,7e,7f (ctd)





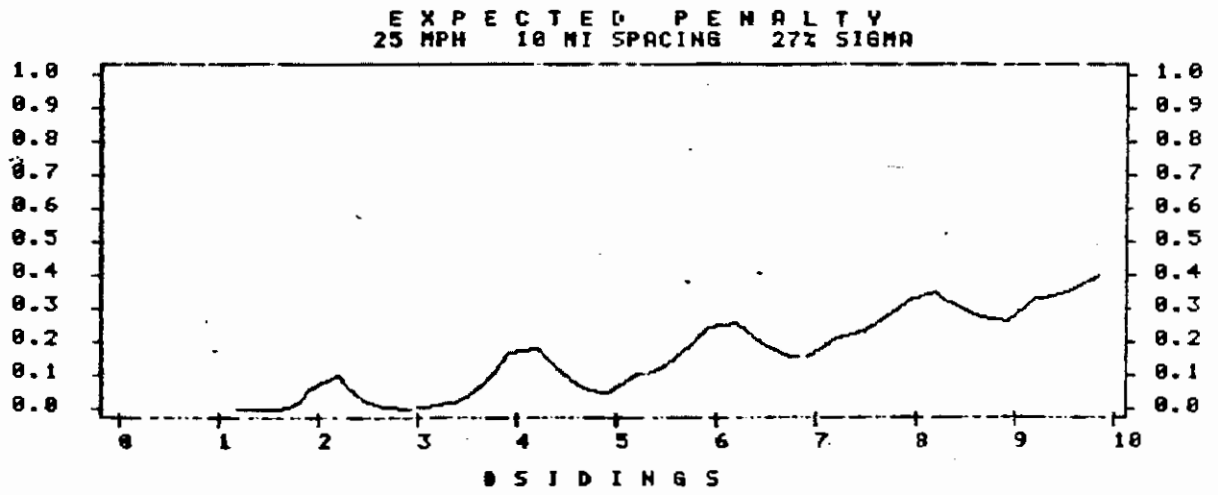
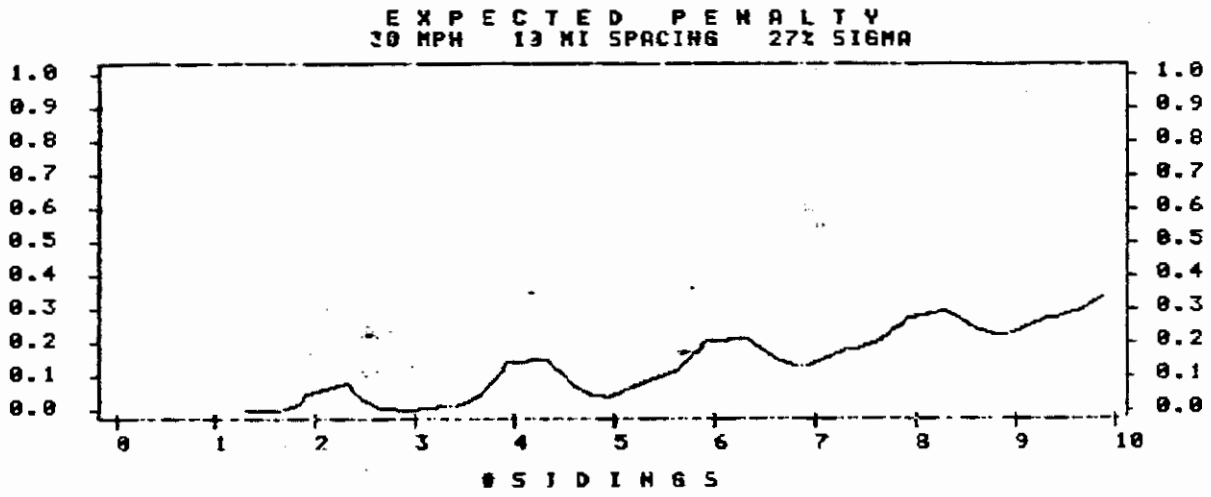
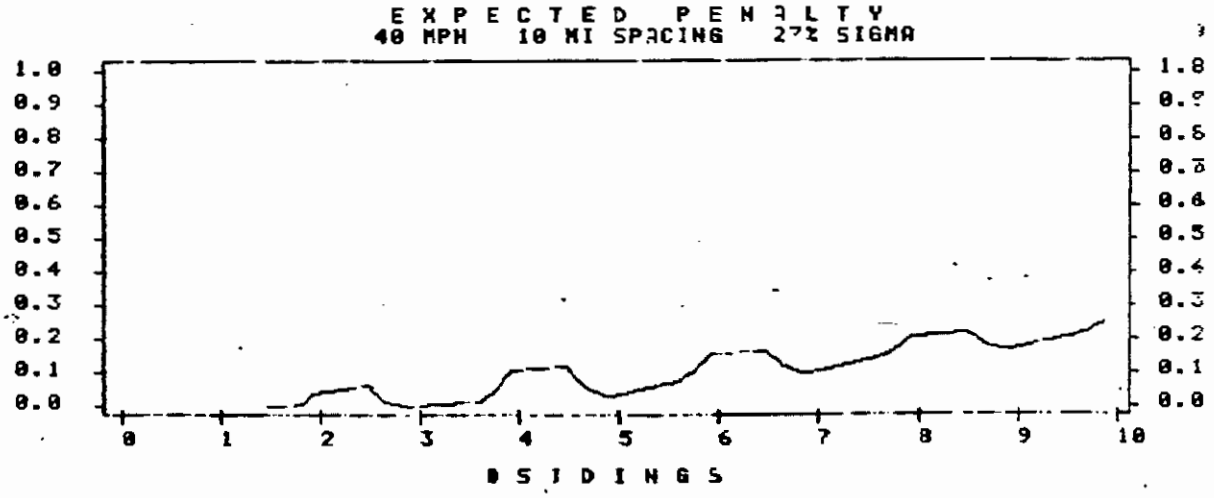


Fig 7g,7h (ctd)


EXPECTED PENALTY
30 MPH 13 MI SPACING 27% SIGMA
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10
SIDINGS


EXPECTED PENALTY
40 MPH 10 MI SPACING 27% SIGMA
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10
SIDINGS

across these three section is only 90 minutes. The practical planning horizon is half of this (since the meet would occur in the middle) or only 45 minutes.

This short planning horizon places severe limitations on the ability of the dispatcher, or a computerized algorithm, to plan meets. Improved locomotive engineer training, or onboard speed regulators which would reduce the level of variability would lengthen this planning horizon and enable a reduction of total train delay.

5. Global Optimization

The algorithm developed here solves the generalized train dispatching problem described in the introduction. It minimizes delay in multiple train conflicts, enables explicit consideration of schedule and 12-hour constraints, and allows anticipation of delay waiting to be received into destination rail yards.

A computer program implementing this algorithm could be used to help dispatchers plan better meets. In such an application, a new meet plan would be generated whenever the need for a decision arises. For example, whenever a train approaches the entrance switch to a passing siding, a new plan would be generated, based on the most current information, to see if that train should be diverted into the siding.

A "snapshot" of the current status of the line would be taken, and all possible train meet strategies would be projected. Then, based on the optimal train meet strategy, the current decision (e.g., whether to divert the train into the siding) would be made.

The algorithm presented here resolves train meets on a single track rail line with passing sidings. Multiple track conflict resolution with overtakes and passing are subjects for future research.

5.1. Instruction Lists and Repeated Simulation

As long as multiple train interactions are rare, local optimization yields a minimum delay solution. However, when multiple train interaction becomes common, local optimization fails, because the impact of decisions on later meets and on the formation of fleets of trains are not considered. Therefore, local optimization no longer necessarily produces the minimum delay dispatching plan.

As the number of meets occurring within the defined planning horizon increases, the number of possible meeting strategies increases exponentially. Two sidings, on either side of the section of single track where the conflict occurs, are potential meeting locations. Because there are two possible resolutions for each conflict, there are 2^n possibile meet strategies, where n is the number of meets to be planned.

The number of train meets, n, is not the same as the number of trains. For example, if trains A and B eastbound are to meet train C westbound, there are three trains, but only two meets: A meeting C, and B meeting C. If there are two westbounds, C and D, then there are four meets: A meeting C, A meeting D, B meeting C and B meeting D. In general, the number of train meets depends not only upon the number of trains

on the railroad but also upon those trains' directions and relative locations.

Some of the meet strategies out of the 2^n theoretically possible may be physically infeasible. For example, one strategy may call for all four meets described above to occur in the same passing siding. This would not be feasible unless the siding had the capacity to hold at least two trains. Exclusion of any physically infeasible strategies may reduce the number of strategies to less than 2^n.

Selecting a siding not adjacent to the conflict area would result in increased delay to one train without any possible savings to the other; therefore, it cannot be an optimum solution. (One train already has zero delay.) It is possible in some cases to have three reasonable meet locations, if the meet in the central location is closely timed, such that both trains would be delayed. Should it be desirable to ultimately delay one of the trains due to another conflict, that delay can be postponed and occur in the downstream siding, as part of that conflict.

From the probability model of the section on local optimization, the expected number of meets increases proportional to the square of volume.

Define: n= Expected number of meets

S= Expected number of strategies

Recall:

Total daily delay = N^2R^2 / 24 (10)

= Expected Number of Meets x

Average Delay per Meet

The average delay per meet is a constant, R/2.

Therefore:

$n = N^2R$ / 12 or

$k = R/12$, $M = kn^2$ (20)

To find how the expected number of strategies is related to volume, substitute into:

$S = 2^n$ (21)

to yield:

$S = 2^{kN^2}$ (22)

The optimal meet strategy can be generated by simulating all 2^n possibilities. All possibilities can be defined by a set of lists of train meeting instructions. One possible instruction list might say: resolve the first two conflicts in favor of the eastbound train, the third conflict in favor of the westbound, and the fourth in favor of the eastbound. If direction 1 is east and direction 2 is west, this meet pattern is specified as 1121. All lists are generated in advance by listing all combinations of 1's and 2's. For example, for four meets, the following instruction lists define all the possibilities: 1111, 1112, 1121, 1122, 1211, 1212,

1221, 1222, 2111, 2112, 2121, 2122, 2211, 2212, 2221, 2222.

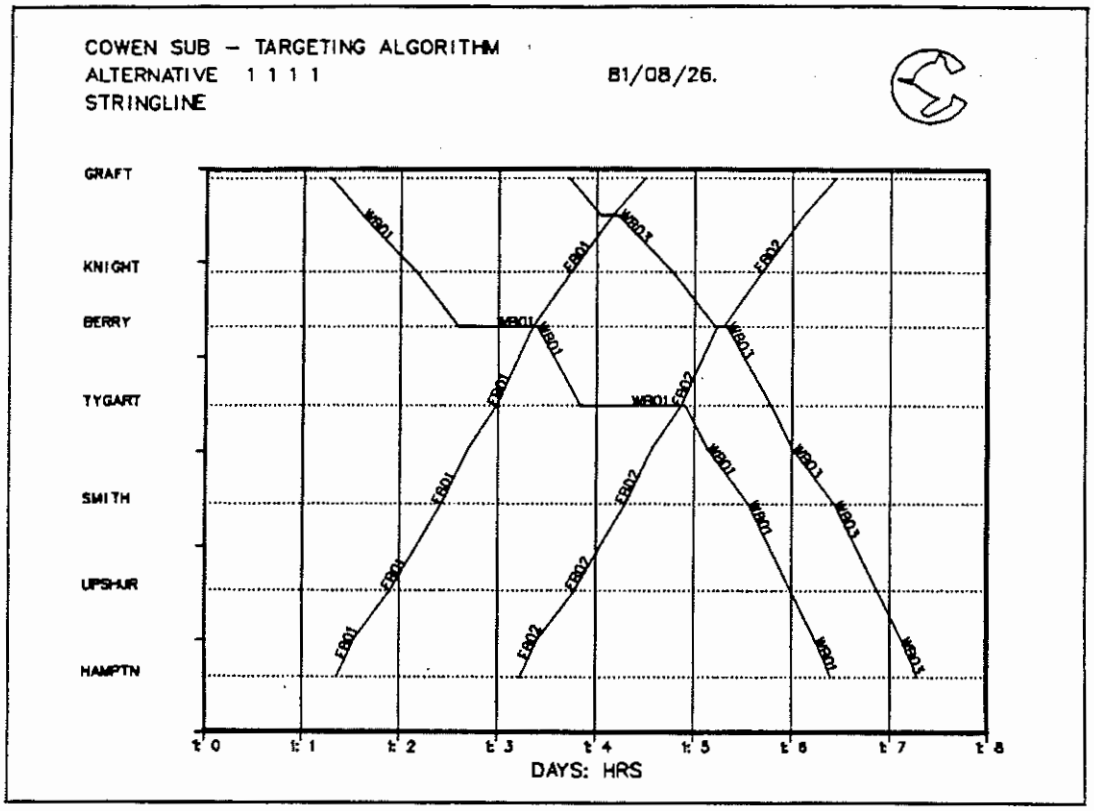
If the simulation program can follow the instructions contained in a list, it may be rerun with each different instruction list, thus simulating all possible meet resolution plans. For one case, this has been done. The results are shown in Figures 8a through 8q, showing all sixteen possible dispatching plans.

In Figure 8a, notice that alternative 1111 gives eastbounds absolute priority; in Fig. 8p, alternative 2222 gives westbounds absolute priority. If the first meet is resolved in favor of EB01, then the second conflict is between EB01 and WB03, but if the first meet favors WB01, then the second conflict is between WB01 and EB02. In other words, the items on the instruction list do not refer to specific meets, but only to whatever conflict occurs next. Finally, in this example the global optimum is 2211, which happens to be the same as the local optimum (fig 8q).

5.2. Branching Procedure

The branching procedure reduces the required computer time by saving intermediate results and by terminating the examination of alternatives as soon as it becomes obvious they cannot be optimal. A branch is an intermediate simulation result. When a conflict occurs, two branches can be created by resolving the

Fig 8a & 8b


COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 1 1 1 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
WB01
WB03
EB01
EB02
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

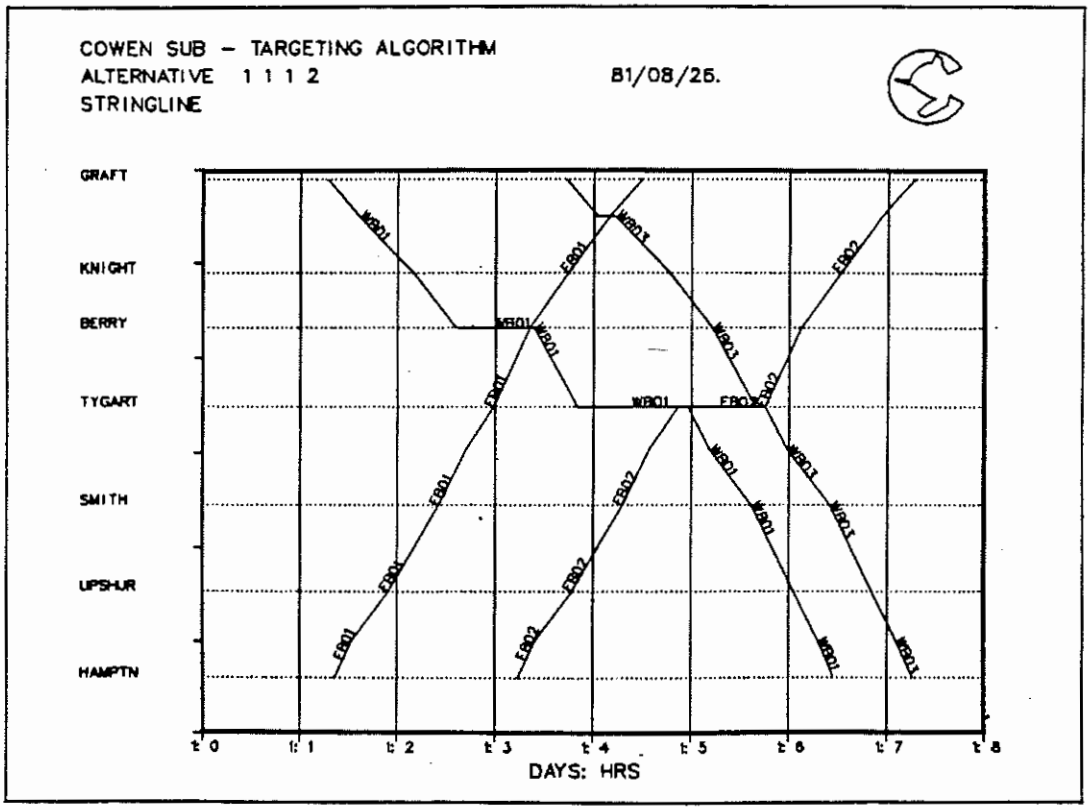
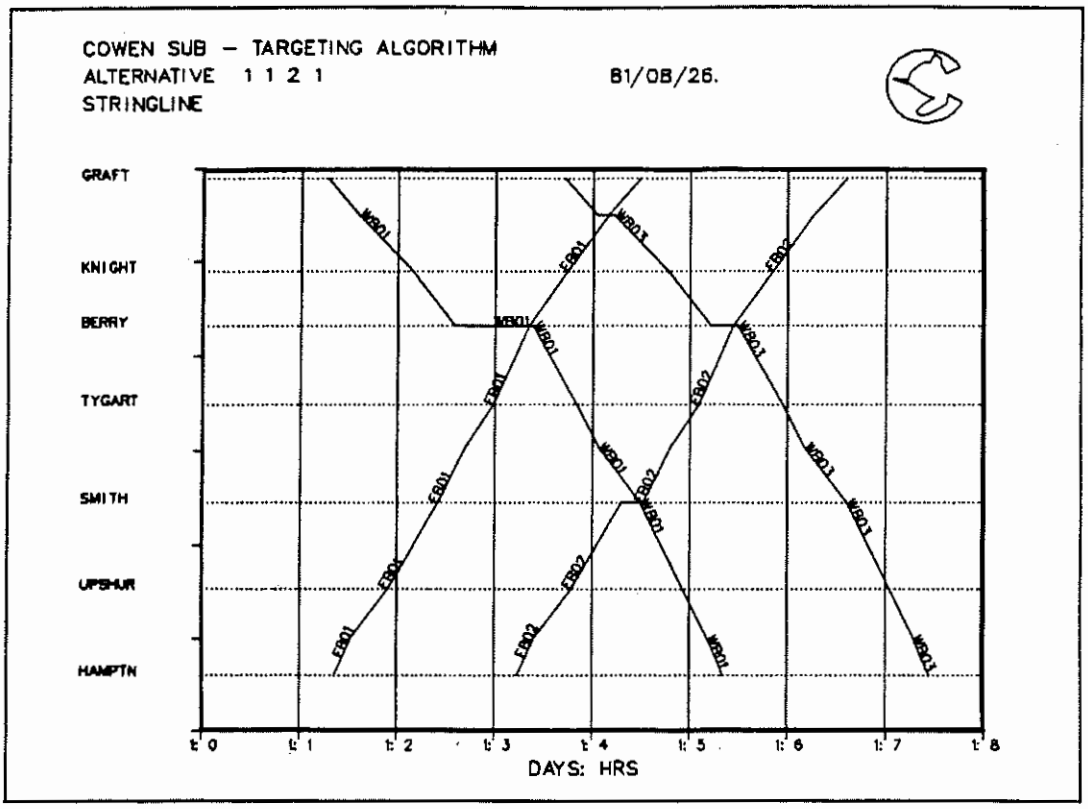

COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 1 1 1 2
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
WB01
WB03
EB01
EB02
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8c & 8d


COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 1 1 2 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

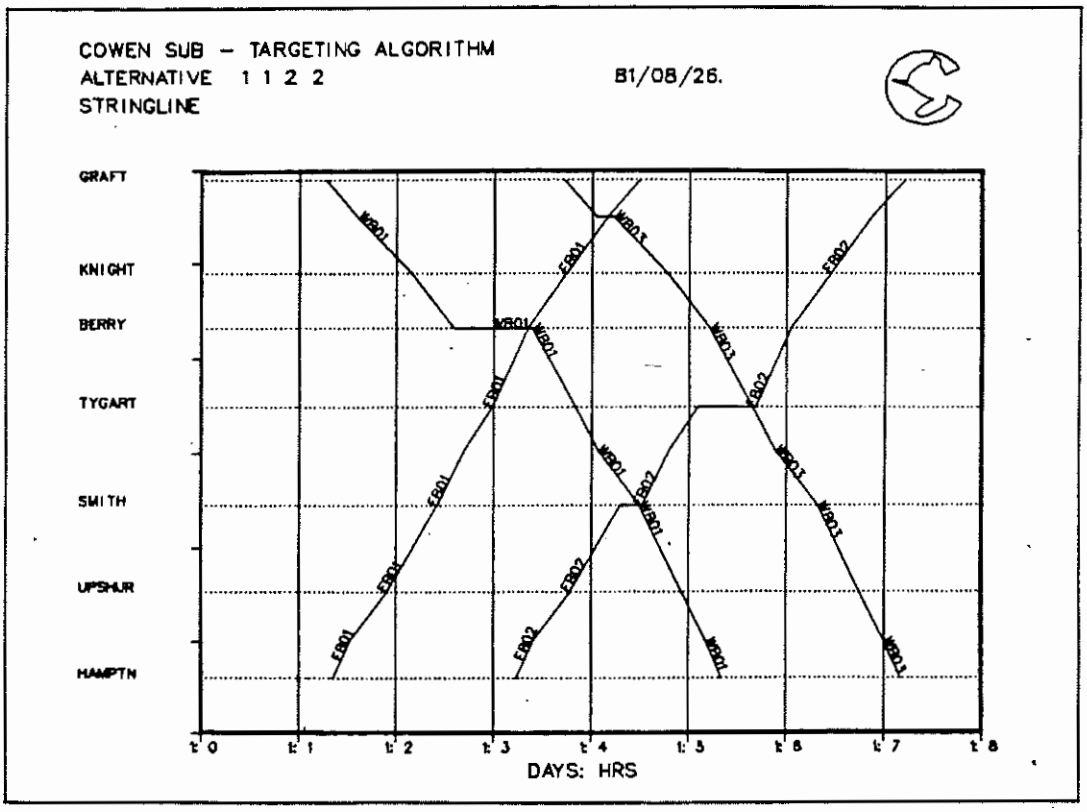
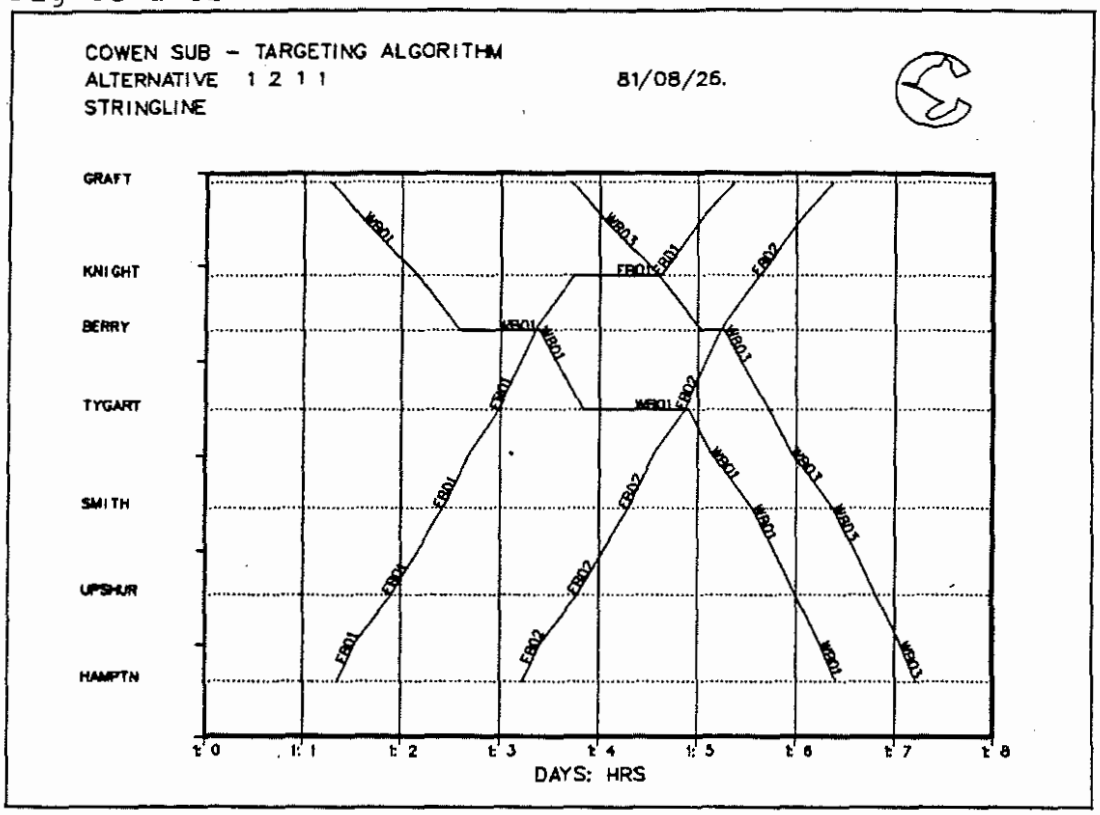

COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 1 1 2 2
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8e & 8f


COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 1 2 1 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

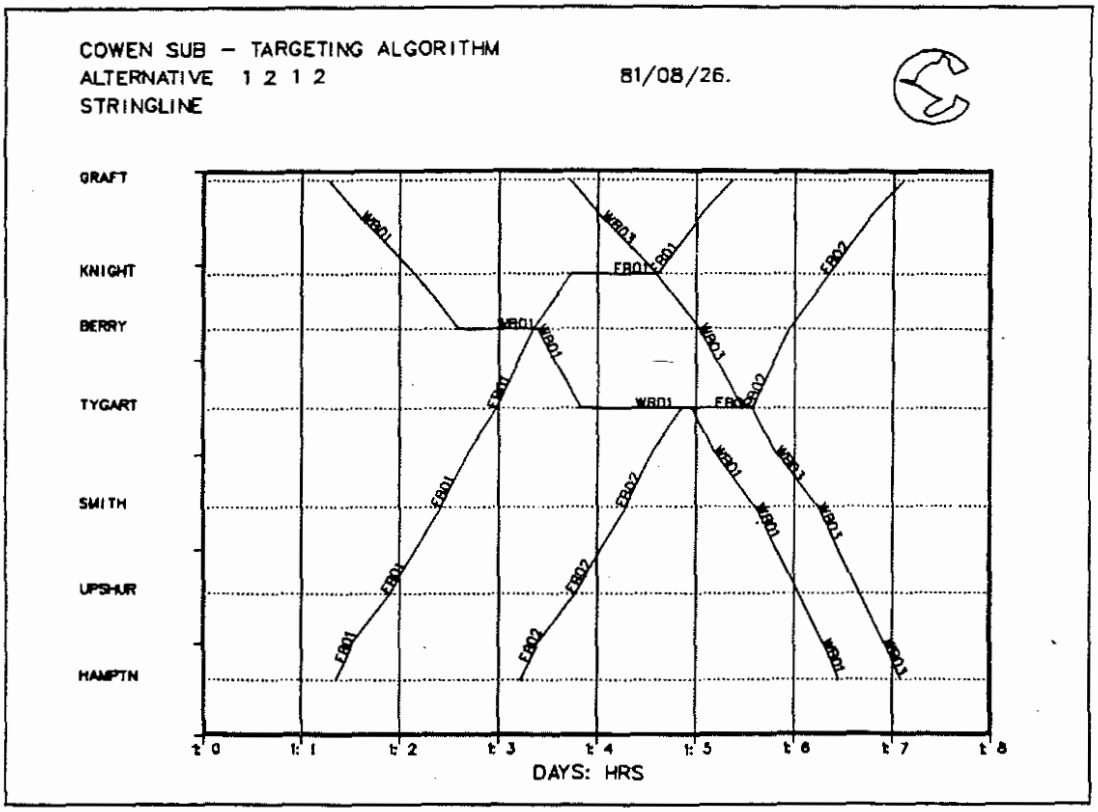
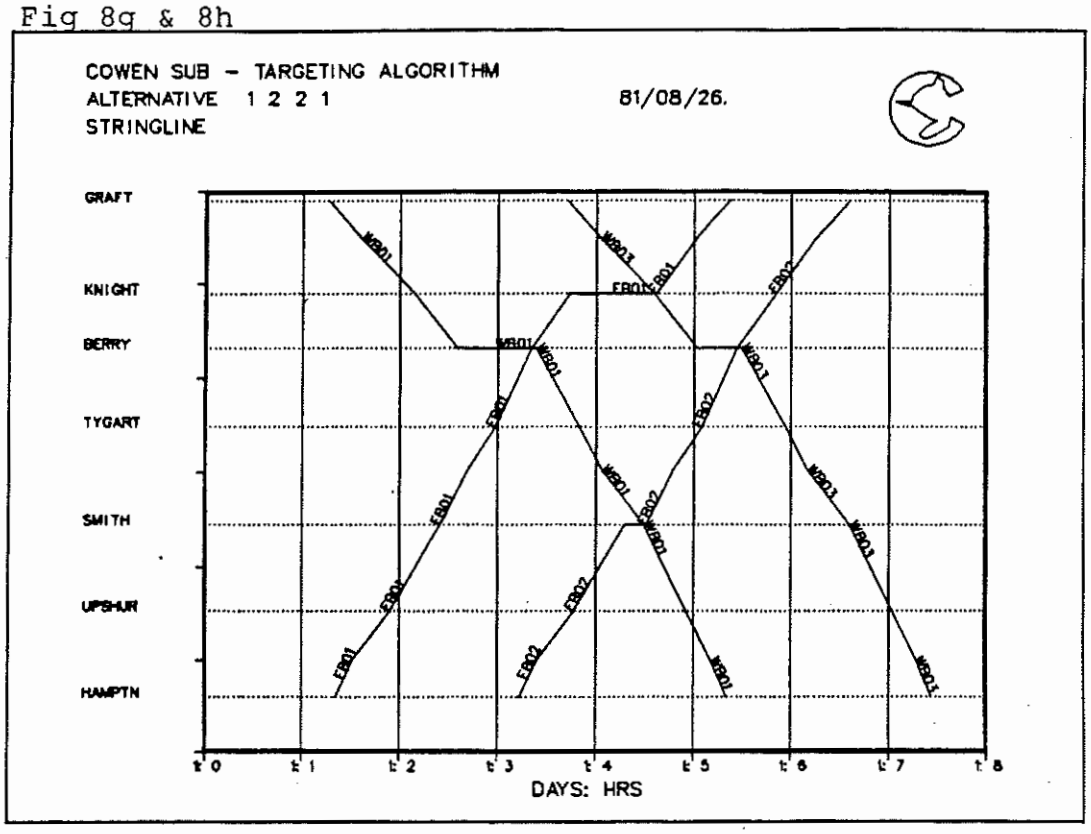

COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 1 2 1 2
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8g & 8h


COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 1 2 2 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

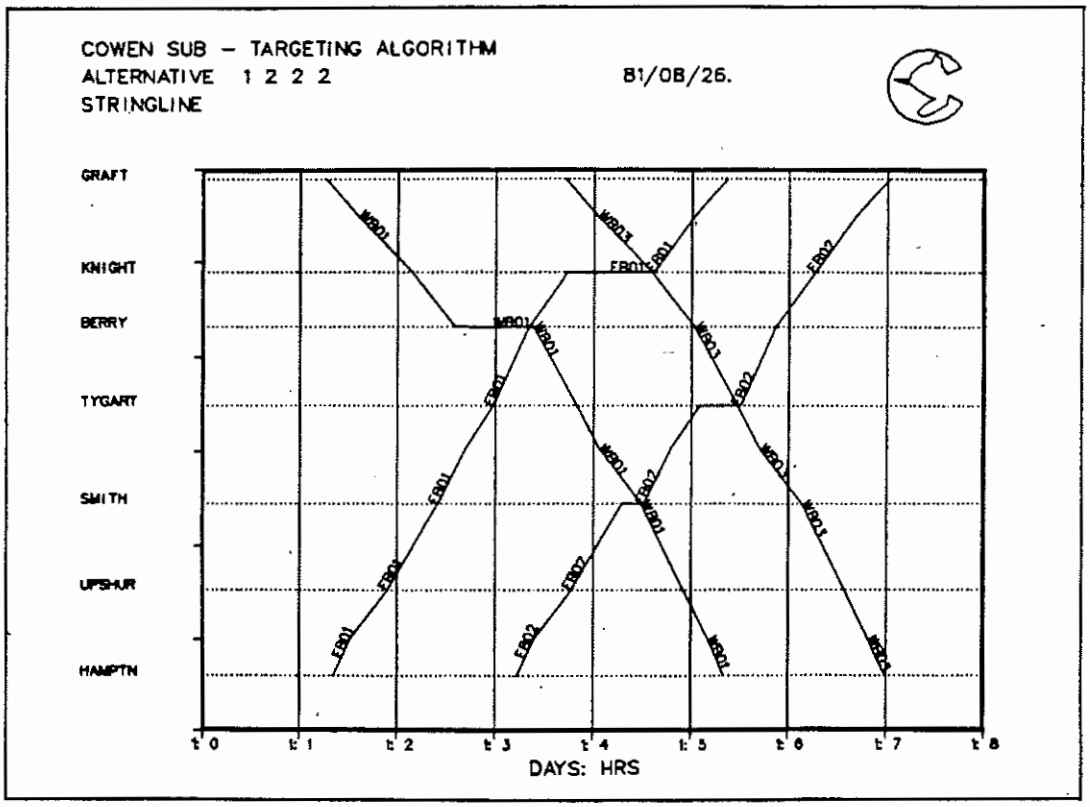
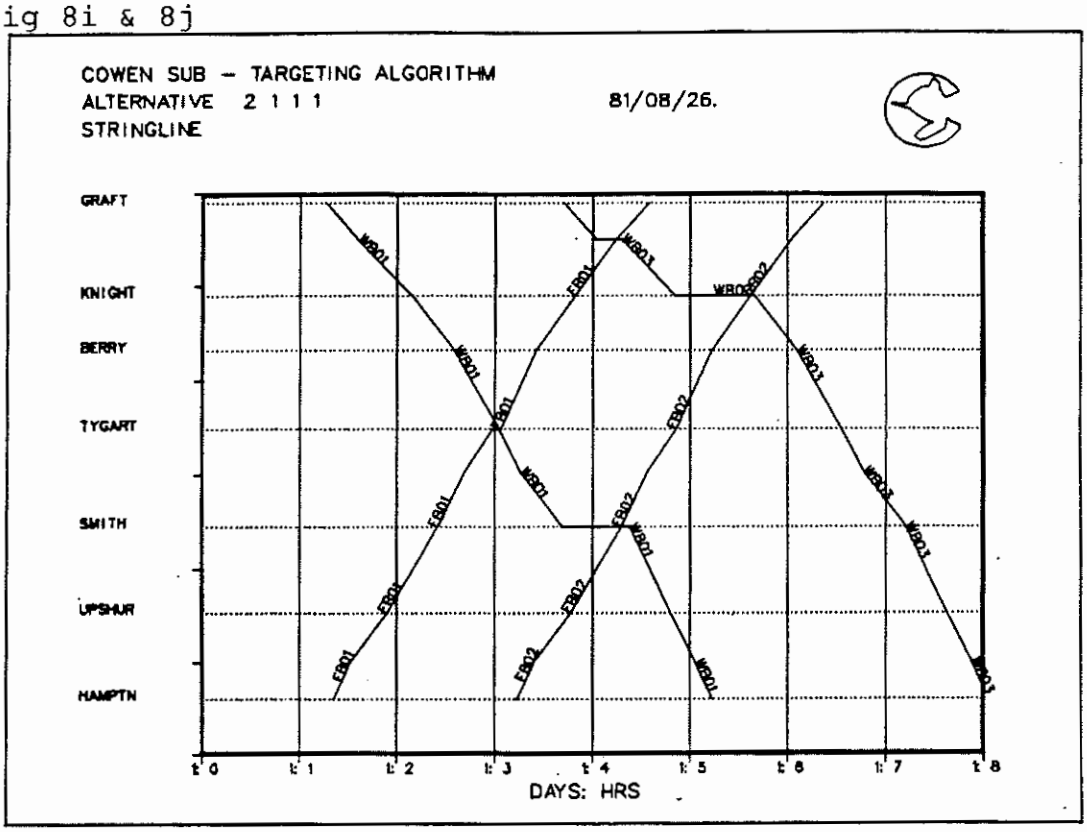

COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 1 2 2 2
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8i & 8j


COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 2 1 1 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

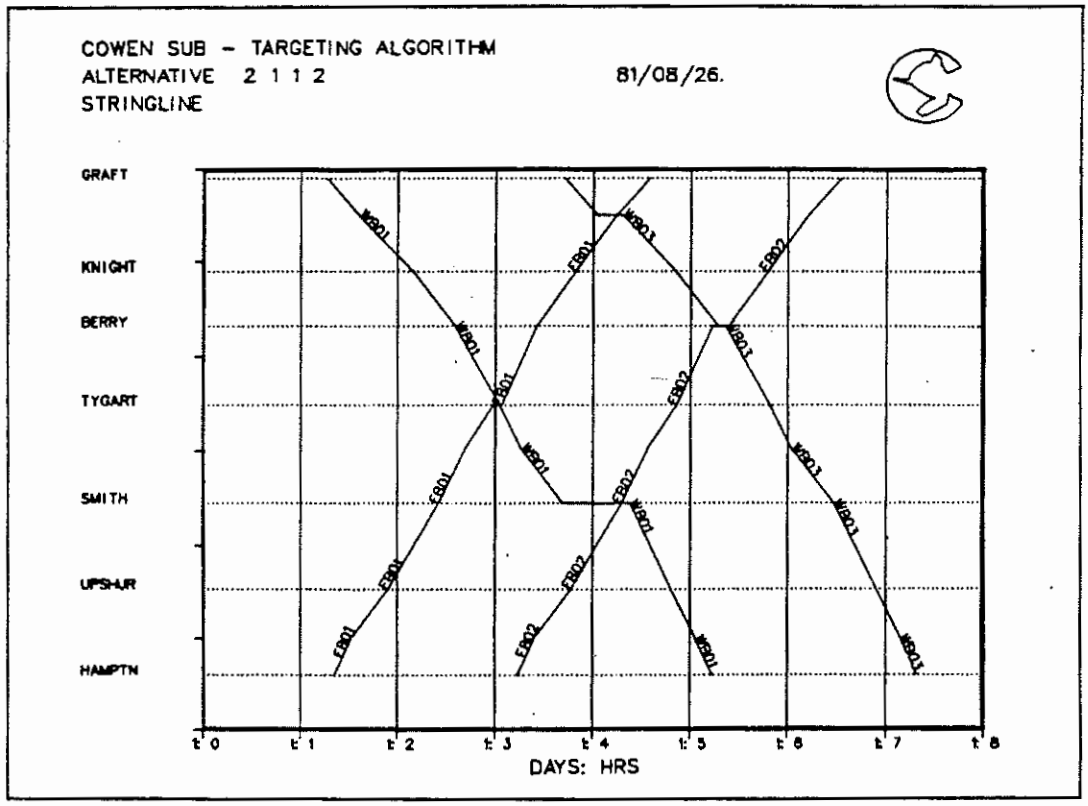
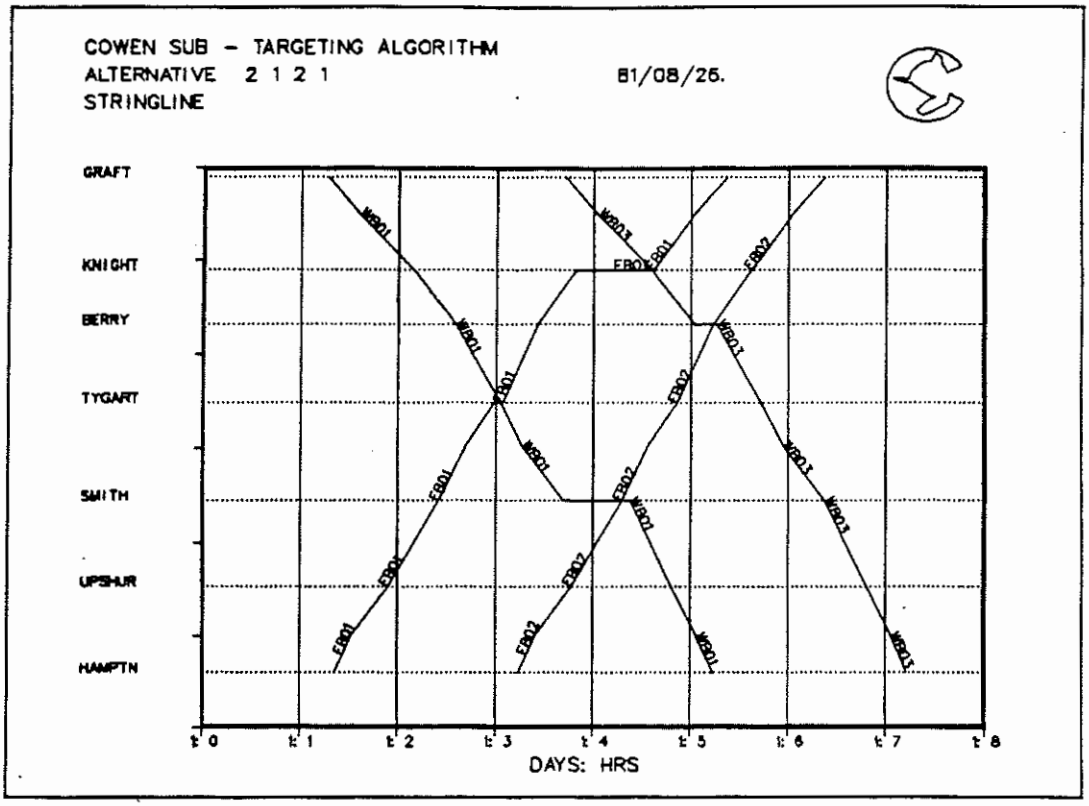

COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 2 1 1 2
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8k & 81


COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 2 1 2 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

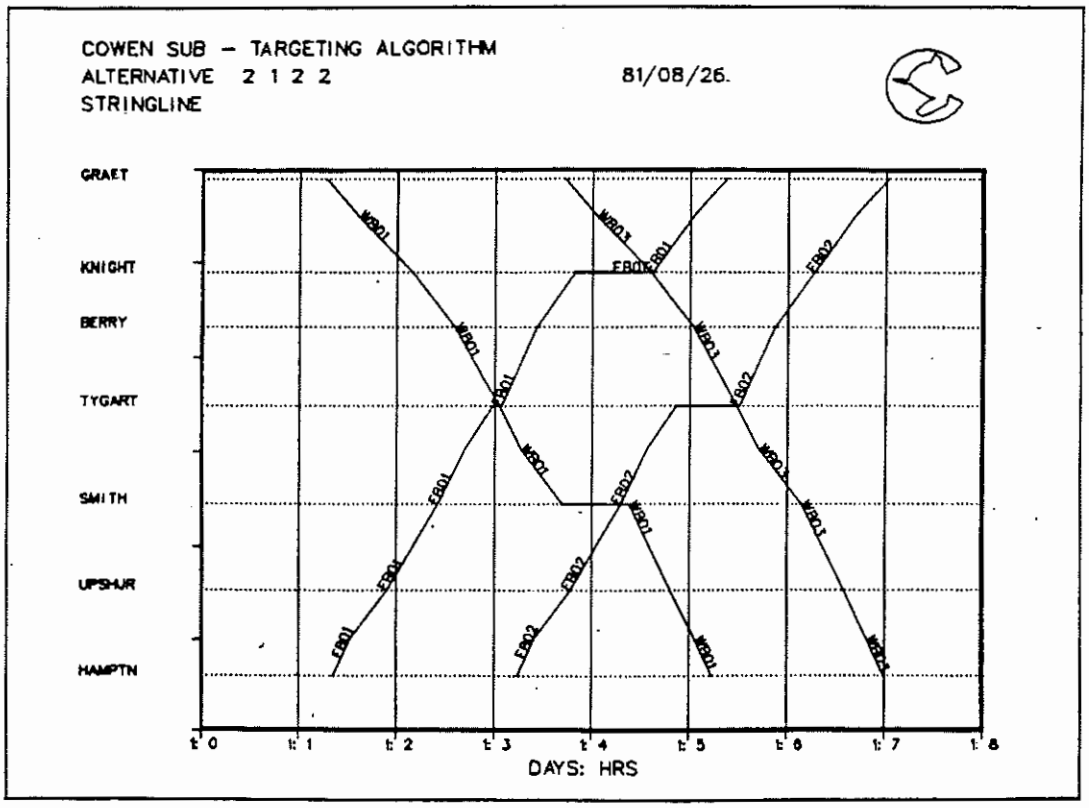
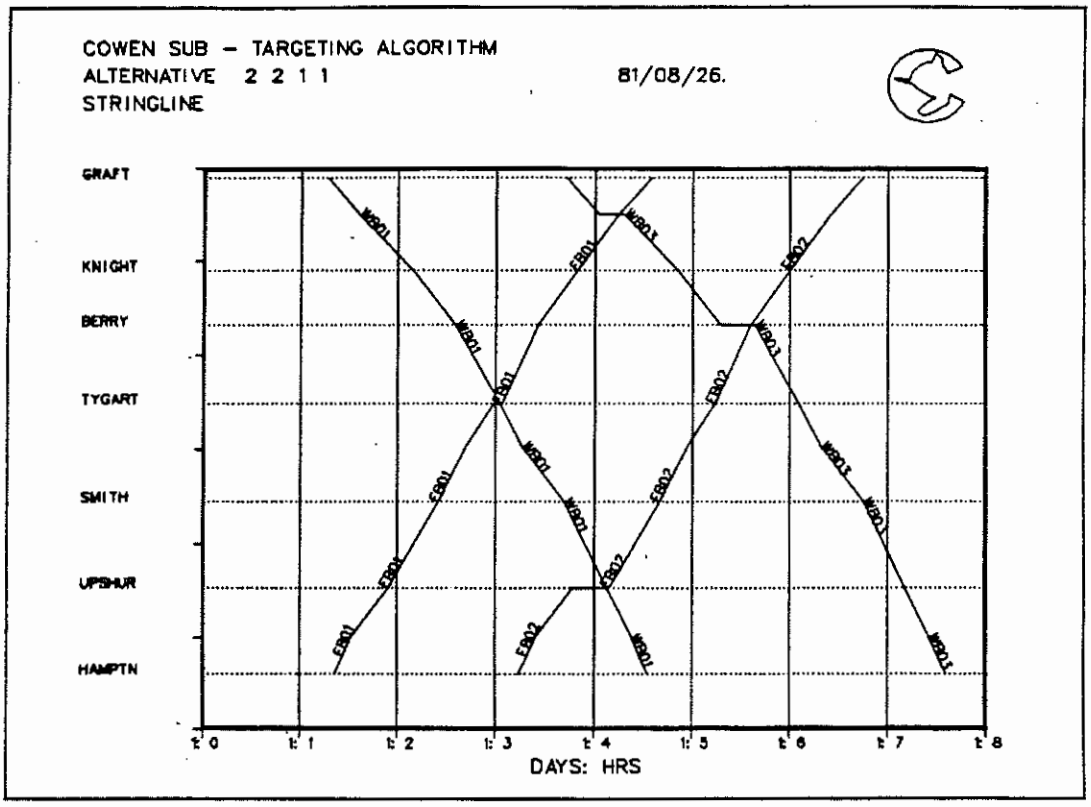

COWEN SUB - TARGETING ALGORITHM
ALTERNATIVE 2 1 2 2
STRINGLINE
81/08/26.
GRAET
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8m & 8n


COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 2 2 1 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
WB01
EB01
EB02
WB03
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

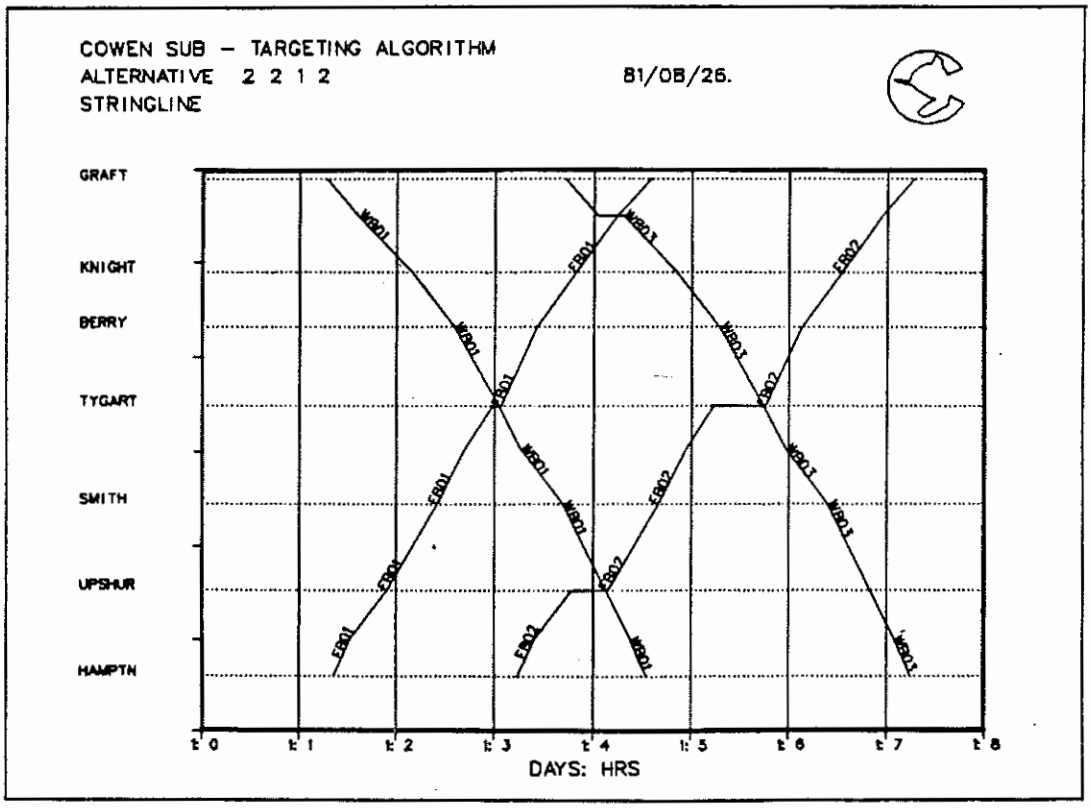
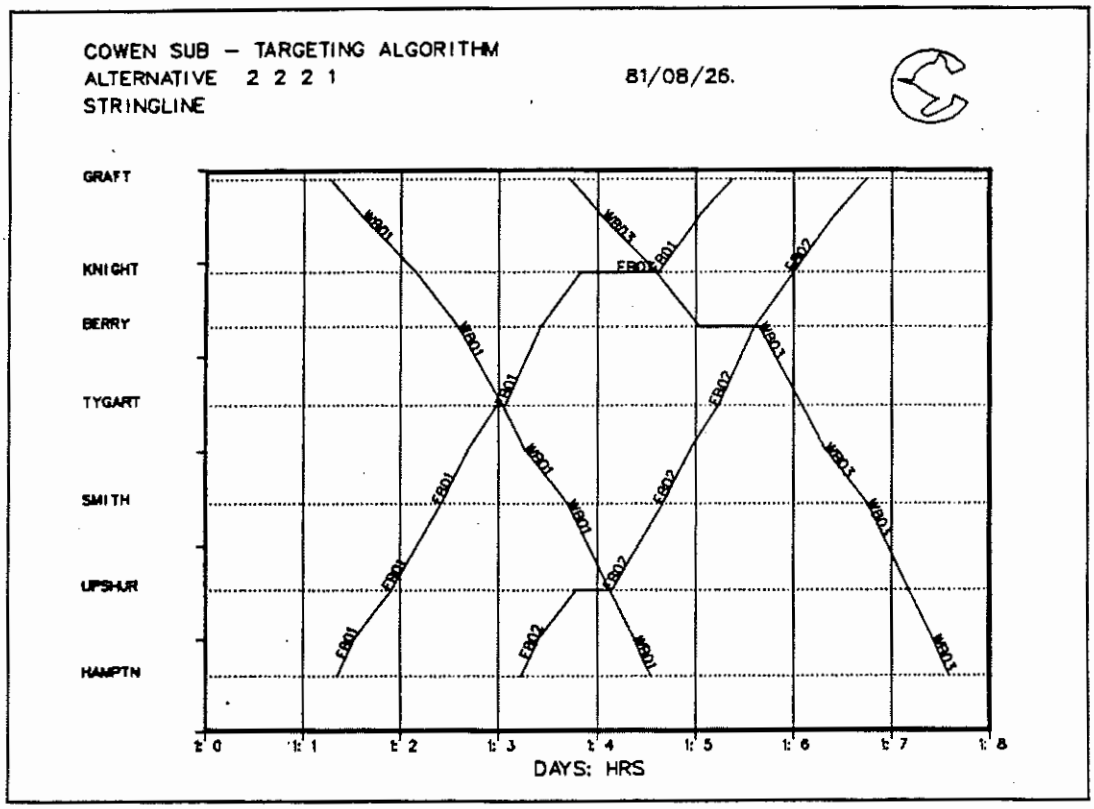

COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 2 2 1 2
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
WB01
EB01
EB02
WB03
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8o & 8p


COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 2 2 2 1
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

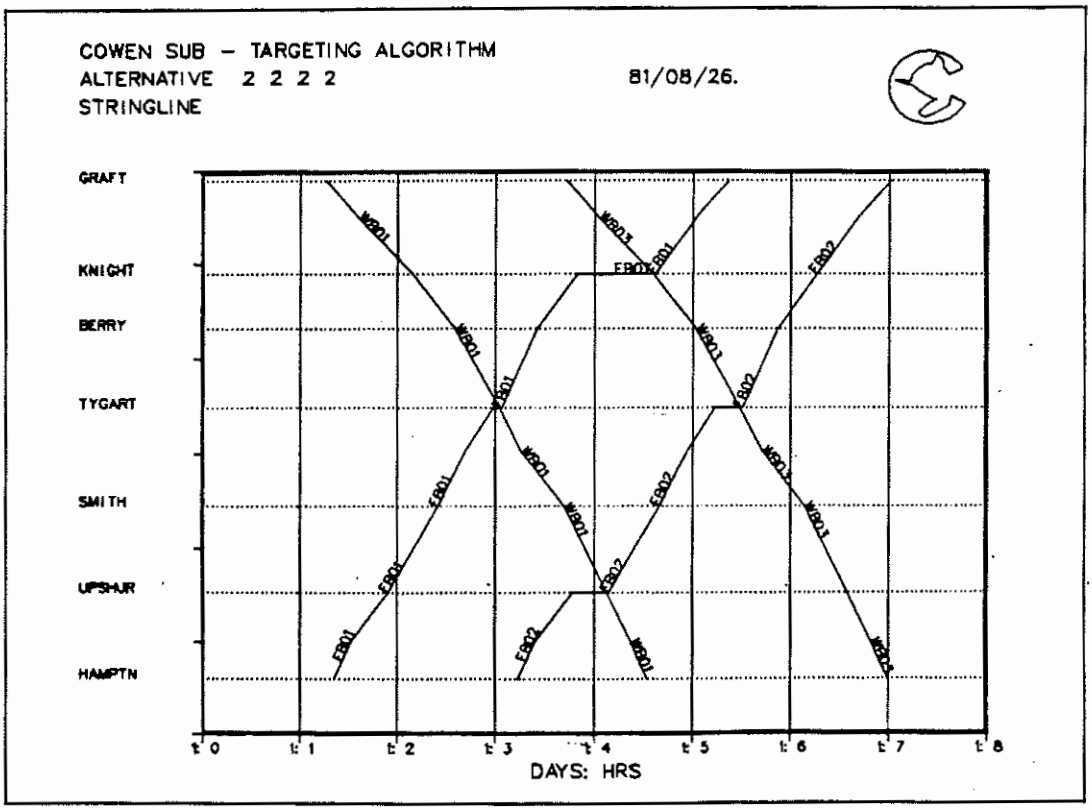
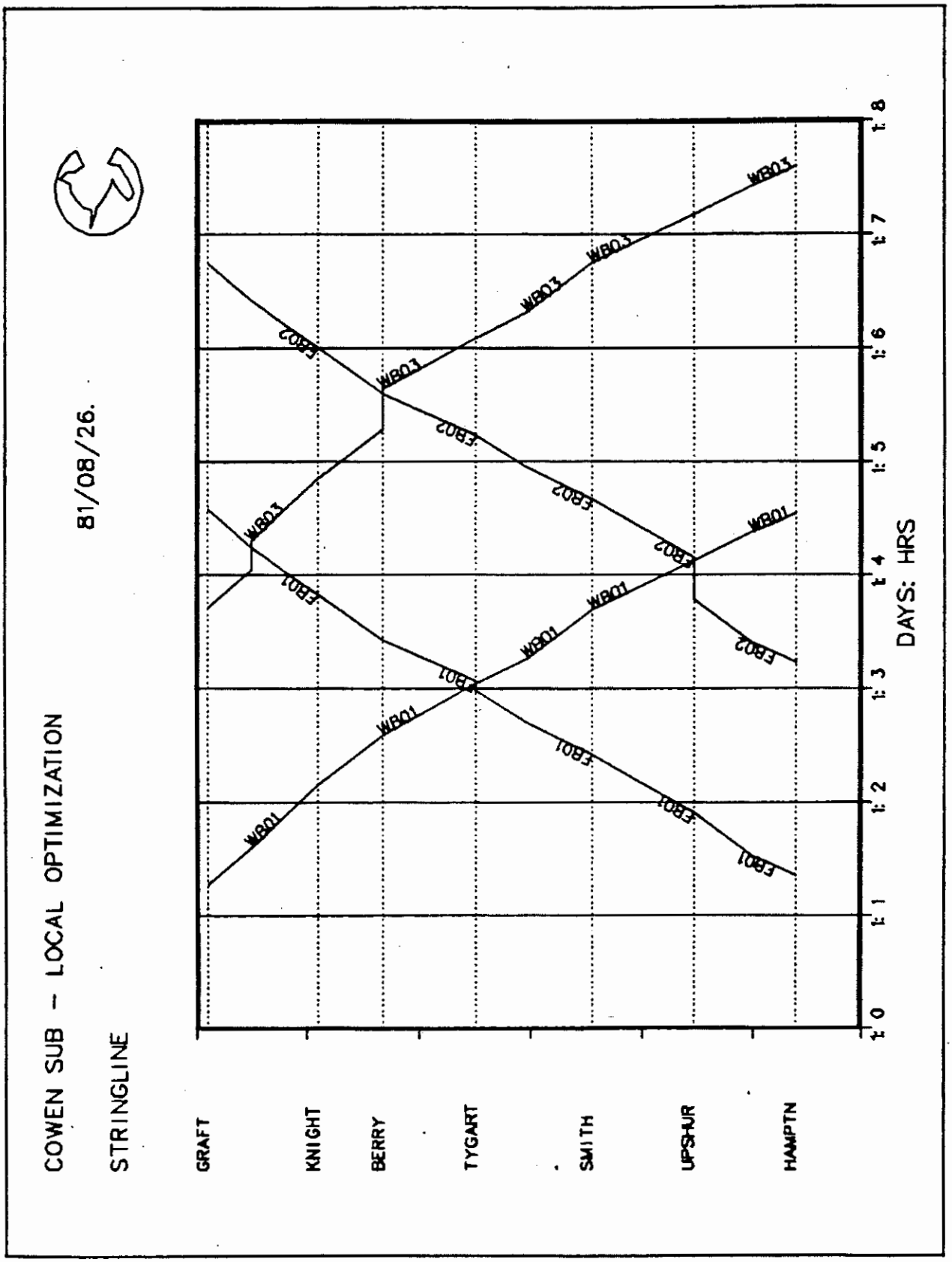

COWEN SUB – TARGETING ALGORITHM
ALTERNATIVE 2 2 2 2
STRINGLINE
81/08/26.
GRAFT
KNIGHT
BERRY
TYGART
SMITH
UPSHUR
HAMPTN
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7
t 8
DAYS: HRS

Fig 8q - Locally optimized solution



conflict both possible ways. Then the branches are saved and the most promising solution identified for further examination.

The previous method, where all possible strategies are simulated from the beginning, is wasteful. For example, alternatives 1121 and 1122 are identical up to the last meet. Rather than starting from the beginning of the simulation each time, the intermediate status of the system can be saved after the third meet, so that the duplicate portion of the simulation need not be repeated. The problem of Figure 8 has been solved using the branching algorithm. The process of solution is graphically explained by Figure 9. The procedure is:

1. Advance trains from siding to siding until a conflict occurs. In Fig 9, the conflict is shown in dashed lines. If no conflicts occur before all the trains terminate, the branch is complete. Any other branches with greater delay than the completed alternative cannot be optimal.

2. Branch by resolving the conflict arbitrarily in favor of each train. This gives two possible meet locations. Add the delay caused by each new meet to the delay already incurred in all previous meets. If this delay is greater than a completed alternative, the present branch cannot be optimal, so eliminate it

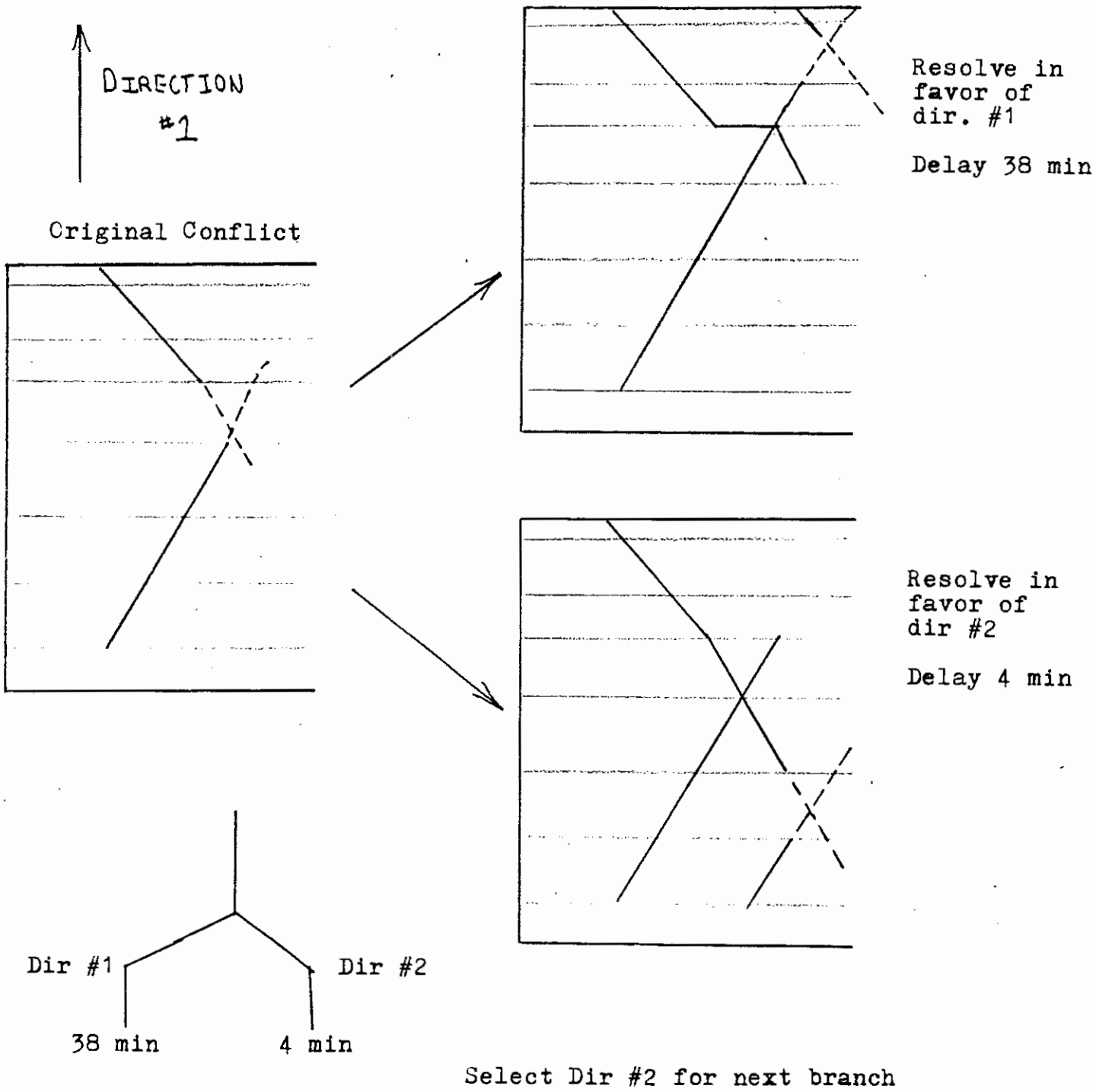
from further consideration. Or, if the selection of the branch would make it impossible to meet a schedule or 12 hour constraint, eliminate the branch.

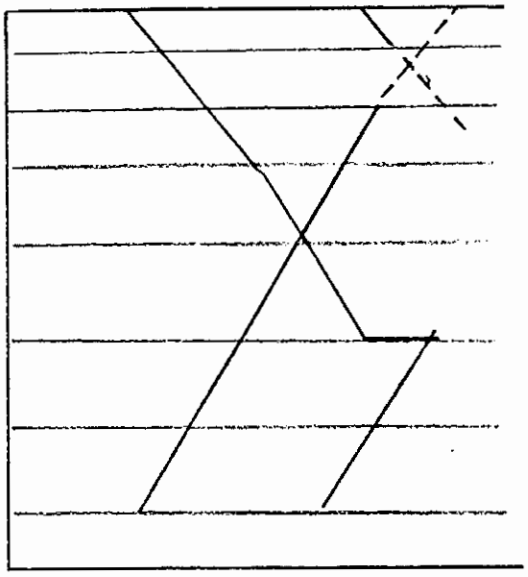
3. Search for the feasible branch having minimum total delay. Using this branch, return to #1. If no incomplete branches remain, the completed branch having minimum delay is globally optimal.

Figure 9 shows the process of determining an optimal train meet strategy on a single tracked line. However, there is no reason why the algorithm cannot be used to determine train pass strategies as well on single or multiple track lines. The appropriate recognition of conflict situations and the resolution of each conflict in each possible way is all that is required.

Each page in Figure 9 shows a single step. A tree diagram shows the status of the current solution at each step. In the time-distance diagrams, the solid lines indicate the resolved conflict areas. Dashed lines indicate unresolved conflicts between sidings. A branch is formed by resolving the dashed-line conflicts in both possible ways. Delay is determined by measuring the length of the horizontal line on the time-distance chart, and is summed for each alternative. The total cumulative delays are shown on the tree diagram at each stage. The solution procedure demonstrated in Figure 9 is outlined above.

Fig 9 - Branching Procedure For Optimization

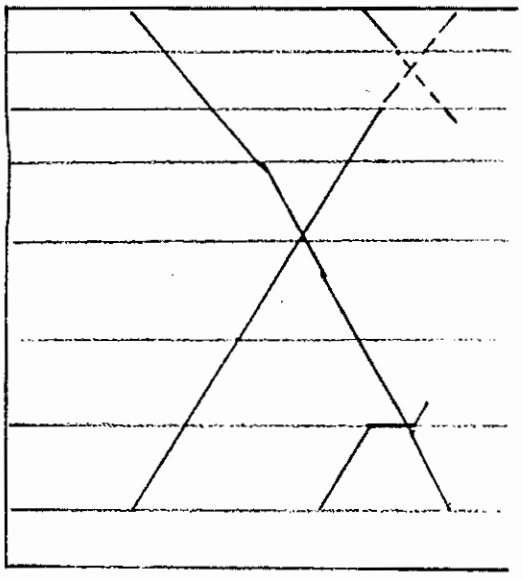




Resolve in favor of dir #1

Delay = 4 min + 34 min

= 38 min

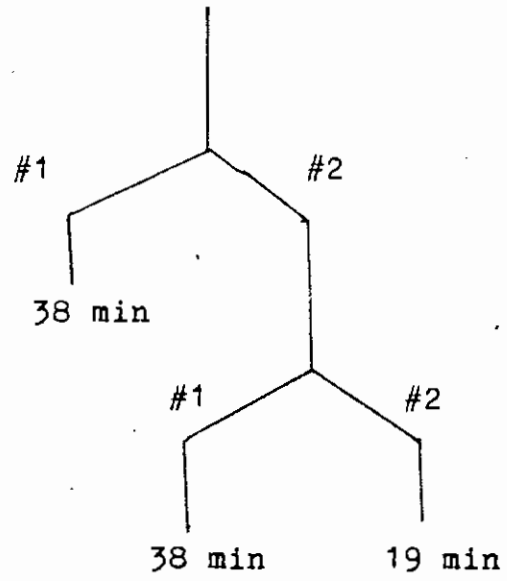


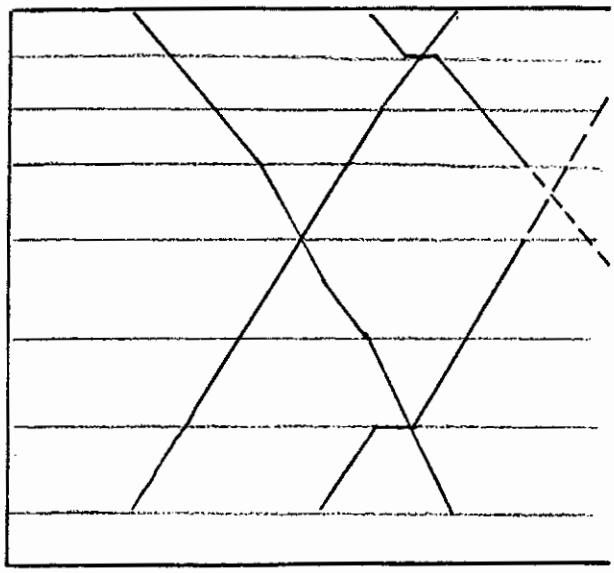
Resolve in favor of dir #2

Delay = 4 min + 15 min

= 19 min

Select Dir #2 for next branch

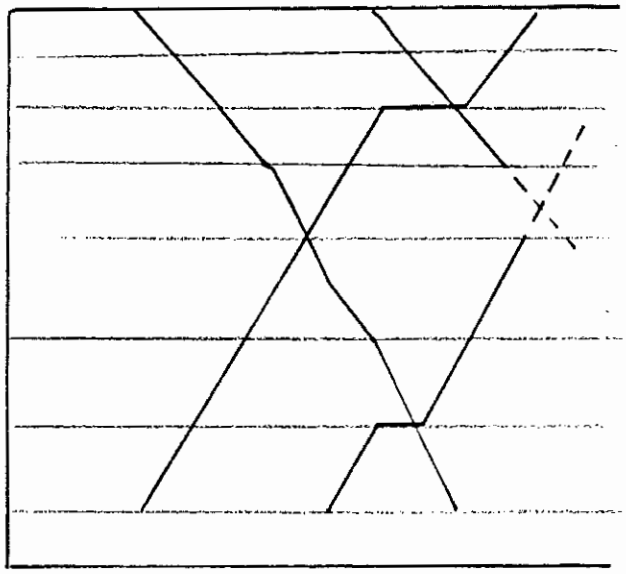




Resolve in favor
of dir #1

Delay = 19 min + 11 min

= 30 min

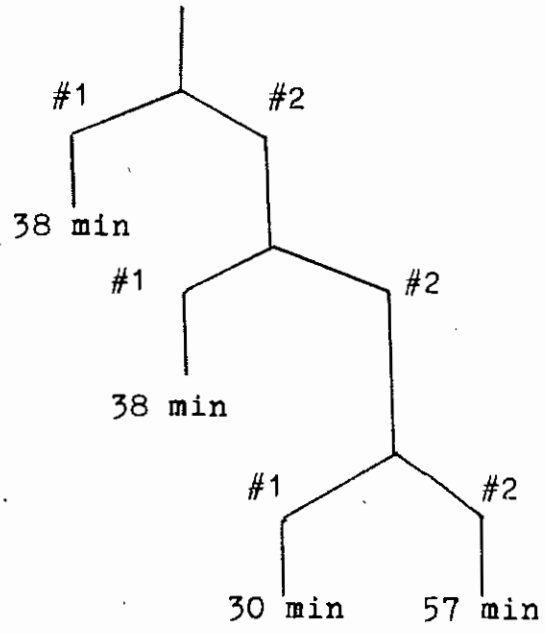


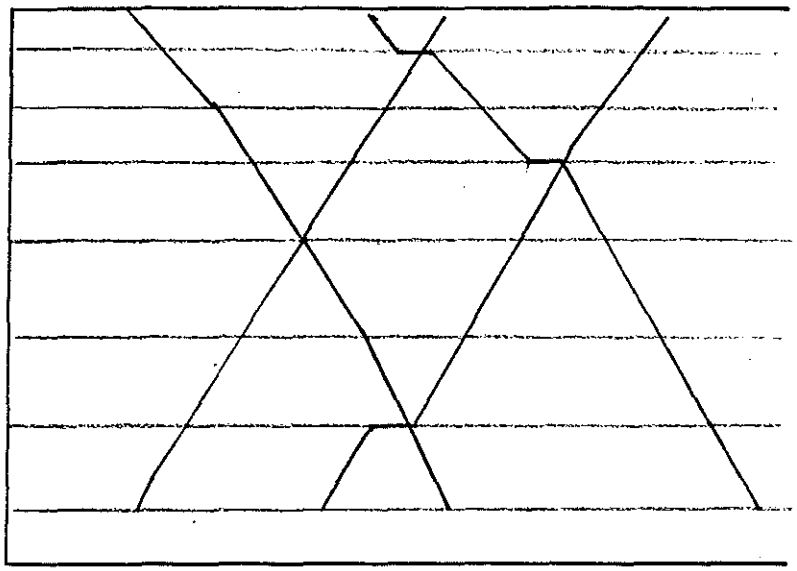
Resolve in favor
of dir #2

Delay = 19 min + 38 min

= 57 min

Select Dir #1 for
next branch



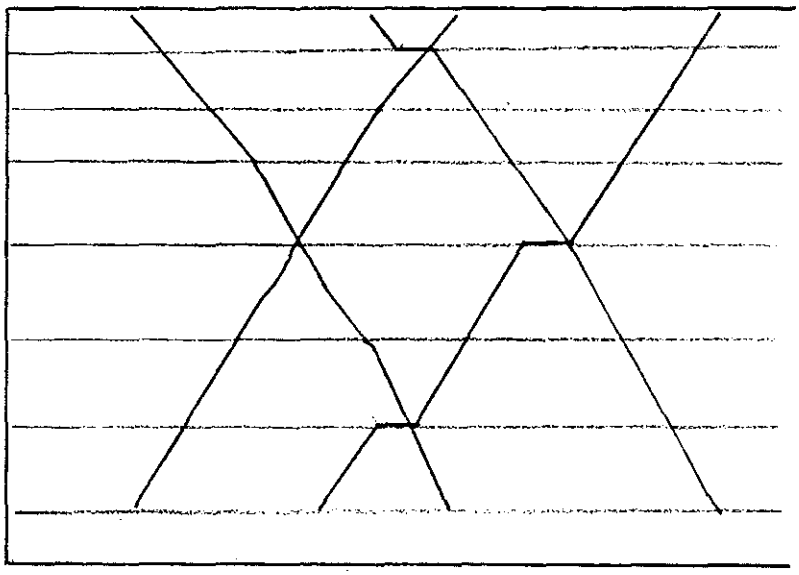


Resolve in favor of dir #1

Delay = 30 min + 15 min

= 45 min

COMPLETE



Resolve in favor of dir #2

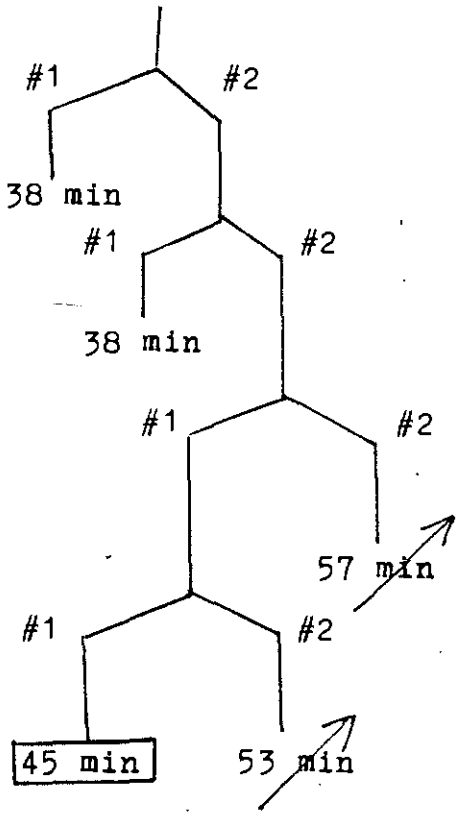
Delay = 30 min + 23 min

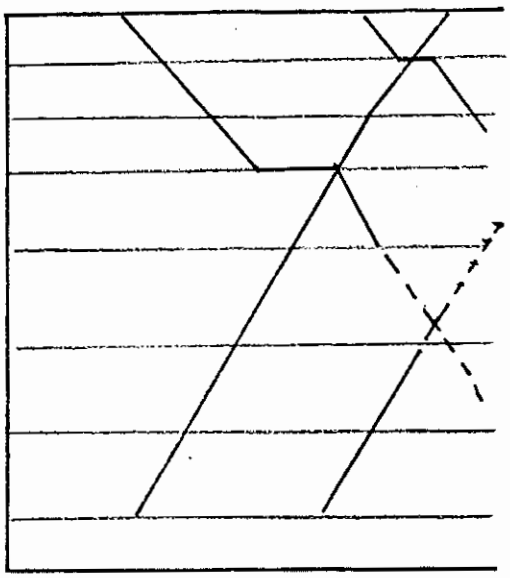
= 53 min

COMPLETE

Option 2-2-1-1 is current optimal solution. Two other branches can be discarded because their accumulated delays are already greater than 45 min. Two branches have delays less than 45 min. and still need exploration.

Select first decision as Dir #1

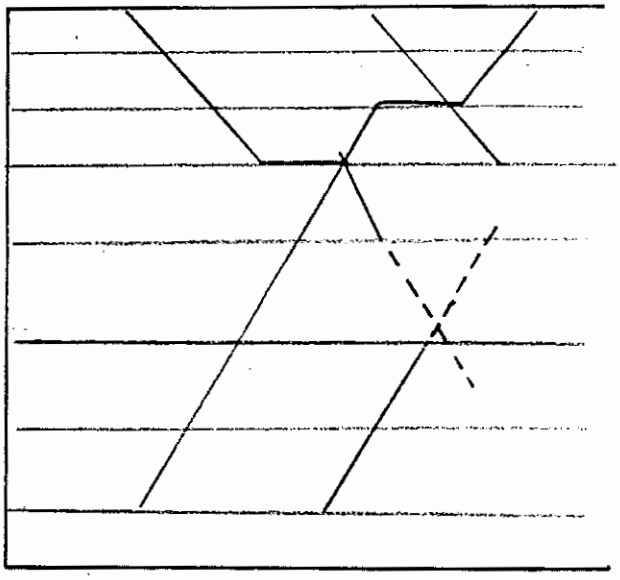




Resolve in favor of dir #1

Delay = 38 min + 8 min

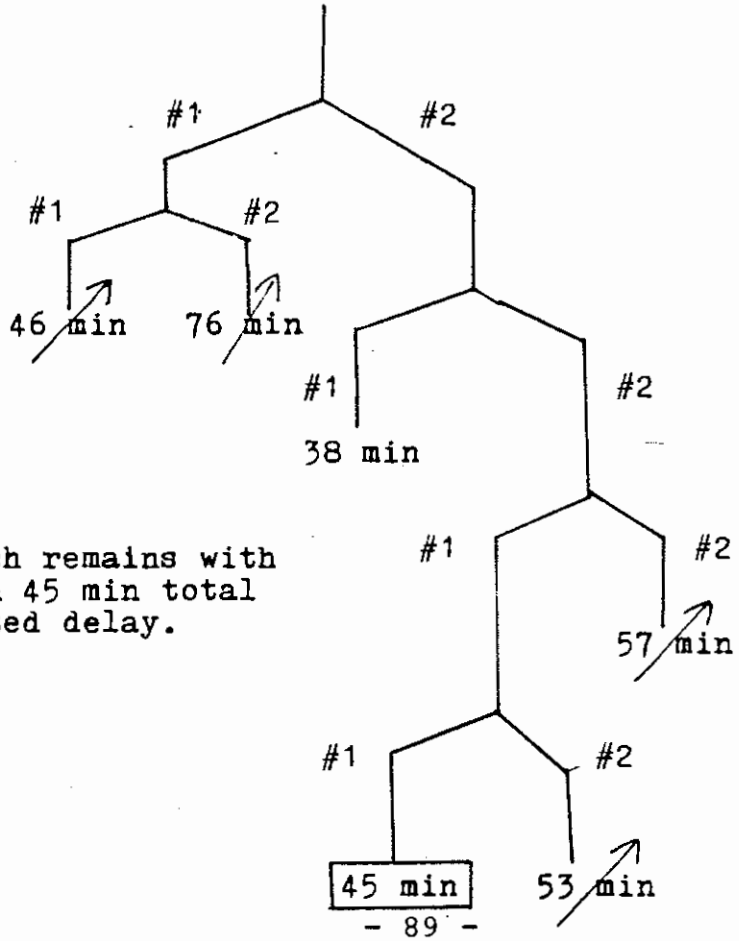
= 46 min



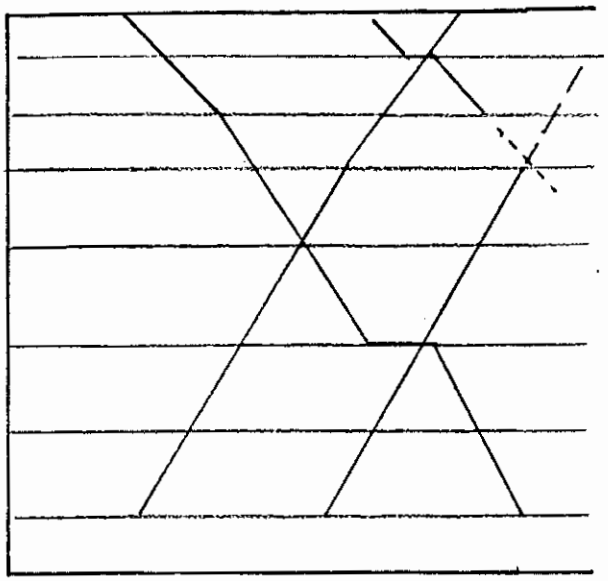
Resolve in favor of dir #2

Delay = 38 min + 38 min

= 76 min



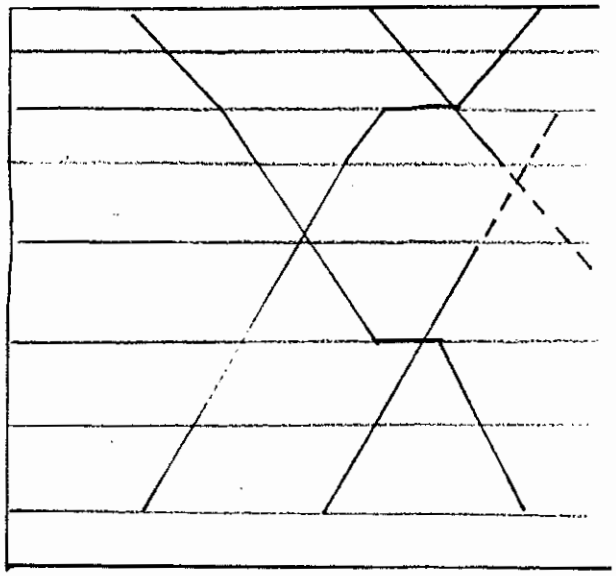
One branch remains with less than 45 min total accumulated delay.



Resolve in favor of dir #1

Delay = 38 min + 11 min

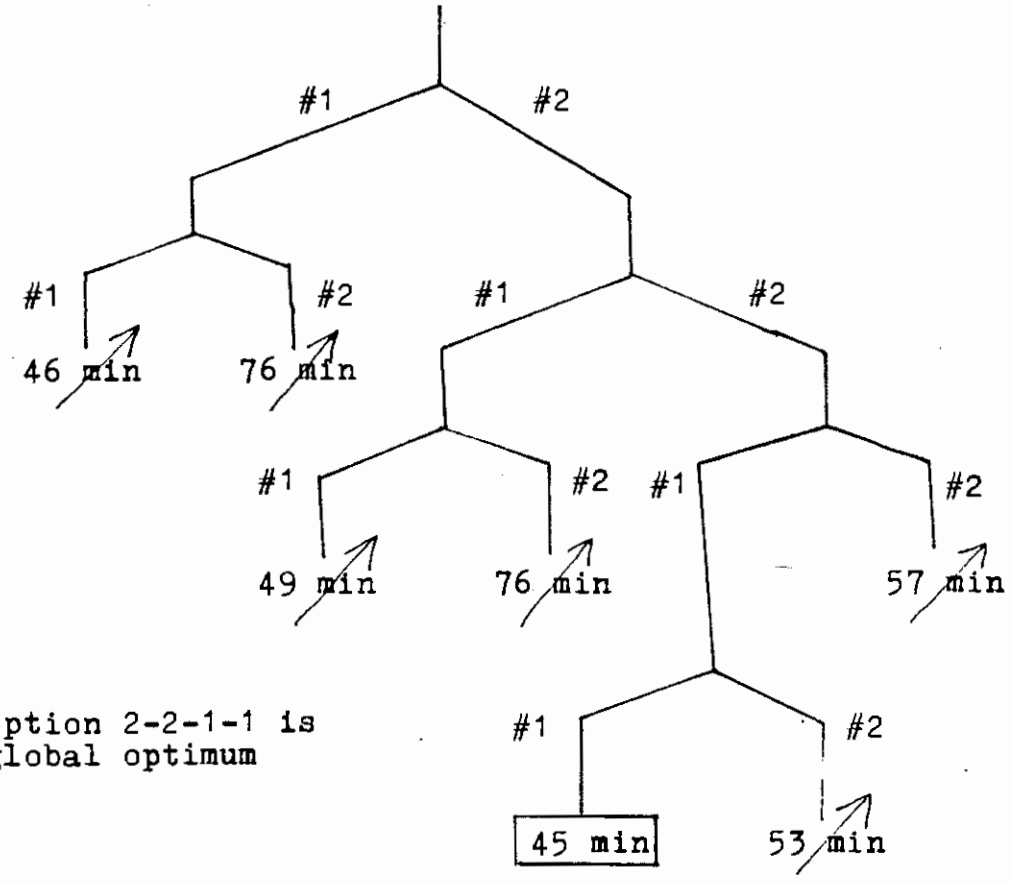
= 49 min



Resolve in favor of dir #2

Delay = 38 min + 38 min

= 76 min



Option 2-2-1-1 is global optimum

5.3. Branching with Uncertain Run Times

The branching procedure trades current delay for future delay. It is willing to accept a higher delay now if a greater amount of delay can be saved in the future. However, it is important to make sure that the future delay savings are real. If they may vanish due to the effect of uncertain train running times, the tradeoff should not be made.

By considering the impact of each meet decision on all subsequent meets, the branching procedure improves on local optimization. However, if the branching procedure accepts higher delay now, and then the anticipated savings are not realized, total delays will be higher than local optimization. Therefore the branching procedure should not be implemented unless uncertain train running times are explicitly considered.

To discount future delay savings to a realistic level, the expected value of delay must be substituted for the amount estimated as the difference between average arrival times. Otherwise the branching procedure will make poor dispatching decisions.

To consider uncertainty, the branching procedure requires three estimates: expected delay, standard deviation, and meet completion time. These can be obtained using the matrix approach described in the

previous section.

Only minor modifications to the branching procedure are required to consider variability in train run times. The following additional steps 2a, 2b and 2c must be added:

2a. When a train meet occurs, rather than using the estimated delay, the expected delay is computed. This expected delay is then used in place of the estimated delay. Substitution of this estimate changes the objective function.

2b. To compute the expected delay and meet completion time, an estimate of the standard deviation is required. Therefore, the standard deviation estimate must be maintained and updated as trains move and meet one another. The standard deviation estimate is increased for each train as its movement is projected down the rail line. When a meet occurs, the standard deviation is corrected.

2c. When a meet occurs, the meet completion time is corrected to the expected completion time rather than the arrival time of the second train.

Computer programs implementing the branching procedure are presented in the Appendix. Sample program outputs serve as numerical examples. Both versions, both not considering and considering uncertainty in running times, are implemented and

presented.

The completed branching procedure minimizes the expected value of train delay, considering the uncertainty in train run times. This automatically provides an appropriate means of comparing and trading uncertain future delays for certain present delays. The output of the procedure is a meet/pass plan which the train dispatcher is free to accept, reject or modify.

6. Conclusions

The primary goal of this thesis has been to describe important issues related to computer assisted train dispatching. The importance of accurate run time prediction was emphasized. A local optimization decision rule to minimize delay was defined. A method of simulating the impact of uncertain train running times was presented. Finally, a a computer procedure to minimize train delay was developed. An attempt has been made to ground all conclusions in firm statistical and optimization theory.

In the writer's view, the most important part of this work relates to handling of uncertainty. This is important for two reasons: first, the branching procedure will lead to poor dispatching with an increase in train delay if uncertain run times are not considered. Second, since variability limits one's ability to plan, the work attempts to realistically assess what is achievable.

Using the procedures outlined here, it is possible to create a train dispatching system which will overcome most of the difficulties of previous installations. Due to the design of the simulation model, the computer generated meet plan will not lock up the railroad. The program recommends a minimum delay course of action. It simulates far enough in

advance to help the dispatcher anticipate congestion prior to its occurrance. The combinatorial problem of multiple train meet interaction is too complex for a human dispatcher to solve, but the computer can provide this information to the dispatcher which he cannot feasibly figure for himself.

Further work remains to be done in order to implement these procedures. First, a more complete and accurate version of the simulation model should be used to simulate rail line operation. Second, an appropriate front end module to enable the model to communicate with the outside world must be developed. All the theoretical problems in these areas have already been solved: it is just a matter of completing the appropriate programming. Indeed, a more complete version has already been written. Several manufacturers have developed their own front end modules; it is only necessary to attach the programming presented here as a subroutine in order to implement the procedures.

Improved train dispatching capabilities become more vital as traffic is concentrated onto fewer routes. Some features of computer assist systems improve dispatching indirectly by reducing the dispatcher's workload. Now, the dispatcher can also receive direct assistance through computer generated

meet/pass plans. Through improved dispatching, capital productivity can be improved by increasing rail line capacity and reducing train delays to a minimum.

Appendix A contains a computer implementation of the branching procedure with uncertain train running times. It also includes the computer program used to generate the simulation results of section 4.

Appendix B deals with the topic of estimation of diversion and acceleration penalties. The procedures of Appendix B can be used to compute D_{min} in Section 3 and to estimate penalties for the simulation models of Appendix A.

NOTES

1. General Railway Signal Company, Elements of Railway Signaling, (1954, 1979), p 1002

2. Gus Welty, "A Growth Market for CTC," Railway Age, February 1983

3. Sales literature of WABCO, Harmon and GRS

4. Personal visit to Corbin KY, March 15, 1983

5. Personal visit to Richmond VA, December 30, 1982

APPENDIX A

PROGRAMMING IMPLEMENTATION

This Appendix presents a FORTRAN computer implementation of the procedures outlined in this thesis. These include three levels of implementation of the simulation and branching procedures of the fifth section. The source code of the program which simulates the impacts of run time variability, from the fourth section, is included last.

An expeditious way to implement long range planning is to adopt a simulation approach. Any of a number of simulation models can be used. CModel, developed by the writer, has been used for the past three years at the Chessie System Railroads to evaluate proposed investments. It enables an analyst to simulate proposed track or operating plan changes and forecast their effects on operating costs.

CModel is a special purpose simulator designed for application to single track rail lines with passing sidings or double track sections. In contrast to a multiple-tracked railroad, single tracked lines offer only a finite number of meeting and passing locations. The primary task on a single track line is the determination of train meeting locations, while on multiple tracked lines, it is train passing locations. A computer simulation model which is appropriate for multiple tracked lines may be inefficient, make poor

decisions and "lock up" when applied to a single track route. A single track model may be incapable of simulating a multiple tracked route. The key is to select the appropriate model for each section of rail line.

Most models capable of investment analysis are capable of meet prediction, because meet prediction is less complex. Investment models must make all meet decisions, while a real time program can depend on the dispatcher to make the difficult decisions. However, appropriate front-end programming must be added to enable the model to communicate with the outside world.

The programs presented here are purposely simplified. The goal is to avoid cluttering the important parts of the program with too many details. Program #1 presents the CModel simulation algorithm. Program #2 implements the branching procedure using the CModel algorithm. Program #3 adds a subroutine to compute expected delay, meet completion time and standard deviation, so that uncertain run times are accounted for. The completed algorithm of Program #3 is built up one step at a time from programs #1 and #2.

In general, a dispatching center's programming code may include two "macro" modules: the control macro and the planning macro. The control macro obtains inputs from the computer ports updating the

current system status, train meet instructions from the planning macro, and functional commands (throw switch, clear signal, meet train) from the dispatcher. Its output is control instructions to the ports. Most signal manufacturers have already developed control macros. Hence, that work is not repeated here.

The planning macro is used to project and plan train meets and passes. It uses stored information about track layout and running times, train location information from the control macro, and train consist and itinerary information from the record-keeping section. It simulates the future operation of the rail line and displays the results to the dispatcher. The dispatcher then may modify the meet plans to his liking. The output of the planning macro is a list of meet and pass locations. The control macro executes these commands just as if they were entered by the dispatcher directly as meet/pass commands.

CModel Simulation Procedure

On a single track line, there are two potential causes of delay. First, delay might be incurred waiting for an opposing train to clear. If a train departs point A, it will arrive and clear at B at the opposite end of single track after the specified running and clearance times have elapsed. No train can depart B until the train from A clears.

Second, delay might be incurred because of minimum train spacing requirements on a following move. After two trains have been dispatched onto single track, the first one will be operating at maximum speed, but the following train will be operating at a lower speed because it will be receiving yellow signals. Consequently the spacing between the two trains will tend to increase, until both trains are receiving clear signals. At this point the trains will be operating at a spacing of two track blocks. If the single track section is long enough, the spacing, usually about 10 minutes, will govern when the trains arrive at the end of single track and clear for opposing traffic. The trains will have no tendency to further increase their spacing if they have similar performance characteristics.

The train interactions on a single track bottleneck can be described completely by focusing on the activity at the entrance and exit switches to the single track. It does not matter whether a given run time results from enroute work or from slow running speed. What is important is the clearance time of trains at each end of the single track segment, since this is the time when opposing trains can proceed.

The simulation procedure, implemented in Program #1, is as follows:

Step 1

Input running + clearance times, and the current train positions.

Siding occupancy information and train arrival times are stored in a two-dimensional array called MODEL. The identity of the train is stored in parallel fashion in TRNUM. LSIDG records the departure time of the previous train in order to enforce the 10 minute minimum train spacing.

For example, suppose Train #1 (eastbound) has been cleared onto single track and is projected to arrive B at 1:45. (This time is converted into common time units = 1.75 hours.) Train #2 (westbound) has just arrived C at 1:30. (1:30= 1.50 hours).

A B C D

#1 1:45 #2 1:30

This is represented in CModel as follows:

0.	0.	1.50	0.	MODEL(J,1)
0	0	2	0	TRNUM(J,1)
0.	1.75	0.	0.	MODEL(J,2)
0	1	0	0	TRNUM(J,2)

Step 2

Apply the first come- first served rule by searching for the train with the earliest adjusted arrival time. Set the train's direction IDIR=1 or 2 depending on which train was selected. Set IOPP=2 or 1, equal to the opposite direction.

In this example, train 2 would be selected for first movement, since 1.5 is less than 1.75.

Step 3

Compute the selected train's arrival time at the far end of the next siding. This time is also used as the time the rear end of the train clears.

Given that section C-B has a 12 minute running time, train #2 would arrive at B at 1.70 = 1.50+.20. (.20 hours= 12 minutes.) By moving train #2 from C to B, B is determined to be the meet location. The model arrays would be updated as follows:

0.	1.70	0.	0.	MODEL(J,1)
0	2	0	0	TRNUM(J,1)
0	1.75	0.	0.	MODEL(J,2)
0	1	0	0	TRNUM(J,2)

Step 4

The trains's arrival time cannot be earlier than the previous arrival stored in LSIDG plus 10 minutes; if it is, the train must be delayed until then. Record the new arrival time in LSIDG for use by the next train.

In the example, compare the value 1.70 with the value stored in LSIDG. Since 10 minutes = .16 hours, if the previous departure from B were later than 1.54, train #2 would be delayed. Store the value 1.70 in LSIDG.

Step 5

If a meet has occurred, charge meet delay. Delay the earliest train until the arrival time of the latest train. Allow 30 seconds to throw the switch and clear the signal. Go to Step (2) and select the next train for movement.

In the example, Train #1 does not clear A-B until 1.75. Therefore train #2 is delayed until 1.75. Allowing .02 hours to line and clear the switch and signal, delay train 2 until 1.77.

0.	1.77	0.	0.	MODEL(J,1)
0	2	0	0	TRNUM(J,1)
0.	1.75	0.	0.	MODEL(J,2)
0	1	0	0	TRNUM(J,2)

Now returning to Step 2, train #1 would be selected for next movement since 1.75 < 1.77. After the next model cycle, the arrays would contain:

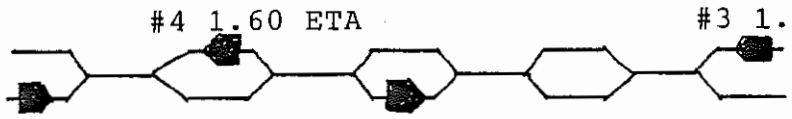
0.	1.77	0.	0.	MODEL(J,1)
0	2	0	0	TRNUM(J,1)
0.	0.	1.95	0.	MODEL(J,1)
0	0	1	0	TRNUM(J,2)

CMODEL Input Data File

Name	Dist	Min	Track Description
A	1	2	
	4	8	
B	1	2	
	3	6	
C	1	2	
	3	6	
D	1	2	
	4	8	
E	1	2	
*			
1	2	1.4	Train location & ETA
3	2	1.55	
5	1	1.45	
2	1	1.6	

Depicts the following track layout & trains:

A		B		C		D		E	
1	4	1	3	1	3	1	4	1	Miles
2	8	2	6	2	6	2	8	2	Minutes



#1 1.40 Actual #2 1.55 ETA

OUTPUT OF PROGRAM 1

File = PROG61/OUT LRL = 128 REC = ASCII

0.00 0	1.60 4	0.00 0	0.00 0	1.45 3
1.40 1	0.00 0	1.55 2	0.00 0	0.00 0

TRAIN 1 DEPART A AT 1.40
TRAIN 1 ARRIVE B AT 1.57
TRAIN 4 MEET 1 AT B DELAY .05 HOURS

0.00 0	1.60 4	0.00 0	0.00 0	1.45 3
0.00 0	1.62 1	1.55 2	0.00 0	0.00 0

TRAIN 3 DEPART E AT 1.45
TRAIN 3 ARRIVE D AT 1.62

0.00 0	1.60 4	0.00 0	1.62 3	0.00 0
0.00 0	1.62 1	1.55 2	0.00 0	0.00 0

TRAIN 2 DEPART C AT 1.55
TRAIN 2 ARRIVE D AT 1.68
TRAIN 3 MEET 2 AT D DELAY .09 HOURS

0.00 0	1.60 4	0.00 0	1.70 3	0.00 0
0.00 0	1.62 1	0.00 0	1.68 2	0.00 0

TRAIN 4 DEPART B AT 1.60
TRAIN 4 ARRIVE A AT 1.77

0.00 0	0.00 0	0.00 0	1.70 3	0.00 0
0.00 0	1.62 1	0.00 0	1.68 2	0.00 0

TRAIN 1 DEPART B AT 1.62
TRAIN 1 ARRIVE C AT 1.75

0.00 0	0.00 0	0.00 0	1.70 3	0.00 0
0.00 0	0.00 0	1.75 1	1.68 2	0.00 0

TRAIN 2 DEPART D AT 1.68
TRAIN 2 ARRIVE E AT 1.85

0.00 0	0.00 0	0.00 0	1.70 3	0.00 0
0.00 0	0.00 0	1.75 1	0.00 0	0.00 0

TRAIN 3 DEPART D AT 1.70
TRAIN 3 ARRIVE C AT 1.84
TRAIN 3 MEET 1 AT C DELAY .10 HOURS

0.00 0	0.00 0	1.84 3	0.00 0	0.00 0
0.00 0	0.00 0	1.86 1	0.00 0	0.00 0

TRAIN 3 DEPART C AT 1.84
TRAIN 3 ARRIVE B AT 1.97

0.00 0	1.97 3	0.00 0	0.00 0	0.00 0
0.00 0	0.00 0	1.86 1	0.00 0	0.00 0

TRAIN 1 DEPART C AT 1.86
TRAIN 1 ARRIVE D AT 1.99

0.00 0	1.97 3	0.00 0	0.00 0	0.00 0
0.00 0	0.00 0	0.00 0	1.99 1	0.00 0

TRAIN 3 DEPART B AT 1.97
TRAIN 3 ARRIVE A AT 2.14

0.00 0	0.00 0	0.00 0	0.00 0	0.00 0
0.00 0	0.00 0	0.00 0	1.99 1	0.00 0

TRAIN 1 DEPART D AT 1.99
TRAIN 1 ARRIVE E AT 2.16

0.00 0	0.00 0	0.00 0	0.00 0	0.00 0
0.00 0	0.00 0	0.00 0	0.00 0	0.00 0

TOTAL DELAY .24 HOURS

EOF

Branching Procedure

The branching procedure considers all possible combinations of train meeting locations. When a conflict occurs, the procedure resolves the conflict in two ways, in the sidings on both sides of the conflict area. The step by step procedure was outlined in section five.

Data describing unfinished branches must be available for further simulation. To save this data, storage space must be allocated. This space can be in core memory or on the disk drive, or both.

Program #2 stores all meet alternatives in core memory by expanding MODEL, LSIDG and TRNUM to three dimensional arrays. The additional subscript LEVEL designates which plan, #1 through #8, the model is currently simulating. Space for 8 simultaneous, independent simulations at one time has been dimensioned.

When a conflict is recognized, a new branch is created by copying the contents of the arrays from one model level to another. Once the original data has been saved, the meet is resolved in favor of one train. Then, the original data is reloaded so that the conflict can be resolved in favor of the other train.

The reader is referred to the source code of this program, located at the back of this appendix, for

additional documentation contained as comments within the code. The following are technical details of the computer implementation:

1. The one dimensional array UNRESV designates which train is to be favored in an unresolved conflict. It is set equal to direction 1 or 2 when a branch is created, and set equal to zero again when the pending train meet has been completed.

2. The array DELAY stores the total accumulated delay: it is used to select the next branch. When a meet occurs, its delay is added to the DELAY of the current LEVEL. A completed branch is labeled by changing its DELAY to a negative value. When total DELAY is printed out, the signs are all reversed again, so that negative total delay implies the branching algorithm did not need to finish simulating that particular alternative.

3. The toggle ISEARC is set to 1 whenever delay is incurred. It triggers a search of DELAY and the potential selection of another branch. Use of the toggle saves computer time by avoiding the need to search DELAY after each model iteration. DELAY is searched only when new delay is incurred.

4. MT stores the train meet plan for each strategy, and MTKT stores the total number of meets in MT. It is used to print out the summary reports at the

end of the simulation.

Program #2 is an efficient implementation of the branching algorithm. It requires less than 30K of memory to execute on a TRS-80 Model III. It is based on a very simple version of CModel. The version of CModel in use at Chessie has many additional features, including weight to power ratios, diversion and acceleration penalties, train length, and more detailed simulation of signal-related catch up and clearance time delays.

Programs #2 and #3 adopt the simplest possible scheme for storage of unfinished branches: core memory storage. Since the number of alternatives increases exponentially with the number of meets, this may lead to the necessity of using disk storage. It may be desirable to store a certain number (perhaps 10 or 20) of the best current plans in memory, and excess, poorer branches in disk storage. Additional work will be necessary to learn how to manage the swapping of various alternatives between disk and core storage. Clearly the optimum tradeoff depends on the particular computer used; the speed and cost of accessing the disk drives, the maximum amount of core memory, and the workload on the machine from other programs.

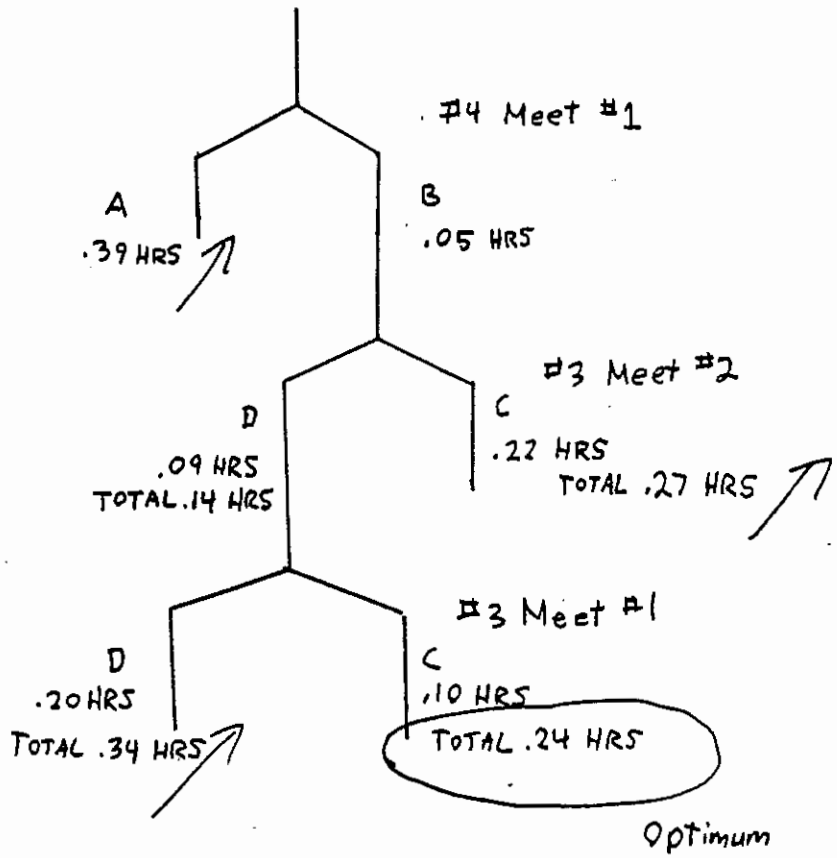
The following page contains output of program #2. The first section shows the messages generated as the

program runs. When the optimum solution has been found, all possible solutions are printed. The negative total delays indicate that the simulation on those alternatives was not finished and even more delay may be incurred. The minimum delay branch of .24 hours has less delay than the unfinished branches having delays of .39, .27 or .34 hours. These final results are summarized in a tree graph.

PROGRAM 2 OUTPUT

```
CONFLICT BETWEEN  1 AND  4
POSSIBLE SIDINGS A    AND B
LEVELS  1 AND  2
TRAIN  4 MEET  1 AT B    DELAY    .05 HOURS
TRAIN  4 MEET  1 AT A    DELAY    .39 HOURS
CONFLICT BETWEEN  2 AND  3
POSSIBLE SIDINGS C    AND D
LEVELS  1 AND  3
TRAIN  3 MEET  2 AT D    DELAY    .09 HOURS
TRAIN  3 MEET  2 AT C    DELAY    .22 HOURS
CONFLICT BETWEEN  3 AND  1
POSSIBLE SIDINGS D    AND C
LEVELS  1 AND  4
TRAIN  3 MEET  1 AT C    DELAY    .10 HOURS
TRAIN  3 MEET  1 AT D    DELAY    .20 HOURS
```

```
TRAIN  4 MEET  1 AT B
TRAIN  3 MEET  2 AT D
TRAIN  3 MEET  1 AT C
LEVEL  1 TOTAL DELAY     .24 HOURS


TRAIN  4 MEET  1 AT A
LEVEL  2 TOTAL DELAY    -.39 HOURS


TRAIN  4 MEET  1 AT B
TRAIN  3 MEET  2 AT C
LEVEL  3 TOTAL DELAY    -.27 HOURS


TRAIN  4 MEET  1 AT B
TRAIN  3 MEET  2 AT D
TRAIN  3 MEET  1 AT D
LEVEL  4 TOTAL DELAY    -.34 HOURS
```



Program #2 Output

Branching with Uncertain Run Times

Replacement of the expected value for the estimated value of delay, and utilizing the expected meet completion time are required in order to consider uncertainty in train running times. Otherwise the procedure of Program #2 is unmodified. The addition of subroutine PRPVAR allows the consideration of uncertain run times in the branching procedure. This subroutine has arguments OFFSET, STDA, STDB, AVG and DELAY.

OFFSET is the difference between the simulated arrival times of the first train (A), and the second (B).

As an input parameter, STDA is the standard deviation in arrival times of the first simulated train. STDB is the standard deviation of the second train. The same variables are used as output parameters to update the standard deviation after the meet.

Program #3 maintains an array TRVAR which contains the standard deviation in train running times. In the example, all trains initially are given a five minute standard deviation. As the trains move down the line, a percentage of the running time is added to the standard deviation estimate. PCTVAR, the assumed percentage variability, is equal to 27% from

Section 2.

The meet completion time is obtained by adding AVG to the arrival time of the earliest simulated train. The model delays both trains until the expected meet completion time rather than the simulated second train's arrival time.

Finally, DELAY returns the expected value of train delay for the meet.

The output of the procedure is the optimal meet plan and the total expected delay. While the optimal meet plan was the same, the deterministic branching procedure's delay estimate was overly optimistic. In fact, the expected delay is .43 hours instead of .24 hours. For this reason, program #3 considered more train meeting strategies.

The procedure of #3 discounts future delay savings. To show this, compare the plans of Levels 1 and 4. The only difference between the plans is that the final meet has been shifted from its optimal location at C, to a suboptimal location at D.

The cost of this shift is the difference in total delay. From Program #2, the cost is .10 hours, equal to .34 hours minus .24 hours. However, based on expected values, the cost is only .06 hours, equal to .49 hours minus .43 hours. The original savings of .10 hours has been discounted to .06 hours, a reduction of

40%.

Once again, the program output is printed. This output follows the same format as Program #2, except additional items are printed. These are the values of the estimated and expected delays, train arrival time standard deviations before and after the meet, second train arrival time and the expected meet completion time. The simulation results are displayed in tree form.

Following these results are the computer source codes of programs #1, 2 and 3, along with the the source code of the program used in section 4.

PROGRAM 3 OUTPUT

```
CONFLICT BETWEEN  1 AND  4
POSSIBLE SIDINGS A    AND B
LEVELS  1 AND  2

ESTIMATED DELAY       .05
STANDARD DEV TRN 1     .13
STANDARD DEV TRN 4     .08
2ND TRAIN ARRIVAL     1.60
ST DEV AFTER MEET      .08
COMPLETION TIME       1.65
EXPECTED DELAY         .12
TRAIN  4 MEET  1 AT B    DELAY    .12 HOURS

ESTIMATED DELAY       .39
STANDARD DEV TRN 1     .08
STANDARD DEV TRN 4     .13
2ND TRAIN ARRIVAL     1.77
ST DEV AFTER MEET      .13
COMPLETION TIME       1.79
EXPECTED DELAY         .39
TRAIN  4 MEET  1 AT A    DELAY    .39 HOURS
```

```
CONFLICT BETWEEN  2 AND  3
POSSIBLE SIDINGS C    AND D
LEVELS  1 AND  3

ESTIMATED DELAY       .09
STANDARD DEV TRN 3     .13
STANDARD DEV TRN 2     .12
2ND TRAIN ARRIVAL     1.68
ST DEV AFTER MEET      .10
COMPLETION TIME       1.73
EXPECTED DELAY         .15
TRAIN  3 MEET  2 AT D    DELAY    .15 HOURS

ESTIMATED DELAY       .22
STANDARD DEV TRN 2     .08
STANDARD DEV TRN 3     .16
2ND TRAIN ARRIVAL     1.75
ST DEV AFTER MEET      .15
COMPLETION TIME       1.78
EXPECTED DELAY        .23
TRAIN  3 MEET  2 AT C    DELAY    .23 HOURS
```

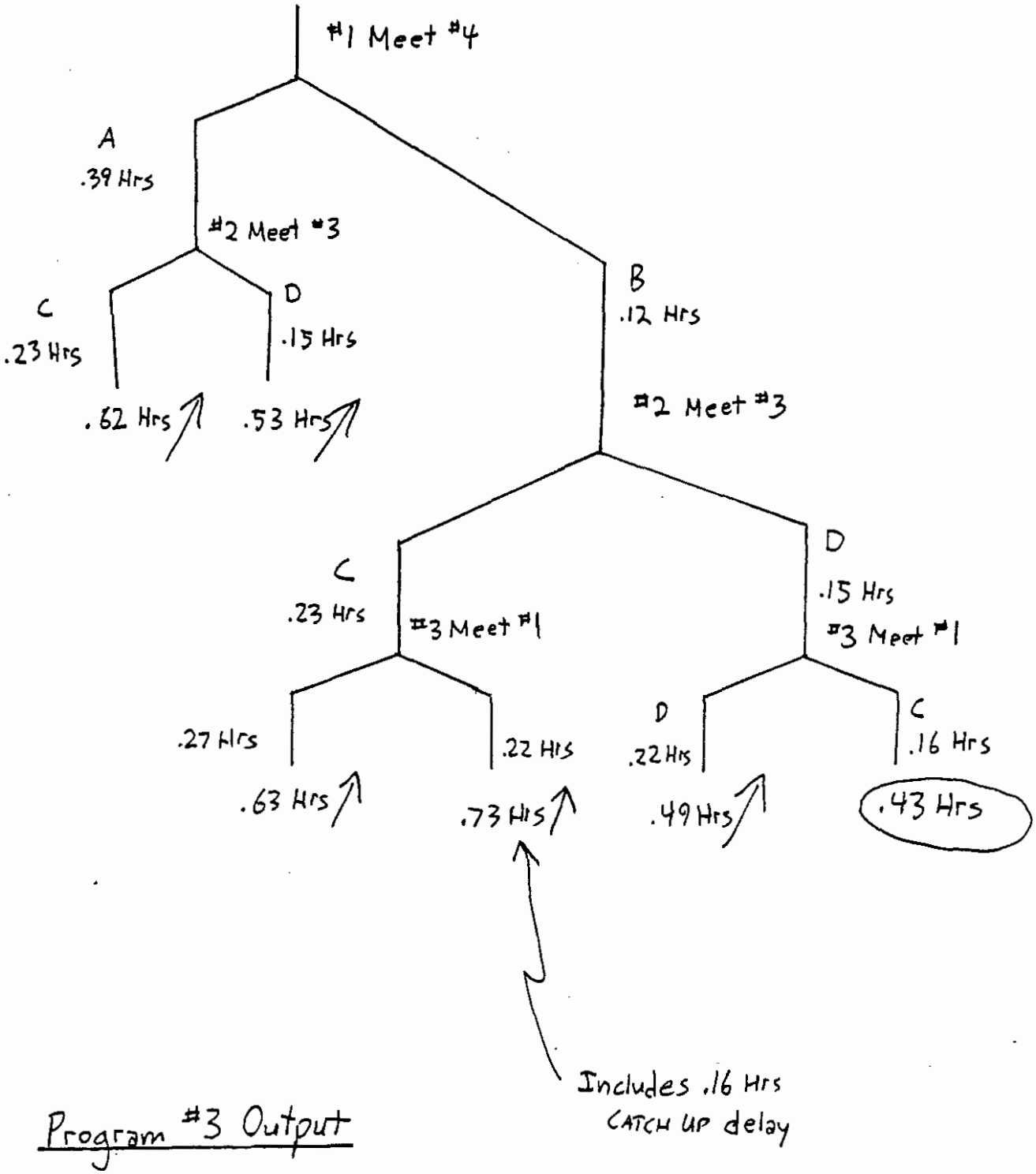
```
CONFLICT BETWEEN  3 AND  1
POSSIBLE SIDINGS D    AND C
LEVELS  1 AND  4

ESTIMATED DELAY       .10
STANDARD DEV TRN 1     .12
STANDARD DEV TRN 3     .14
2ND TRAIN ARRIVAL     1.89
ST DEV AFTER MEET      .11
COMPLETION TIME       1.94
EXPECTED DELAY         .16
TRAIN  3 MEET  1 AT C    DELAY     .16 HOURS

ESTIMATED DELAY       .21
STANDARD DEV TRN 3     .10
STANDARD DEV TRN 1     .15
2ND TRAIN ARRIVAL     1.94
ST DEV AFTER MEET      .14
COMPLETION TIME       1.97
EXPECTED DELAY         .22
TRAIN  3 MEET  1 AT D    DELAY     .22 HOURS
```

```
CONFLICT BETWEEN  1 AND  3
POSSIBLE SIDINGS B    AND C
LEVELS  3 AND  5

ESTIMATED DELAY       .20
STANDARD DEV TRN 3     .15
STANDARD DEV TRN 1     .12
2ND TRAIN ARRIVAL     1.96
ST DEV AFTER MEET      .11
COMPLETION TIME       1.99
EXPECTED DELAY         .22
TRAIN  3 MEET  1 AT C    DELAY     .22 HOURS

ESTIMATED DELAY       .26
STANDARD DEV TRN 1     .08
STANDARD DEV TRN 3     .19
2ND TRAIN ARRIVAL     1.91
ST DEV AFTER MEET      .18
COMPLETION TIME       1.94
EXPECTED DELAY         .27
TRAIN  3 MEET  1 AT B    DELAY     .27 HOURS
```

```
CONFLICT BETWEEN  2 AND  3
POSSIBLE SIDINGS C    AND D
LEVELS  2 AND  6

ESTIMATED DELAY       .09
STANDARD DEV TRN 3     .13
STANDARD DEV TRN 2     .12
2ND TRAIN ARRIVAL     1.68
ST DEV AFTER MEET      .10
COMPLETION TIME       1.73
EXPECTED DELAY         .15
TRAIN  3 MEET  2 AT D    DELAY     .15 HOURS

ESTIMATED DELAY       .22
STANDARD DEV TRN 2     .08
STANDARD DEV TRN 3     .16
2ND TRAIN ARRIVAL     1.75
ST DEV AFTER MEET      .15
COMPLETION TIME       1.78
EXPECTED DELAY         .23
TRAIN  3 MEET  2 AT C    DELAY     .23 HOURS
```

```
TRAIN  4 MEET  1 AT B
TRAIN  3 MEET  2 AT D
TRAIN  3 MEET  1 AT C
LEVEL  1 TOTAL DELAY      .43 HOURS


TRAIN  4 MEET  1 AT A
TRAIN  3 MEET  2 AT D
LEVEL  2 TOTAL DELAY     -.53 HOURS


TRAIN  4 MEET  1 AT B
TRAIN  3 MEET  2 AT C
TRAIN  3 MEET  1 AT C
LEVEL  3 TOTAL DELAY     -.73 HOURS
```

```
TRAIN   4 MEET   1 AT B
TRAIN   3 MEET   2 AT D
TRAIN   3 MEET   1 AT D
LEVEL   4 TOTAL DELAY     -.49 HOURS


TRAIN   4 MEET   1 AT B
TRAIN   3 MEET   2 AT C
TRAIN   3 MEET   1 AT B
LEVEL   5 TOTAL DELAY     -.63 HOURS


TRAIN   4 MEET   1 AT A
TRAIN   3 MEET   2 AT C
LEVEL   6 TOTAL DELAY     -.62 HOURS
```



Program #3 Output

PROGRAM #1 - CMODEL ALGORITHM

File = PROG61/FOR LRL = 256 REC = ASCII

```
00100          PROGRAM CMODEL
00200          REAL SLNGH(10),STIME(10),BLNGH(10),BTIME(10),
00300         *  MODEL(10,2), LSIDG(10,2)
00400          INTEGER TRNUM(10,2)
00500          INTEGER*4 NAME(10), ISTAR, KSTAR
00600          DATA MODEL,LSIDG,TRNUM /40*0.,20*0/
00700          DATA ISTAR /4H*   /
00800          DELAY=0.
00900    C
01000    C       PROGRAM 6.1- LOCAL OPTIMIZATION
01100    C       (FIRST COME FIRST SERVED)
01200    C       DETERMINISTIC SOLUTION
01300    C
01400    C       THE FINAL OUTPUT INCLUDES ARRIVAL AND
01500    C       DEPARTURE REPORTS IN ADDITION TO THE
01600    C       MEETING LOCATIONS. AT THE END, TOTAL
01700    C       DELAY IS REPORTED
01800    C
01900    C
02000    C       INPUT TRACK DATA
02100    C
02200          CALL OPEN(5,'TRACK/DAT',256)
02300          DO 600 J=1,10
02400    C
02500    C       INPUT SIDING NAME, LENGTH & RUN TIME
02600    C
02700          READ(5,200)NAME(J),SLNGH(J),STIME(J)
02800          STIME(J)=STIME(J)/60.
02900    200   FORMAT(A4,2F5.0)
03000    C
03100    C       INPUT SINGLE TRACK LENGTH & RUN TIME
03200    C
03300          READ(5,200) KSTAR,BLNGH(J),BTIME(J)
03400          BTIME(J)=BTIME(J)/60.
03500          IF( KSTAR .EQ. ISTAR) GO TO 20
03600    600   CONTINUE
03700    20    NSDGS=J
03800    C
03900    C       INPUT TRAIN POSITIONS AND E.T.A.
04000    C
```

```
04100           DO 601 J=1,5
04200           READ(5,202,END=10) ICORD, IDIR, TIME
04300    202    FORMAT(2I5,F8.2)
04400           MODEL(ICORD,IDIR)= TIME
04500           TRNUM(ICORD,IDIR)= J
04600    601    LSIDG(ICORD,IDIR)= TIME
04700    C
04800    C       SEARCH FOR EARLIEST TIME
04900    C        SET ICORD= SIDING LOCATION
05000    C            IDIR = DIRECTION
05100    C            IOPP = OPPOSITE DIRECTION
05200    C            OF TRAIN SELECTED FOR MOVEMENT
05300    C
05400    10     ALATE=-999999.
05500    13     EARLY= 999999.
05600           DO 700 J=1,NSDGS
05700           IF( MODEL(J,2) .EQ. 0.) GO TO 700
05800           ADJUST= MODEL(J,2)
05900           IF( ADJUST .GE. EARLY .OR. ADJUST .LE. ALATE)
06000          * GO TO 700
06100           IDIR= 2
06200           IOPP= 1
06300           ICORD= J
06400           EARLY= ADJUST
06500     700   CONTINUE
06600    C
06700    C       OTHER DIRECTION
06800    C
06900           DO 701 J=1,NSDGS
07000           IF( MODEL(J,1) .EQ. 0.) GO TO 701
07100           ADJUST= MODEL(J,1)
07200           IF( ADJUST .GE. EARLY .OR. ADJUST .LE. ALATE)
07300          *  GO TO 701
07400           IDIR= 1
07500           IOPP= 2
07600           ICORD= J
07700           EARLY= ADJUST
07800     701   CONTINUE
07900    C
08000    C        REPORT DELAY AND STOP IF ALL TRAINS
08100    C        HAVE TERMINATED
08200    C
08300           IF( EARLY .EQ. 999999.) GO TO 30
08400           ALATE=EARLY
08500    C
08600    C        SET LOCATION OF DESTINATION SIDING
08700    C          NEXT SIDING COORDINATE +1 OR -1
08800    C          DEPENDING ON TRAIN DIRECTION
08900    C
```

```
09000           IF(IDIR .EQ. 1) NEXT= ICORD-1
09100           IF(IDIR .EQ. 2) NEXT= ICORD+1
09200     C
09300     C       IF NEXT SIDING IS OCCUPIED, SEARCH FOR
09400     C       SECOND BEST TRAIN
09500     C
09600           IF( MODEL(NEXT,IDIR) .NE. 0.) GO TO 13
09700     C
09800     C     RUN TRAIN(ICORD,IDIR)
09900     C     COMPUTE ARRIVAL TIME AT NEXT LOCATION
10000     C
10100           IF(IDIR .EQ. 2) MODEL(NEXT,2)= MODEL(ICORD,2) +
10200          *  (BTIME(ICORD)+ STIME(NEXT))
10300           IF(IDIR .EQ. 1) MODEL(NEXT,1)= MODEL(ICORD,1) +
10400          *  (BTIME(NEXT)+ STIME(NEXT))
10500           WRITE(1,101)TRNUM(ICORD,IDIR),NAME(ICORD),
10600          * MODEL(ICORD,IDIR)
10700     101   FORMAT(' TRAIN ',I2,' DEPART ',A4,' AT ',F6.2)
10800     C
10900     C    CATCH UP DELAY    MINIMUM 10 MIN. SPACING
11000     C
11100           IF(MODEL(NEXT,IDIR) .GE. LSIDG(NEXT,IDIR) + .16)
11200          *  GO TO 15
11300           OFFSET= LSIDG(NEXT,IDIR)-MODEL(NEXT,IDIR)+.16
11400           DELAY= DELAY+ OFFSET
11500           WRITE(1,103) TRNUM(ICORD,IDIR), NAME(NEXT),
11600          * OFFSET
11700     103   FORMAT(' TRAIN ',I2,' CATCH UP ',A4,' DELAY ',
11800          *  F6.2,' HOURS ')
11900           MODEL(NEXT,IDIR)= LSIDG(NEXT,IDIR) + .16
12000     15    WRITE(1,102) TRNUM(ICORD,IDIR),NAME(NEXT),
12100          * MODEL(NEXT,IDIR)
12200     102   FORMAT(' TRAIN ',I2,' ARRIVE ',A4,' AT ',F6.2)
12300           TRNUM(NEXT,IDIR)= TRNUM(ICORD,IDIR)
12400     C
12500     C       CHECK FOR OPPOSING TRAIN
12600     C       TO DETERMINE IF A MEET HAS OCCURRED
12700     C
12800           IF(MODEL(NEXT,IOPP) .EQ. 0.) GO TO 16
12900     C
13000     C       DELAY OR BE DELAYED BY OPPOSING TRAIN
13100     C
13200           OFFSET= MODEL(NEXT,1)-MODEL(NEXT,2)
13300           DELMT= ABS(OFFSET)+ .02
13400           DELAY= DELAY+ DELMT
13500           IF(MODEL(NEXT,IOPP) .LT. MODEL(NEXT,IDIR))
13600          *  GO TO 17
13700           MODEL(NEXT,IDIR)= MODEL(NEXT,IOPP)+ .02
13800           GO TO 18
```

```
13900     17   MODEL(NEXT,IOPP)= MODEL(NEXT,IDIR)+ .02
14000    C
14100    C       WRITE MEET MESSAGE
14200    C
14300    18    WRITE(1,100) TRNUM(NEXT,1),TRNUM(NEXT,2),
14400         *  NAME(NEXT), DELMT
14500    100   FORMAT(' TRAIN ',I2,' MEET ',I2,' AT ',A4,
14600         * ' DELAY ',F6.2,' HOURS')
14700    C
14800    C       RECORD ARRIVAL TIME IN LSIDG FOR
14900    C       CATCH UP DELAY OF NEXT TRAIN
15000    C
15100     16   LSIDG(ICORD,IDIR)= MODEL(ICORD,IDIR)
15200    C
15300    C       CLEAR PREVIOUS LOCATION
15400    C
15500          MODEL(ICORD,IDIR)= 0.
15600          TRNUM(ICORD,IDIR)= 0
15700    C
15800    C       TERMINATE TRAIN IF END OF LINE REACHED
15900    C
16000          IF( IDIR .EQ. 1 .AND. NEXT .EQ. 1 .OR.
16100         * IDIR .EQ. 2 .AND. NEXT .EQ. NSDGS)
16200         * MODEL(NEXT,IDIR)=0.
16300          GO TO 10
16400    30    WRITE(1,105) DELAY
16500    105   FORMAT(' TOTAL DELAY ',F8.2,' HOURS'/)
16600          STOP
16700          END
EOF
```

PROGRAM #2 - BRANCHING

File = PROG62/FOR LRL = 256 REC = ASCII

```
00100          PROGRAM CMODEL
00200          REAL SLNGH(10),STIME(10),BLNGH(10),BTIME(10),
00300         *  MODEL(10,2,8), LSIDG(10,2,8), DELAY(8)
00400          INTEGER TRNUM(10,2,8), UNRESV(8), MT(3,3,8),
00500         * MTKT(8)
00600          INTEGER*4 NAME(10), ISTAR, KSTAR
00700          DATA MODEL,LSIDG,TRNUM /320*0.,160*0/
00800          DATA DELAY, MTKT /8*0., 8*0/
00900          DATA ISTAR /4H*   /
01000    C
01100    C       PROGRAM 6.2- GLOBAL OPTIMIZATION BRANCHING
01200    C       DETERMINISTIC SOLUTION
01300    C
01400    C
01500          LEVEL=1
01600          KLEVEL=1
01700          UNRESV(1)=0
01800          FBEST=999999.
01900    C
02000    C       INPUT TRACK DATA
02100    C
02200          CALL OPEN(5,'TRACK/DAT',256)
02300          DO 600 J=1,10
02400    C
02500    C       INPUT SIDING NAME, LENGTH & RUN TIME
02600    C
02700          READ(5,200)NAME(J),SLNGH(J),STIME(J)
02800          STIME(J)=STIME(J)/60.
02900    200   FORMAT(A4,2F5.0)
03000    C
03100    C       INPUT SINGLE TRACK LENGTH & RUN TIME
03200    C
03300          READ(5,200) KSTAR,BLNGH(J),BTIME(J)
03400          BTIME(J)=BTIME(J)/60.
03500          IF( KSTAR .EQ. ISTAR) GO TO 20
03600    600   CONTINUE
03700    20    NSDGS=J
03800    C
03900    C       INPUT TRAIN POSITIONS AND E.T.A.
04000    C
04100          DO 601 J=1,5
04200          READ(5,202,END=10) ICORD, IDIR, TIME
04300    202   FORMAT(2I5,F8.2)
04400          MODEL(ICORD,IDIR,1)= TIME
```

```
04500           TRNUM(ICORD,IDIR,1)= J
04600     601   LSIDG(ICORD,IDIR,1)= TIME
04700     C
04800     C      SEARCH FOR EARLIEST TIME
04900     C       SET ICORD= SIDING LOCATION
05000     C           IDIR = DIRECTION
05100     C           IOPP = OPPOSITE DIRECTION
05200     C           OF TRAIN SELECTED FOR MOVEMENT
05300     C
05400     10    ALATE=-999999.
05500     13    EARLY= 999999.
05600           DO 700 J=1,NSDGS
05700           IF( MODEL(J,2,KLEVEL) .EQ. 0.) GO TO 700
05800     C
05900     C       SELECT TRAIN BASED ON ETA NEXT LOCATION
06000     C       SO CONFLICT RECOGNITION WORKS
06100     C
06200           NEXT=J+1
06300           RTIME= BTIME(J)+ STIME(NEXT)
06400           ADJUST= MODEL(J,2,KLEVEL)+ RTIME
06500           IF( ADJUST .GE. EARLY .OR. ADJUST .LE. ALATE)
06600          * GO TO 700
06700           IDIR= 2
06800           IOPP= 1
06900           ICORD= J
07000           EARLY= ADJUST
07100      700  CONTINUE
07200     C
07300     C      OTHER DIRECTION
07400     C
07500           DO 701 J=1,NSDGS
07600           IF( MODEL(J,1,KLEVEL) .EQ. 0.) GO TO 701
07700           NEXT=J-1
07800           RTIME= BTIME(NEXT)+ STIME(NEXT)
07900           ADJUST= MODEL(J,1,KLEVEL)+ RTIME
08000           IF( ADJUST .GE. EARLY .OR. ADJUST .LE. ALATE)
08100          *  GO TO 701
08200           IDIR= 1
08300           IOPP= 2
08400           ICORD= J
08500           EARLY= ADJUST
08600      701  CONTINUE
08700     C
08800     C       REPORT DELAY AND STOP IF ALL TRAINS
08900     C       HAVE TERMINATED
09000     C
```

```
09100           IF( EARLY .NE. 999999.) GO TO 30
09200    C
09300    C        MARK CURRENT LEVEL AS FINISHED BY
09400    C        NEGATIVE SIGN
09500    C
09600           IF( DELAY(KLEVEL) .LT. FBEST)
09700          *  FBEST= DELAY(KLEVEL)
09800           DELAY(KLEVEL)= -DELAY(KLEVEL)
09900    C
10000    C        GO TO 32 AND SEARCH FOR BEST UNFINISHED
10100    C        LEVEL
10200    C
10300           GO TO 32
10400    30     ALATE=EARLY
10500    C
10600    C        SET LOCATION OF DESTINATION SIDING
10700    C         NEXT SIDING COORDINATE +1 OR -1
10800    C         DEPENDING ON TRAIN DIRECTION
10900    C
11000           IF(IDIR .EQ. 1) NEXT=ICORD- 1
11100           IF(IDIR .EQ. 2) NEXT= ICORD+1
11200    C
11300    C        CONFLICT RECOGNITION IF NEXT SIDING
11400    C        OPPOSITE LOCATION IS OCCUPIED, THEN
11500    C        BRANCH
11600    C
11700           ISEARC= 0
11800           IF( MODEL(NEXT,IOPP,KLEVEL) .EQ. 0.) GO TO 14
11900           IF( UNRESV(KLEVEL) .EQ. 0) GO TO 50
12000           IF( UNRESV(KLEVEL) .EQ. IDIR) GO TO 14
12100           N=ICORD
12200           ICORD= NEXT
12300           NEXT= N
12400           N= IDIR
12500           IDIR= IOPP
12600           IOPP= N
12700           GO TO 14
12800    C
12900    C        COPY MODEL STATUS TO ANOTHER ARRAY LEVEL
13000    C
13100    50     LEVEL=LEVEL+1
13200           ITRNUM= TRNUM(ICORD,IDIR,KLEVEL)
13300           NUMOPP= TRNUM(NEXT,IOPP,KLEVEL)
13400           WRITE(2,402) ITRNUM,NUMOPP,NAME(ICORD),NAME(NEXT)
13500    402    FORMAT(' CONFLICT BETWEEN ',I2,' AND ',I2,/,
13600          * ' POSSIBLE SIDINGS ',A4,' AND ',A4)
```

```
13700           WRITE(2,400) KLEVEL, LEVEL
13800    400    FORMAT(' LEVELS ',I2,' AND ',I2)
13900           UNRESV(KLEVEL)=IDIR
14000           UNRESV(LEVEL)= IOPP
14100           DO 800 J=1,NSDGS
14200           DO 800 K=1,2
14300           MODEL(J,K,LEVEL)=MODEL(J,K,KLEVEL)
14400           LSIDG(J,K,LEVEL)=LSIDG(J,K,KLEVEL)
14500    800    TRNUM(J,K,LEVEL)=TRNUM(J,K,KLEVEL)
14600           DELAY(LEVEL)= DELAY(KLEVEL)
14700           KTMT=MTKT(KLEVEL)
14800           MTKT(LEVEL)= KTMT
14900           DO 801 J=1,KTMT
15000           DO 801 K=1,3
15100    801    MT(K,J,LEVEL)= MT(K,J,KLEVEL)
15200    C
15300    C        IF NEXT SIDING IS OCCUPIED SEARCH FOR NEXT
15400    C        BEST TRAIN
15500    C
15600    14     IF(MODEL(NEXT,IDIR,KLEVEL) .NE. 0.) GO TO 13
15700    C
15800    C      RUN TRAIN(ICORD,IDIR)
15900    C      COMPUTE ARRIVAL TIME AT NEXT LOCATION
16000    C
16100           IF(IDIR .EQ. 2) MODEL(NEXT,2,KLEVEL)=
16200          * MODEL(ICORD,2,KLEVEL)+BTIME(ICORD)+ STIME(NEXT)
16300           IF(IDIR .EQ. 1) MODEL(NEXT,1,KLEVEL)=
16400          * MODEL(ICORD,1,KLEVEL)+BTIME(NEXT)+ STIME(NEXT)
16500    C
16600    C     CATCH UP DELAY    MINIMUM 10 MIN. SPACING
16700    C
16800           IF(MODEL(NEXT,IDIR,KLEVEL) .GE.
16900          * LSIDG(NEXT,IDIR,KLEVEL)+.16)  GO TO 15
17000           OFFSET= LSIDG(NEXT,IDIR,KLEVEL)-
17100          * MODEL(NEXT,IDIR,KLEVEL)+ .16
17200           DELAY(KLEVEL)=  DELAY(KLEVEL)+ OFFSET
17300           ISEARC= 1
17400           MODEL(NEXT,IDIR,KLEVEL)= LSIDG(NEXT,IDIR,KLEVEL)
17500          *  + .16
17600    15     TRNUM(NEXT,IDIR,KLEVEL)= TRNUM(ICORD,IDIR,KLEVEL)
17700    C
17800    C        CHECK FOR OPPOSING TRAIN
17900    C        TO DETERMINE IF A MEET HAS OCCURRED
18000    C
18100           IF(MODEL(NEXT,IOPP,KLEVEL) .EQ. 0.) GO TO 16
18200    C
18300    C        DELAY OR BE DELAYED BY OPPOSING TRAIN
18400    C
```

```
18500          OFFSET= MODEL(NEXT,1,KLEVEL)-MODEL(NEXT,2,KLEVEL)
18600          DELMT= ABS(OFFSET)+ .02
18700          DELAY(KLEVEL)= DELAY(KLEVEL)+ DELMT
18800          ISEARC= 1
18900          IF(MODEL(NEXT,IOPP,KLEVEL) .LT.
19000         *  MODEL(NEXT,IDIR,KLEVEL)) GO TO 17
19100          MODEL(NEXT,IDIR,KLEVEL)= MODEL(NEXT,IOPP,KLEVEL)
19200         * + .02
19300          GO TO 18
19400     17   MODEL(NEXT,IOPP,KLEVEL)= MODEL(NEXT,IDIR,KLEVEL)
19500         *  + .02
19600    C
19700    C       STORE MEET MESSAGE IN MT, MTKT ARRAYS
19800    C
19900    18    MTKT(KLEVEL)=MTKT(KLEVEL)+1
20000          KTMT= MTKT(KLEVEL)
20100          MT(1,KTMT,KLEVEL)=TRNUM(NEXT,1,KLEVEL)
20200          MT(2,KTMT,KLEVEL)=TRNUM(NEXT,2,KLEVEL)
20300          MT(3,KTMT,KLEVEL)=NEXT
20400          UNRESV(KLEVEL)= 0
20500          WRITE(2,100)TRNUM(NEXT,1,KLEVEL),
20600         * TRNUM(NEXT,2,KLEVEL),NAME(NEXT),DELMT
20700    100   FORMAT(' TRAIN ',I2,' MEET ',I2,' AT ',A4,
20800         * ' DELAY ',F6.2,' HOURS')
20900    C
21000    C       RECORD ARRIVAL TIME IN LSIDG FOR
21100    C       CATCH UP DELAY OF NEXT TRAIN
21200    C
21300     16   LSIDG(ICORD,IDIR,KLEVEL)=MODEL(ICORD,IDIR,KLEVEL)
21400    C
21500    C       CLEAR PREVIOUS LOCATION
21600    C
21700          MODEL(ICORD,IDIR,KLEVEL)= 0.
21800          TRNUM(ICORD,IDIR,KLEVEL)= 0
21900    C
22000    C       TERMINATE TRAIN IF END OF LINE REACHED
22100    C
22200          IF( IDIR .EQ. 1 .AND. NEXT .EQ. 1 .OR.
22300         * IDIR .EQ. 2 .AND. NEXT .EQ. NSDGS)
22400         * MODEL(NEXT,IDIR,KLEVEL)=0.
22500          IF( ISEARC .EQ. 0) GO TO 10
22600    C
22700    C       SEARCH FOR BEST UNFINISHED LEVEL
22800    C
22900    32    BEST= 999999.
23000          DO 806 J=1,LEVEL
23100          IF( DELAY(J) .LT. 0.) GO TO 806
```

```
23200           IF( DELAY(J) .GE. BEST) GO TO 806
23300           IF( DELAY(J) .GE. FBEST) GO TO 806
23400           KLEVEL=J
23500           BEST= DELAY(J)
23600     806   CONTINUE
23700           IF( BEST .NE. 999999.) GO TO 10
23800           DO 805 KLEVEL=1,LEVEL
23900           WRITE(2,107)
24000     107   FORMAT(/)
24100           KTMT=MTKT(KLEVEL)
24200           DO 807 J=1,KTMT
24300           NEXT= MT(3,J,KLEVEL)
24400     807   WRITE(2,105)MT(1,J,KLEVEL), MT(2,J,KLEVEL),
24500          * NAME(NEXT)
24600     105   FORMAT(' TRAIN ',I2,' MEET ',I2,' AT ',A4)
24700           DELAY(KLEVEL)= -DELAY(KLEVEL)
24800     805   WRITE(2,106) KLEVEL, DELAY(KLEVEL)
24900     106   FORMAT(' LEVEL',I3,' TOTAL DELAY',F8.2,' HOURS'/)
25000           STOP
25100           END
EOF
```

PROGRAM #3- BRANCHING - UNCERTAIN TIMES

File = PROG63/FOR LRL = 256 REC = ASCII

```
00100          PROGRAM CMODEL
00200          REAL SLNGH(10),STIME(10),BLNGH(10),BTIME(10),
00300         *  MODEL(10,2,8), LSIDG(10,2,8), DELAY(8),
00400         *  TRVAR(5,8)
00500          INTEGER TRNUM(10,2,8), UNRESV(8), MT(3,3,8),
00600         * MTKT(8)
00700          INTEGER*4 NAME(10), ISTAR, KSTAR
00800          DATA MODEL,LSIDG,TRNUM /320*0.,160*0/
00900          DATA DELAY, MTKT /8*0., 8*0/
01000          DATA ISTAR /4H*   /
01100    C
01200    C       PROGRAM 6.3- GLOBAL OPTIMIZATION BRANCHING
01300    C       INCLUDING UNCERTAIN RUN TIMES
01400    C
01500    C       THE FINAL OUTPUT INCLUDES ARRIVAL AND
01600    C       DEPARTURE REPORTS IN ADDITION TO THE
01700    C       MEETING LOCATIONS. AT THE END, TOTAL
01800    C       DELAY IS REPORTED
01900    C
02000          PCTVAR= .27
02100          LEVEL=1
02200          KLEVEL=1
02300          UNRESV(1)=0
02400          FBEST=999999.
02500    C
02600    C       INPUT TRACK DATA
02700    C
02800          CALL OPEN(5,'TRACK/DAT',256)
02900          DO 600 J=1,10
03000    C
03100    C       INPUT SIDING NAME, LENGTH & RUN TIME
03200    C
03300          READ(5,200)NAME(J),SLNGH(J),STIME(J)
03400          STIME(J)=STIME(J)/60.
03500    200   FORMAT(A4,2F5.0)
03600    C
03700    C       INPUT SINGLE TRACK LENGTH & RUN TIME
03800    C
03900          READ(5,200) KSTAR,BLNGH(J),BTIME(J)
04000          BTIME(J)=BTIME(J)/60.
04100          IF( KSTAR .EQ. ISTAR) GO TO 20
04200    600   CONTINUE
04300    20    NSDGS=J
04400    C
04500    C       INPUT TRAIN POSITIONS AND E.T.A.
04600    C
```

```
04700          DO 601 J=1,5
04800          READ(5,202,END=10) ICORD, IDIR, TIME, TRVAR(J,1)
04900    202   FORMAT(2I5,2F8.2)
05000          MODEL(ICORD,IDIR,1)= TIME
05100          TRNUM(ICORD,IDIR,1)= J
05200    601   LSIDG(ICORD,IDIR,1)= TIME
05300    C
05400    C      SEARCH FOR EARLIEST TIME
05500    C
05600    10    ALATE=-999999.
05700    13    EARLY= 999999.
05800          DO 700 J=1,NSDGS
05900          IF( MODEL(J,2,KLEVEL) .EQ. 0.) GO TO 700
06000          NEXT=J+1
06100          RTIME= BTIME(J)+ STIME(NEXT)
06200          ADJUST= MODEL(J,2,KLEVEL)+ RTIME
06300          IF( ADJUST .GE. EARLY .OR. ADJUST .LE. ALATE)
06400         * GO TO 700
06500          IDIR= 2
06600          IOPP= 1
06700          ICORD= J
06800          EARLY= ADJUST
06900     700  CONTINUE
07000    C
07100    C      OTHER DIRECTION
07200    C
07300          DO 701 J=1,NSDGS
07400          IF( MODEL(J,1,KLEVEL) .EQ. 0.) GO TO 701
07500          NEXT=J-1
07600          RTIME= BTIME(NEXT)+ STIME(NEXT)
07700          ADJUST= MODEL(J,1,KLEVEL)+ RTIME
07800          IF( ADJUST .GE. EARLY .OR. ADJUST .LE. ALATE)
07900         *  GO TO 701
08000          IDIR= 1
08100          IOPP= 2
08200          ICORD= J
08300          EARLY= ADJUST
08400     701  CONTINUE
08500    C
08600    C       REPORT DELAY AND STOP IF ALL TRAINS
08700    C       HAVE TERMINATED
08800    C
08900          IF( EARLY .NE. 999999.) GO TO 30
09000    C
09100    C       MARK CURRENT LEVEL AS FINISHED BY
09200    C       NEGATIVE SIGN
09300    C
09400          IF( DELAY(KLEVEL) .LT. FBEST)
09500         *  FBEST= DELAY(KLEVEL)
```

```
09600           DELAY(KLEVEL)= -DELAY(KLEVEL)
09700     C
09800     C       GO TO 32 AND SEARCH FOR BEST UNFINISHED
09900     C       LEVEL
10000     C
10100           GO TO 32
10200     30    ALATE=EARLY
10300           IF(IDIR .EQ. 1) NEXT=ICORD- 1
10400           IF(IDIR .EQ. 2) NEXT= ICORD+1
10500     C
10600     C       CONFLICT RECOGNITION IF NEXT SIDING
10700     C       OPPOSITE LOCATION IS OCCUPIED, THEN
10800     C       BRANCH
10900     C
11000           ISEARC= 0
11100           IF( MODEL(NEXT,IOPP,KLEVEL) .EQ. 0.) GO TO 14
11200           IF( UNRESV(KLEVEL) .EQ. 0) GO TO 50
11300           IF( UNRESV(KLEVEL) .EQ. IDIR) GO TO 14
11400           N=ICORD
11500           ICORD= NEXT
11600           NEXT= N
11700           N= IDIR
11800           IDIR= IOPP
11900           IOPP= N
12000           GO TO 14
12100     C
12200     C       COPY MODEL STATUS TO ANOTHER ARRAY LEVEL
12300     C
12400     50    LEVEL=LEVEL+1
12500           ITRNUM=TRNUM(ICORD,IDIR,KLEVEL)
12600           NUMOPP=TRNUM(NEXT,IOPP,KLEVEL)
12700           WRITE(2,402)ITRNUM,NUMOPP,NAME(ICORD),NAME(NEXT)
12800     402   FORMAT(' CONFLICT BETWEEN ',I2,' AND ',I2,/,
12900          * ' POSSIBLE SIDINGS ',A4,' AND ',A4)
13000           WRITE(2,400) KLEVEL, LEVEL
13100     400   FORMAT(' LEVELS ',I2,' AND ',I2,/)
13200           UNRESV(KLEVEL)=IDIR
13300           UNRESV(LEVEL)= IOPP
13400           DO 800 J=1,NSDGS
13500           DO 800 K=1,2
13600           MODEL(J,K,LEVEL)=MODEL(J,K,KLEVEL)
13700           LSIDG(J,K,LEVEL)=LSIDG(J,K,KLEVEL)
13800     800   TRNUM(J,K,LEVEL)=TRNUM(J,K,KLEVEL)
13900           DELAY(LEVEL)= DELAY(KLEVEL)
14000           KTMT=MTKT(KLEVEL)
14100           MTKT(LEVEL)= KTMT
14200           DO 801 J=1,KTMT
14300           DO 801 K=1,3
14400     801   MT(K,J,LEVEL)= MT(K,J,KLEVEL)
```

```
14500           DO 802 J=1,5
14600     802   TRVAR(J,LEVEL)= TRVAR(J,KLEVEL)
14700     C
14800     C       IF NEXT SIDING IS OCCUPIED SEARCH FOR NEXT
14900     C       BEST TRAIN
15000     C
15100     14    IF(MODEL(NEXT,IDIR,KLEVEL) .NE. 0.) GO TO 13
15200     C
15300     C     RUN TRAIN(ICORD,IDIR)
15400     C     COMPUTE ARRIVAL TIME AT NEXT LOCATION
15500     C
15600           I1= TRNUM(ICORD,IDIR,KLEVEL)
15700           IF(IDIR .EQ. 2) RTIME= BTIME(ICORD)+ STIME(NEXT)
15800           IF(IDIR .EQ. 1) RTIME= BTIME(NEXT) + STIME(NEXT)
15900           MODEL(NEXT,IDIR,KLEVEL)= MODEL(ICORD,IDIR,KLEVEL)
16000          *  + RTIME
16100           TRVAR(I1,KLEVEL)= TRVAR(I1,KLEVEL)+ RTIME*PCTVAR
16200     C
16300     C    CATCH UP DELAY    MINIMUM 10 MIN. SPACING
16400     C
16500           IF(MODEL(NEXT,IDIR,KLEVEL) .GE.
16600          * LSIDG(NEXT,IDIR,KLEVEL)+.16)  GO TO 15
16700           OFFSET= LSIDG(NEXT,IDIR,KLEVEL)-
16800          * MODEL(NEXT,IDIR,KLEVEL)+ .16
16900           DELAY(KLEVEL)=  DELAY(KLEVEL)+ OFFSET
17000           ISEARC= 1
17100           MODEL(NEXT,IDIR,KLEVEL)= LSIDG(NEXT,IDIR,KLEVEL)
17200          *  + .16
17300     15    TRNUM(NEXT,IDIR,KLEVEL)= TRNUM(ICORD,IDIR,KLEVEL)
17400     C
17500     C       CHECK FOR OPPOSING TRAIN
17600     C
17700           IF(MODEL(NEXT,IOPP,KLEVEL) .EQ. 0.) GO TO 16
17800     C
17900     C       DELAY OR BE DELAYED BY OPPOSING TRAIN
18000     C
18100           DELMT=ABS(MODEL(NEXT,1,KLEVEL)-
18200          * MODEL(NEXT,2,KLEVEL)) +.02
18300           ISEARC= 1
18400           IF(MODEL(NEXT,IOPP,KLEVEL) .LT.
18500          *  MODEL(NEXT,IDIR,KLEVEL)) GO TO 17
18600           I1= TRNUM(ICORD,IDIR,KLEVEL)
18700           I2= TRNUM(NEXT,IOPP,KLEVEL)
18800           STDA=TRVAR(I1,KLEVEL)
18900           STDB=TRVAR(I2,KLEVEL)
```

```
19000           WRITE(2,320) DELMT, I1, STDA, I2, STDB
19100    320    FORMAT(' ESTIMATED DELAY ',F8.2,/,
19200          *       ' STANDARD DEV TRN',I2,F8.2,/,
19300          *       ' STANDARD DEV TRN',I2,F8.2)
19400           WRITE(2,322) MODEL(NEXT,IOPP,KLEVEL)
19500    322    FORMAT(' 2ND TRAIN ARRIVAL ',F8.2)
19600           CALL PRPVAR(DELMT,STDA,STDB,AVG,EXPDEL)
19700           AVG= MODEL(NEXT,IDIR,KLEVEL)+ AVG
19800           WRITE(2,321) STDA, AVG, EXPDEL
19900    321    FORMAT(' ST DEV AFTER MEET',F8.2,/,
20000          *       ' COMPLETION TIME  ',F8.2,/,
20100          *       ' EXPECTED DELAY   ',F8.2)
20200           TRVAR(I1,KLEVEL)= STDA
20300           TRVAR(I2,KLEVEL)= STDB
20400           MODEL(NEXT,IOPP,KLEVEL)= AVG
20500           MODEL(NEXT,IDIR,KLEVEL)= MODEL(NEXT,IOPP,KLEVEL)
20600          *  + .02
20700           GO TO 18
20800    17     I1= TRNUM(NEXT,IOPP,KLEVEL)
20900           I2= TRNUM(ICORD,IDIR,KLEVEL)
21000           STDA=TRVAR(I1,KLEVEL)
21100           STDB=TRVAR(I2,KLEVEL)
21200           WRITE(2,320) DELMT, I1,STDA, I2, STDB
21300           WRITE(2,322) MODEL(NEXT,IDIR,KLEVEL)
21400           CALL PRPVAR(DELMT,STDA,STDB,AVG,EXPDEL)
21500           AVG= AVG+ MODEL(NEXT,IOPP,KLEVEL)
21600           WRITE(2,321) STDA, AVG, EXPDEL
21700           TRVAR(I1,KLEVEL)=STDA
21800           TRVAR(I2,KLEVEL)=STDB
21900           MODEL(NEXT,IDIR,KLEVEL)= AVG
22000           MODEL(NEXT,IOPP,KLEVEL)= MODEL(NEXT,IDIR,KLEVEL)
22100          * + .02
22200    C
22300    C        STORE MEET MESSAGE
22400    C
22500    18     MTKT(KLEVEL)=MTKT(KLEVEL)+1
22600           KTMT= MTKT(KLEVEL)
22700           MT(1,KTMT,KLEVEL)=TRNUM(NEXT,1,KLEVEL)
22800           MT(2,KTMT,KLEVEL)=TRNUM(NEXT,2,KLEVEL)
22900           MT(3,KTMT,KLEVEL)=NEXT
23000           UNRESV(KLEVEL)= 0
23100           DELAY(KLEVEL)= DELAY(KLEVEL)+ EXPDEL
23200           WRITE(2,100)TRNUM(NEXT,1,KLEVEL),
23300          * TRNUM(NEXT,2,KLEVEL),NAME(NEXT),EXPDEL
23400    100    FORMAT(' TRAIN ',I2,' MEET ',I2,' AT ',A4,
23500          * ' DELAY ',F6.2,' HOURS',/)
```

```
23600    C
23700    C       RECORD ARRIVAL TIME IN LSIDG FOR
23800    C       CATCH UP DELAY OF NEXT TRAIN
23900    C
24000     16   LSIDG(ICORD,IDIR,KLEVEL)=MODEL(ICORD,IDIR,KLEVEL)
24100    C
24200    C       CLEAR PREVIOUS LOCATION
24300    C
24400          MODEL(ICORD,IDIR,KLEVEL)= 0.
24500          TRNUM(ICORD,IDIR,KLEVEL)= 0
24600    C
24700    C       TERMINATE IF END OF LINE REACHED
24800    C
24900          IF( IDIR .EQ. 1 .AND. NEXT .EQ. 1 .OR.
25000         * IDIR .EQ. 2 .AND. NEXT .EQ. NSDGS)
25100         * MODEL(NEXT,IDIR,KLEVEL)=0.
25200          IF( ISEARC .EQ. 0) GO TO 10
25300    C
25400    C       SEARCH FOR BEST UNFINISHED LEVEL
25500    C
25600    32    BEST= 999999.
25700          DO 806 J=1,LEVEL
25800          IF( DELAY(J) .LT. 0.) GO TO 806
25900          IF( DELAY(J) .GE. BEST) GO TO 806
26000          IF( DELAY(J) .GE. FBEST) GO TO 806
26100          KLEVEL=J
26200          BEST= DELAY(J)
26300    806   CONTINUE
26400          IF( BEST .NE. 999999.) GO TO 10
26500          DO 805 KLEVEL=1,LEVEL
26600          KTMT=MTKT(KLEVEL)
26700          WRITE(2,107)
26800    107   FORMAT(/)
26900          DO 807 J=1,KTMT
27000          NEXT= MT(3,J,KLEVEL)
27100    807   WRITE(2,106)MT(1,J,KLEVEL), MT(2,J,KLEVEL),
27200         * NAME(NEXT)
27300    106   FORMAT(' TRAIN ',I2,' MEET ',I2,' AT ',A4)
27400          DELAY(KLEVEL)= -DELAY(KLEVEL)
27500    805   WRITE(2,105) KLEVEL, DELAY(KLEVEL)
27600    105   FORMAT(' LEVEL',I3,' TOTAL DELAY',F8.2,' HOURS'/)
27700          STOP
27800          END
27900          SUBROUTINE PRPVAR(OFFSET,STDA,STDB,AVG,DELAY)
28000    C
28100    C       SIMULATES IMPACTS OF UNCERTAIN RUN TIMES
28200    C       DURING TRAIN MEETS
28300    C
28400             DIMENSION CNOR(10), ARRIVE(2,10)
28500             DATA CNOR /-1.64,-1.04,-.67,-.38,-.12,.12,.38,
28600         *   .67,1.04,1.64/
```

```
28700           DO 600 J=1,10
28800           ARRIVE(1,J)=STDA*CNOR(J)
28900   600     ARRIVE(2,J)=STDB*CNOR(J)+ OFFSET
29000   C
29100   C       NOW ESTIMATE STD DEV OF 2ND ARRIVAL
29200   C       AND DELAY INCURRED
29300   C
29400           DELAY=0.
29500           SUM=0.
29600           SUMSQ=0.
29700           DO 601 J=1,10
29800           DO 601 K=1,10
29900           ARIV= ARRIVE(1,J)
30000           IF( ARRIVE(2,K) .GT. ARIV) ARIV=ARRIVE(2,K)
30100   C
30200   C       TALLY STATISTICS ON 2ND ARRIVAL
30300   C
30400           SUM=SUM+ ARIV
30500           SUMSQ= SUMSQ+ ARIV**2
30600           DEL=ABS(ARRIVE(1,J)-ARRIVE(2,K))
30700           DELAY=DELAY+DEL
30800   601     CONTINUE
30900           AVG= SUM/100.
31000           VARIAN= SUMSQ/100.-AVG**2
31100           IF(VARIAN .LT. 0.) VARIAN=0.
31200           STDEV=1.074*SQRT(VARIAN)
31300           STDA= STDEV
31400           STDB= STDEV
31500           DELAY=DELAY/100.
31600           RETURN
31700           END
EOF
```

SIMULATES IMPACT OF VARIABLE RUN TIMES

File = SPVAR/FOR LRL = 256 REC = ASCII

```
00100           DIMENSION CNOR(10),DELAY(10,10,10), DAVG(10),
00200          * PENALT(10,10), ARRIVE(2,10,10), ARAVG(2,10),
00250          * PROB(10)
00300           INTEGER*1 LTTR
00400           DATA CNOR /-1.64,-1.04,-.67,-.38,-.12,.12,.38,
00500          *  .67,1.04,1.64/
00510           DATA PROB /10*0./
00600           WRITE(1,100)
00700    100    FORMAT(' ENTER AVERAGE TRAIN SPEED')
00800           CALL URREAD(1,AVGSPD)
00900           WRITE(1,103)
01000    103    FORMAT(' ENTER RUN TIME STD DEV %')
01100           CALL URREAD(1,RSTD)
01200           WRITE(1,101)
01300    101    FORMAT(' ENTER DISTANCE BETWEEN SIDINGS')
01400           CALL URREAD(1,D)
01410           WRITE(1,104)
01420    104    FORMAT(' ENTER NUMBER OF SIDINGS')
01430           CALL UIREAD(1,NSDGS)
01500           WRITE(1,102)
01600    102    FORMAT(' ENTER OFFSET TIME OF TRAIN #1 VS TRAIN #2')
01700           CALL URREAD(1,OFFSET)
01800    C
01900    C      COMPUTE ARRIVAL TIME MATRICES
02000    C
02100           DO 600 J=1,10
02200           T=(1+RSTD*CNOR(J))*D/AVGSPD
02250           ARRIVE(1,J,1)=0.
02300           ARRIVE(2,J,NSDGS)=OFFSET
02350           DO 600 K=2,NSDGS
02400           L=NSDGS+1-K
02450           ARRIVE(1,J,K)=ARRIVE(1,J,K-1)+T
02500    600    ARRIVE(2,J,L)=ARRIVE(2,J,L+1)+T
02510           ARAVG(1,1)=0.
02520           ARAVG(2,NSDGS)=OFFSET
02530           DO 604 K=2,NSDGS
02540           L=NSDGS+1-K
02550           ARAVG(1,K)=ARAVG(1,K-1)+D/AVGSPD
02560    604    ARAVG(2,L)=ARAVG(2,L+1)+D/AVGSPD
02700           DO 606 L=1,NSDGS
02800           LTTR=64+L
02900    606    WRITE(1,300)LTTR,((ARRIVE(J,K,L),K=1,10),J=1,2)
03000    300    FORMAT(' ARRIVALS AT ',A1,2(/,1X,10F7.2))
03100    C
03200    C      COMPUTE DELAY MATRICES
03300    C
03310           DO 602 L=1,NSDGS
03320           DAVG(L)=ABS(ARAVG(1,L)-ARAVG(2,L))
03400           DO 602 J=1,10
03500           DO 602 K=1,10
03700    602    DELAY(J,K,L)=ABS(ARRIVE(1,J,L)-ARRIVE(2,K,L))
03800           DO 601 L=1,NSDGS
```

```
03900                LTTR=64+L
04000    601         WRITE(1,303)LTTR,((DELAY(J,K,L),K=1,10),J=1,10)
04100    303         FORMAT(' DELAYS AT ',A1,10(/,1X,10F7.2))
04200    C
04300    C           SELECT MEET LOCATION TO MINIMIZE DELAY TO
04400    C           AVERAGE TRAIN
04500    C
04600                DMIN=99999.
04700                DO 607 L=1,NSDGS
04800                IF(DAVG(L) .GT. DMIN) GO TO 607
04900                DMIN=DAVG(L)
05000                IMEET=L
05100    607         CONTINUE
05110                LTTR=64+IMEET
05210                WRITE(1,203) LTTR
05310    203         FORMAT(/,' MEET PLANNED AT ',A1/)
05400    C
05500    C           NOW COMPUTE PENALTY MATRIX
05600    C
05800                DO 603 J=1,10
05900                DO 603 K=1,10
05910                DMIN=99999.
05920                DO 609 L=1,NSDGS
05930                IF( DELAY(J,K,L) .GT. DMIN) GO TO 609
05940                IMIN=L
05950                DMIN=DELAY(J,K,L)
05960    609         CONTINUE
06400                PENALT(J,K)= DELAY(J,K,IMEET)-DMIN
06500    603         PROB(IMIN)=PROB(IMIN)+.01
07100                WRITE(1,307) (PROB(L),L=1,NSDGS)
07200    307         FORMAT(' SIDING PROBABILITIES ',/,10F7.2)
07300                WRITE(1,306)((PENALT(J,K),K=1,10),J=1,10)
07400    306         FORMAT(' PENALTIES',10(/,1X,10F7.2))
07410
07500    C
07600    C           NOW COMPUTE STATISTICS
07700    C           - DELAY OF AVERAGE TRAIN AT BOTH LOCATIONS
07800    C           - EXPECTED DELAY, CONSIDERING VARIANCE
07900    C           - PENALTY FOR FIXING MEET AT ONE LOCATION
08000    C              IN ADVANCE
08100    C
08200                WRITE(1,200) DA
08300                WRITE(1,205) DB
08400    200         FORMAT(' AVG TRAIN DELAY AT A IN HOURS: ',F6.2/)
08500    205         FORMAT(' AVG TRAIN DELAY AT B IN HOURS: ',F6.2/)
08600                EXPA=0.
08700                EXPB=0.
08800                TOTPEN=0.
08900                DO 700 J=1,10
09000                DO 701 K=1,10
```

```
09100            TOTPEN= PENALT(J,K)/100.+TOTPEN
09200            EXPA= DELAY(J,K,1)/100.+EXPA
09300    701     EXPB= DELAY(J,K,2)/100.+EXPB
09400    700     CONTINUE
09500            WRITE(1,201) EXPA
09600            WRITE(1,212) EXPB
09700    201     FORMAT(' EXPECTED DELAY AT A  :',F6.2/)
09800    212     FORMAT(' EXPECTED DELAY AT B  :',F6.2/)
10300            WRITE(1,202) TOTPEN
10400    202     FORMAT(' DELAY PENALTY FOR LOCKING IN MEET NOW:',F
10500            STOP
10600            END
EOF
```

APPENDIX B

DIVERSION AND ACCELERATION PENALTIES

Two very important components of train delay are time loss diverting into and accelerating out of passing sidings prior to and following a train meet. Depending on the situation, these diversion and acceleration penalties may vary from less than a minute to over an hour.

Figure 1 is a time distance diagram comparing the performance of two trains, one which does not stop, and the other which is diverted through a low speed passing siding and which stops momentarily. That part of the total time loss which occurs prior to the initial stop is called the "Diversion penalty" and is labeled "D". The additional time lost accelerating out of the passing siding after the completion of a meet is called the "Acceleration penalty" and is labeled "A".

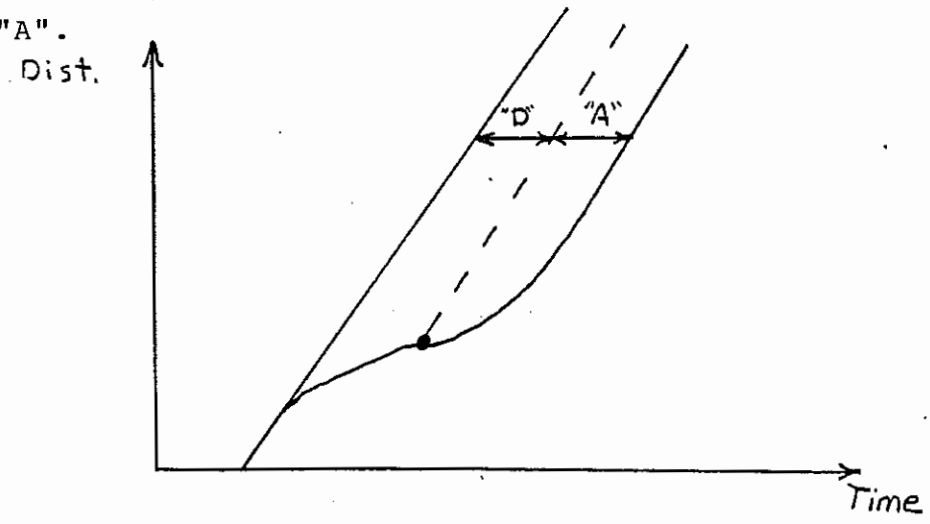


Figure 1 - Diversion Penalty, "D", and Acceleration Penalty, "A"

The following is a detailed study of the processes of diverting into and accelerating out of passing sidings and double track sections. From this study, computational procedures to estimate time loss are proposed and implemented.

Diversion Penalties

In high speed double track sections, no diversion penalties exist, since there are no reduced speeds approaching a meet location. If the meet location is a low-speed (10 or 15 mph) passing siding, one train must go into the siding. If one train is already in the siding or holding the main track from a previous meet in the same location, the other train must take the other track. If neither train has yet been assigned to the siding or main line, either of two decision rules could be used:

1. The first train to arrive at the location takes the siding.

2. The lowest priority train takes the siding.

Normally, a dispatcher should put the first train to arrive into the siding. However, if the trains are expected to arrive nearly at the same time, the dispatcher will not be able to distinguish which train will actually arrive first. He would then wish to put the lower priority train into the siding. This process can be modeled by specifying a "tightness parameter"

which serves as the cutoff point. If the meet is "tighter" than this number, e.g. five minutes, then the low priority train takes the siding. Thus, if the trains arrive two minutes apart, the lower priority train takes the siding; but if they arrive eight minutes apart, the first train would take the siding regardless of priority.

After determining which train is to take the siding, that train must be penalized for the extra time it takes to clear the main line. The penalty time is added to the train's running time as a diversion delay. The penalty has two parts:

1. The reduced speed running from a restrictive distant signal to the home signal at the entrance to the siding.

2. The reduced speed running through the switch and to the far end of the siding.

The difference between the normal mainline running time and the time required at reduced speed equals the diversion penalty. For example, suppose normal running speed is 30 mph, but the train must reduce to 15 mph while approaching and entering the siding. If the siding is one mile long and the train must reduce speed one mile before entering the siding, the train would require 8 minutes to traverse this two miles. However, at normal speed of 30 mph, only 4

minutes would be required. Therefore the diversion penalty is 8-4= 4 minutes.

The time penalty for pulling into the passing siding is proportional to the siding length, but the time required to clear for opposing traffic is proportional to the train length.

A similar penalty is incurred when the meet must occur in a branch line. The train must either back into or back out of the branch line. For a long train, the process of pulling past the branch line entrance switch, backing into the branch line at a safe speed, and closing the switch may take an hour or more. Or, if the train pulls into the branch line, it must back out after the meet has been completed, and the penalty will appear on the "acceleration" side of the meet. This penalty is generally proportional to train length, since most of the time is lost during the backup move.

Acceleration Penalties

After the two trains have met, the train that was stopped will initially suffer at least an additional one to two minutes delay. This allows for relining the switch and signal and reaction time until the train actually starts. If the train is at the end of a high speed double track section, it can immediately accelerate to full speed. If the train is in a slow

speed passing siding, it will lose additional time while pulling out of the siding and through the exit switch. It cannot begin to accelerate until the last car clears the passing siding.

The time penalty for pulling out of the passing siding is proportional to the train length. If a train is one mile long and the passing siding speed is 10 mph, six minutes would be required to pull out of the passing siding. If normal running speed is 30 mph, only 2 minutes would be required at cruise speed. Therefore, the time lost pulling out of the siding is 6-2= 4 minutes.

The penalty for accelerating to full speed depends on the weight and horsepower of the locomotives, on the size of the train and on the track gradient up or down. The only reliable way to calculate this penalty is through use of a mini-Train Performance Calculator (TPC) routine to simulate the acceleration curve for each train. A fixed penalty is not appropriate since the amount of penalty will vary between different trains and locations.

The flowchart of Figure 2 describes an algorithm to simulate the acceleration of a train, from which the acceleration penalty can be estimated. In words, the train resistance and tractive force can be estimated at any train speed. The net force is the

difference between tractive force and resistance. When divided by the train weight, yielding net force per ton, and a conversion factor of 100, the result is acceleration in miles/hour per second. The procedure uses a fixed speed increment of one mile/hour. The time required to increase train speed by one mph equals 1/acceleration. The distance traveled equals this time multiplied by the speed the train is traveling. Two typical acceleration curves estimated by mini-TPC have been plotted in Figures 3 and 4.

Table 1 is a sample output of mini-TPC showing speed, time and distance as a train accelerates to 25 miles per hour. From this output, the acceleration penalty can be calculated. The simulated train (2 GP-40 locomotives, 100 loaded hopper cars 110 tons per car) required 5.18 minutes and traveled 1.17 miles before it reached normal speed of 25 mph. At 25 mph, only 2.81 minutes would have been required to travel 1.17 miles. Therefore the acceleration penalty is 5.18-2.81= 2.37 minutes.

To estimate the penalty for a train accelerating to 25 mph, the program would have to loop 25 times, since it uses a speed increment of one mph. This can be time consuming. A TRS-80 microcomputer, utilizing a Z-80 processor running at 2.03 MHz, with a Microsoft FORTRAN compiled program, requires 2.3 seconds to

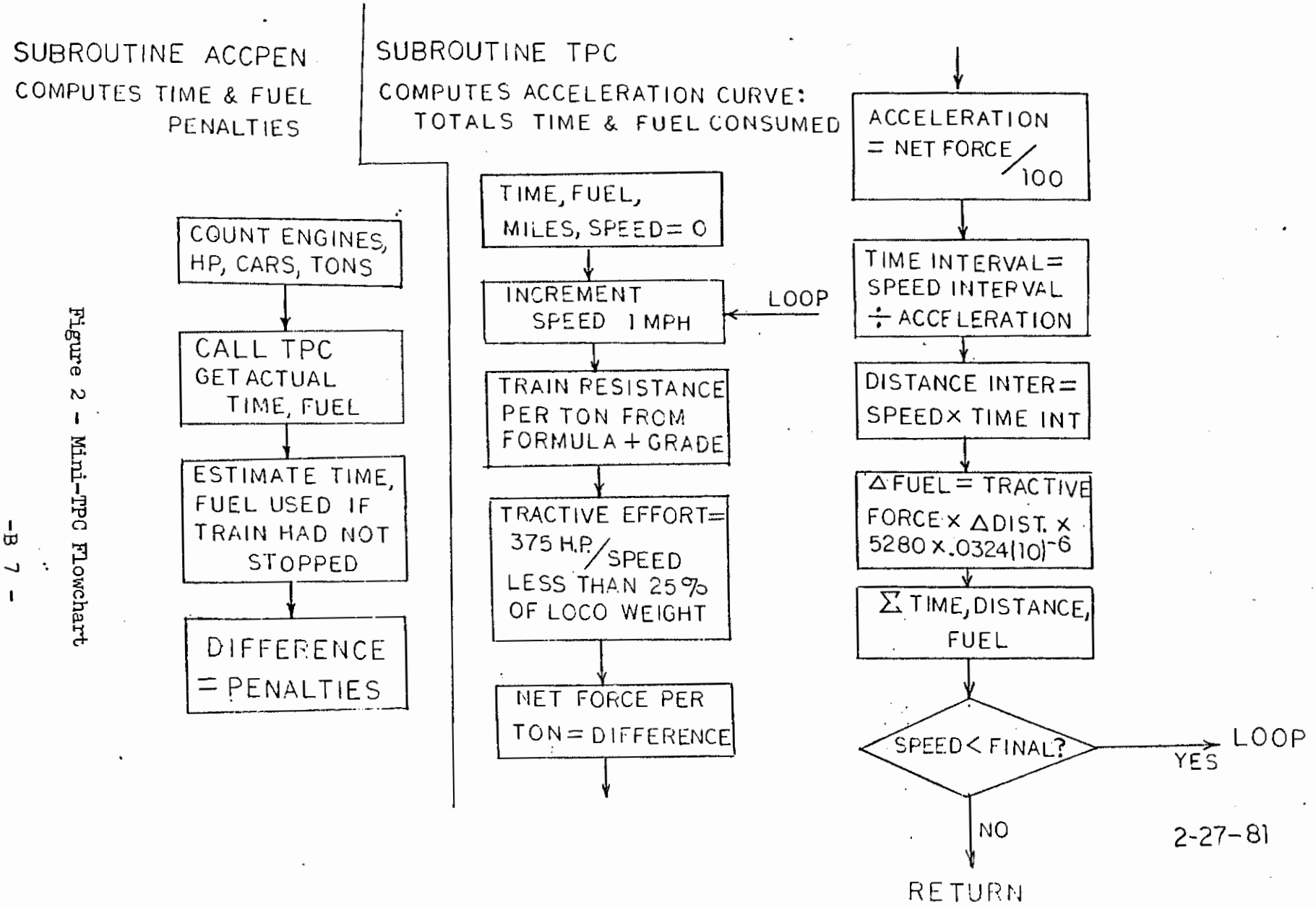


Figure 2 - Mini-TPC Flowchart

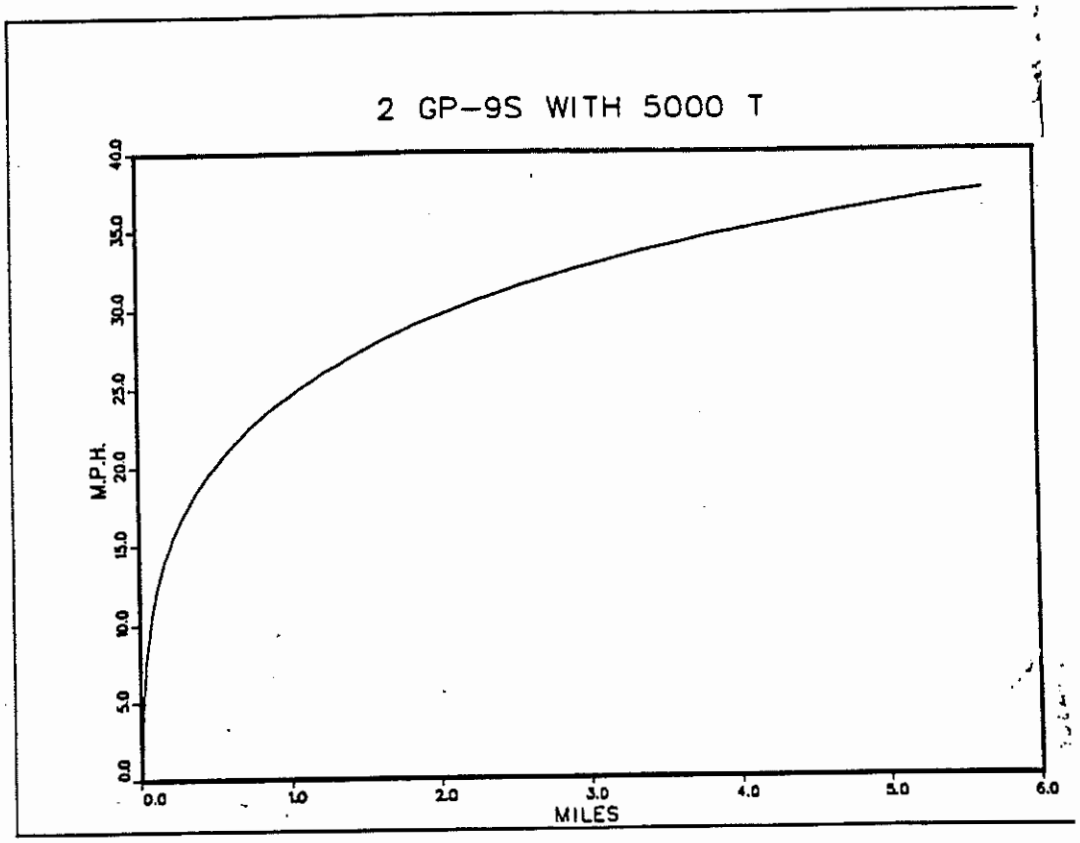
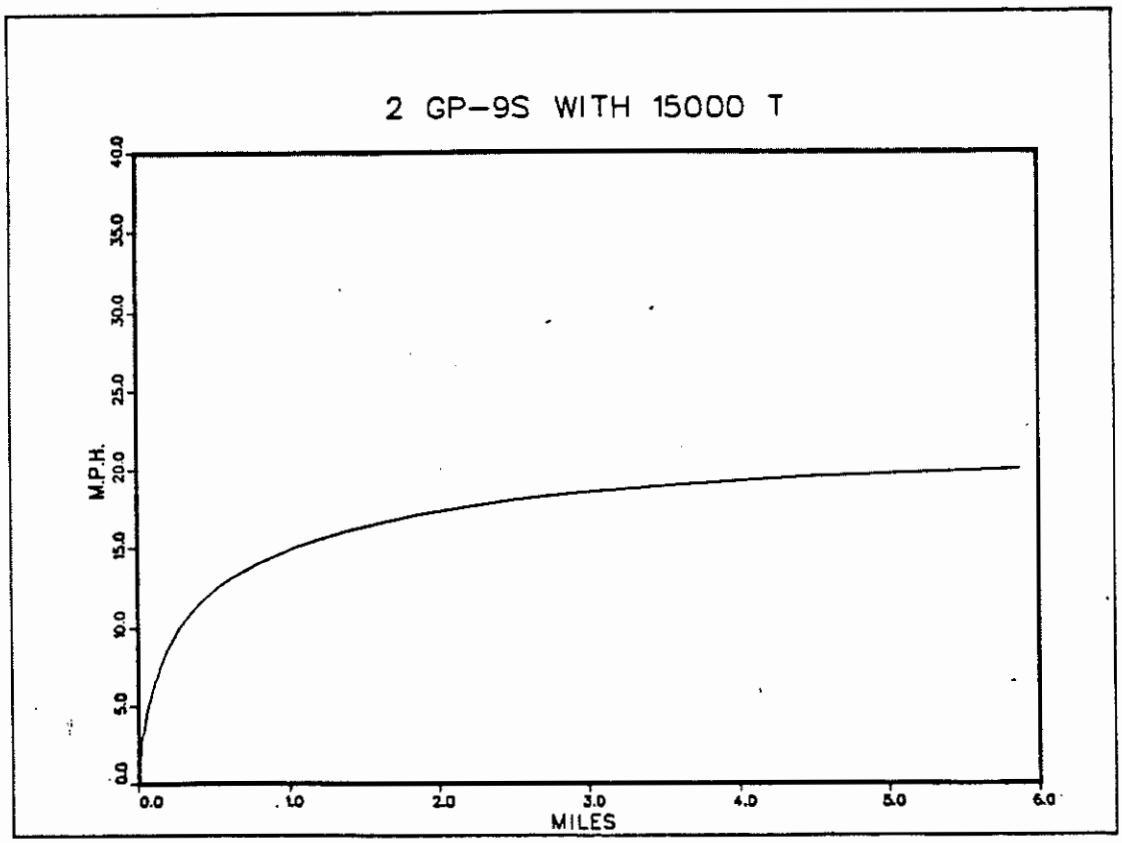


Fig. 3 and 4 - Acceleration Profiles from rest



TIME	MILES	SPEED	FUEL
0.00	0.00	0.00	0.00
.18	.00	1.00	.03
.37	.01	2.00	.13
.55	.01	3.00	.30
.73	.02	4.00	.53
.92	.04	5.00	.84
1.10	.06	6.00	1.21
1.28	.08	7.00	1.65
1.47	.10	8.00	2.15
1.66	.12	9.00	2.73
1.84	.15	10.00	3.37
2.03	.19	11.00	4.09
2.22	.22	12.00	4.88
2.40	.26	13.00	5.74
2.59	.30	14.00	6.67
2.78	.35	15.00	7.67
2.97	.40	16.00	8.75
3.16	.45	17.00	9.90
3.36	.51	18.00	11.12
3.56	.57	19.00	12.44
3.78	.64	20.00	13.87
4.02	.73	21.00	15.40
4.28	.82	22.00	17.05
4.56	.92	23.00	18.83
4.86	1.04	24.00	20.74
5.18	1.17	25.00	22.80

Table 1 - Acceleration Profile
for 2 GP-40's with 100 loads

evaluate each acceleration penalty to 25 mph. While this may not seem like much time, consider that acceleration penalties may need to be estimated thousands of times per day on a typical real-time system. Can a more efficient algorithm be found to estimate acceleration penalties?

On level track, a relatively simple formula can be used in place of the mini TPC routine. This formula was derived from the mini TPC results plotted on the four speed vs penalty graphs, Figures 5, 6, 7, and 8. These graphs show the acceleration penalties for trains with two GP-40 locomotives, and 50, 100, 150 and 200 loaded hopper cars. The graphs show that the acceleration penalty is directly proportional to final train speed through a large range of speeds (0-25 mph) and the entire simulated range of weight to power ratios. The penalty plots as a straight line; the only difference between graphs is the slope of the line.

Figure 9 shows the amount of penalty at 25 mph as a function of the weight to power ratio. Any speed less than 25 mph can be estimated from this chart simply by making a proportional reduction in the estimated acceleration penalty. This is because of the observed linear behavior of the acceleration penalty at speeds less than 25 mph.

Figure 9 can be approximated by a quadratic

function:

Penalty= .7887 W + .2529 W^2

where W= the train's weight to power ratio

= Tons/HP

This function was estimated by taking three points on the curve and finding, by algebra, a quadratic function which passes through all three points. The functional form:

Penalty= A W^2 + B W + C

was assumed. Since the curve passes through the origin (0,0), the constant term C equals zero. Taking the two points:

Wt/Power= 1.876 Penalty=2.37 minutes

Wt/Power= 3.709 Penalty=6.41 minutes

The values of A and B can be estimated:

P= A W + B W^2

2.37 = 1.876 A + 3.519 B (3.519= 1.876^2)

6.41 = 3.709 A + 13.757 B

multiply the first equation by 1.977 and eliminate:

4.69 = 3.709 A + 6.957 B

- - - - - - - - - - - - - -

1.72 = 6.8 B

therefore B= .2529

hence A= .7887

The values predicted by this quadratic equation and the true mini-TPC estimate are always within 10% of each other. They are within 3% of each other when the weight to power ratio exceeds 1.5. Since the curve flattens out at extremely low weight/power ratios, apply a minimum penalty of .69 minutes.

A more efficient estimator for acceleration penalty is:

1) Estimate the 25 m.p.h. penalty from:

$$\text{Penalty} = .7887\ W + .2529\ W^2$$

2) If (1) is less than .69 minutes, increase it to .69 minutes.

3) Estimate the actual penalty in proportion to the 25 mph penalty. In other words, the penalty at 20 mph is 80% of the 25 mph penalty. If a GP-40 is starting a 5000 ton train, its weight/power ratio is 1.667. The 25 mph penalty is:

$$\text{Penalty} = .7887\ (1.6667) + .2529\ (1.6667^2)$$

$$= 2.02 \text{ minutes}$$

If the final speed is only 20 mph, then the acceleration penalty is estimated as .80 x 2.02 = 1.61 minutes.

Figure 10 shows the 25 mph penalty as a function of gradient. No simple function has been found to describe the joint behavior of the acceleration

penalty as a function of both weight to power ratio and gradient. The use of the approximation formula above is recommended in the case of no gradients when the final train speed is less than 25 or 30 mph. If final train speed exceeds 30 mph, or if a significant gradient is present, use of the mini-TPC algorithm is recommended.

The FORTRAN source code for the mini-TPC program is included at the end of Appendix B.

Figures 5, 6, 7 and 8 - Acceleration Penalty in minutes for accelerating to indicated speed.

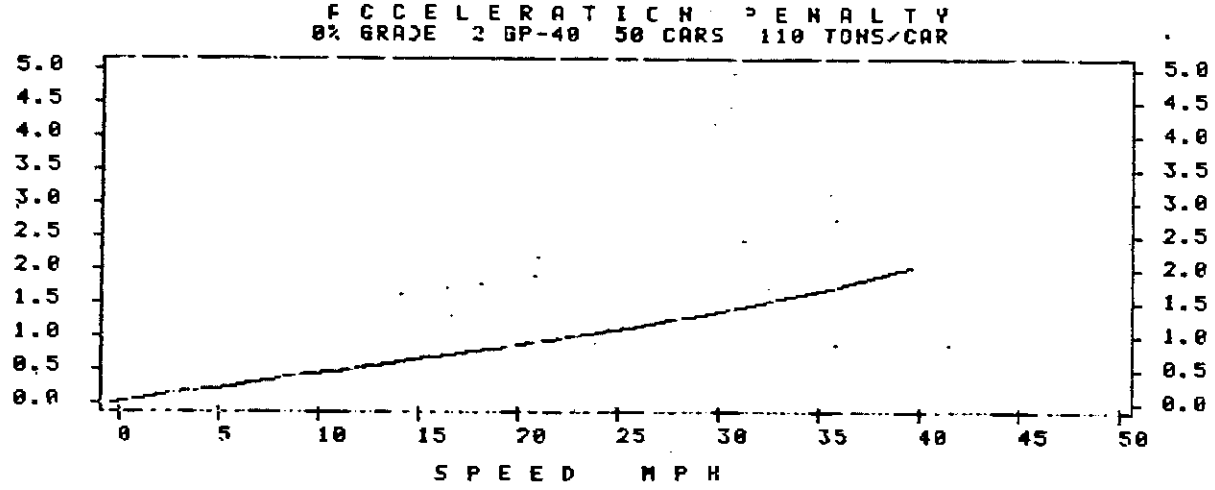


Fig 5

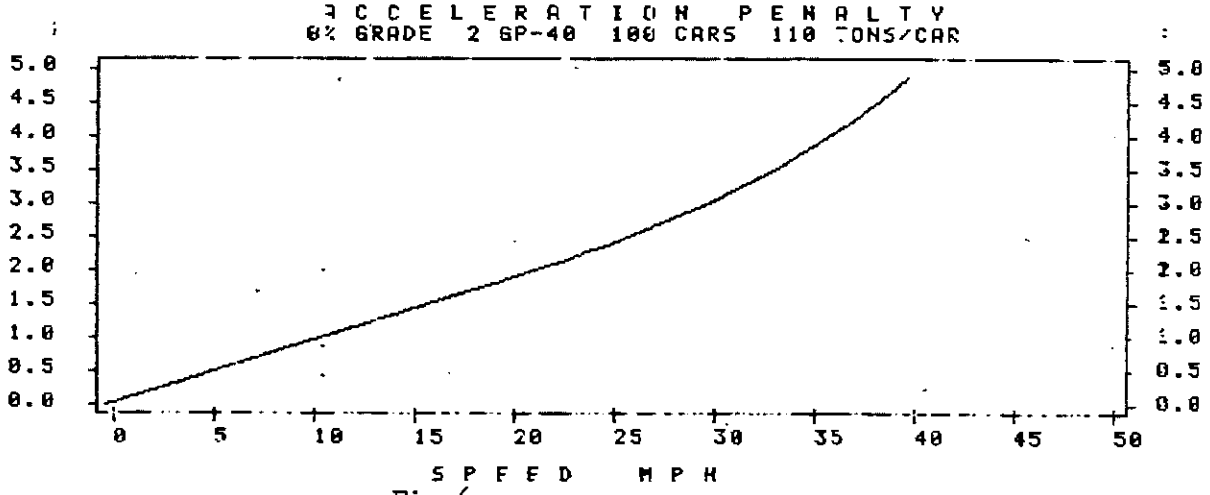


Fig 6

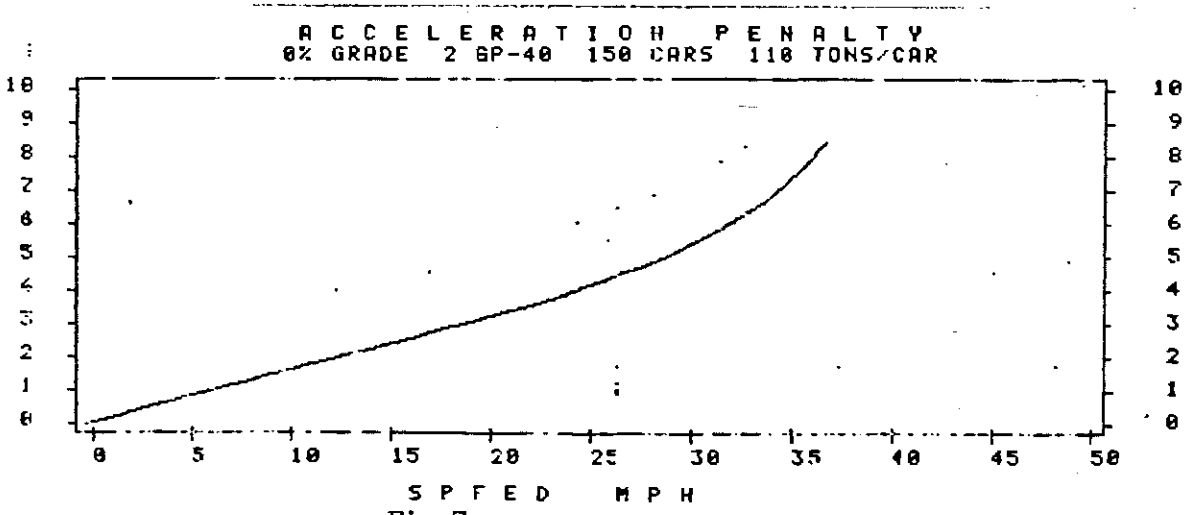


Fig 7

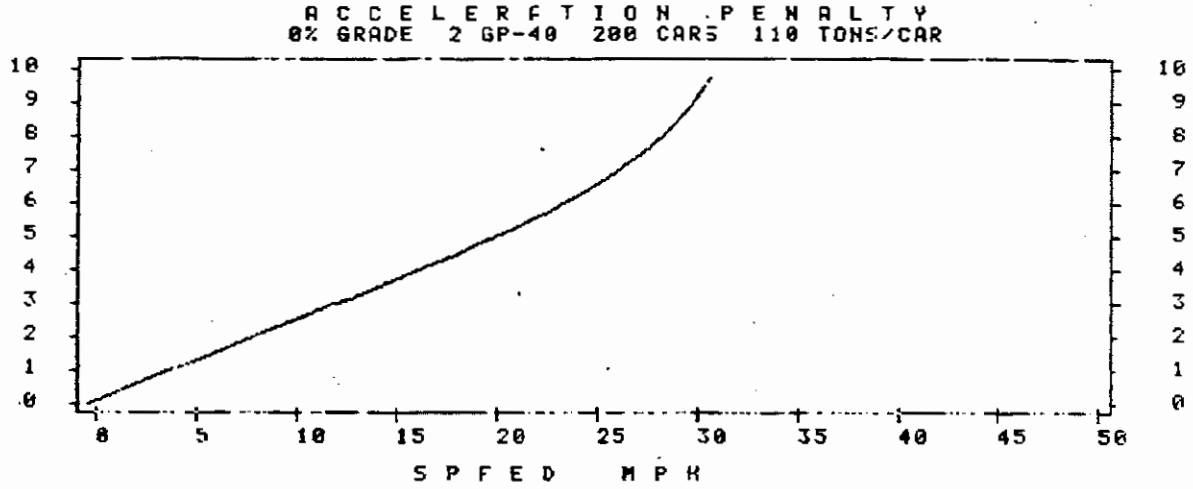


Fig 8

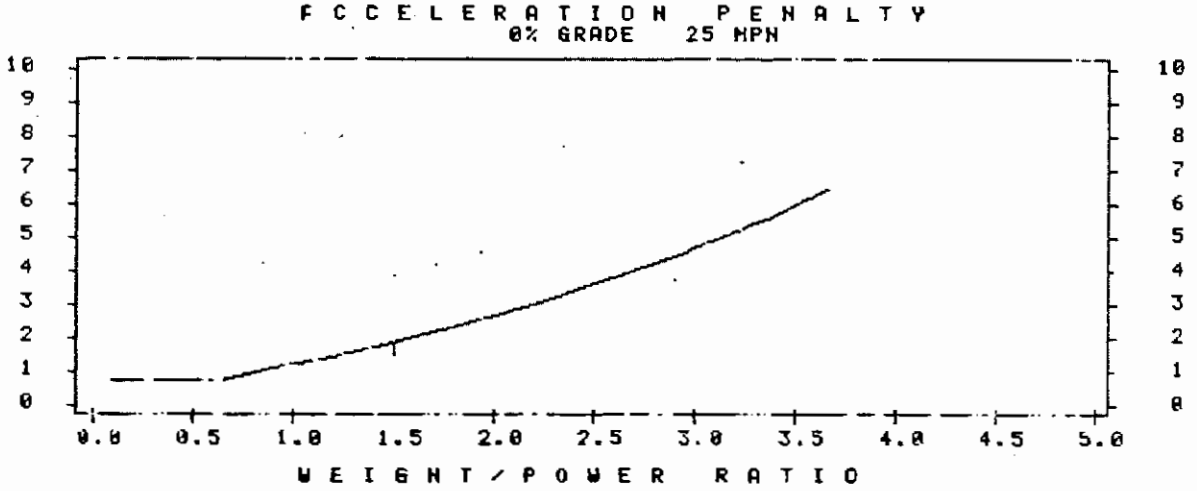


Fig 9 - 25 mph acceleration penalty as a function of weight to power ratio.

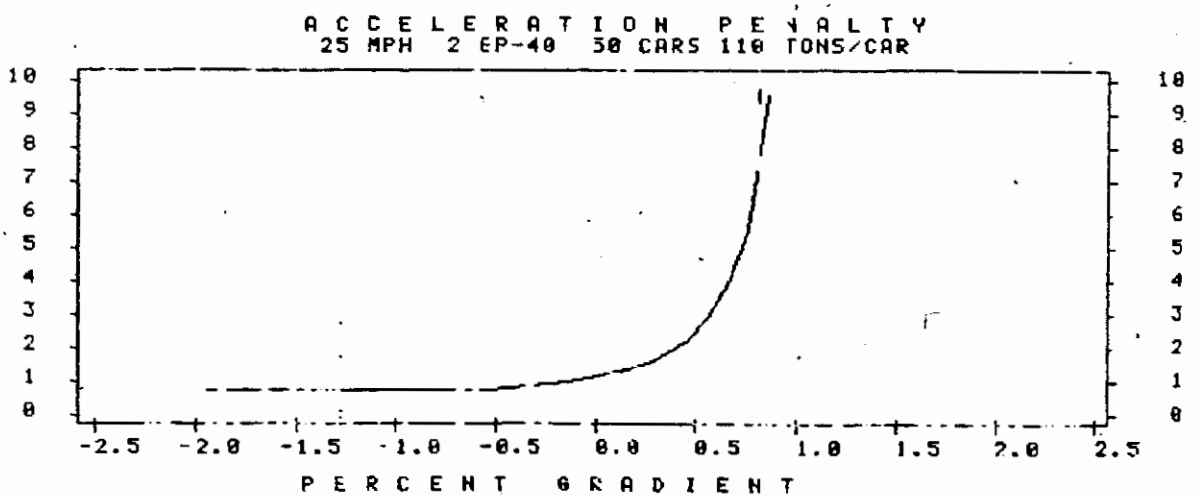


Fig 10 - 25 mph penalty for indicated train as a function of track gradient.

Mini TPC FORTRAN source code

File = TPC1/FOR LRL = 256 REC = ASCII

```
00100            PROGRAM ACCPEN
00200            IMPLICIT REAL(A-Z)
00210            INTEGER J, KK, KL
00400            WRITE(1,101)
00500    101     FORMAT(' NUMBER OF UNITS')
00600            CALL URREAD(1,ENGS)
00700            A=ENGS*3000.*375.
00800            ENGWT=256000.*ENGS
00900            WRITE(1,102)
01000    102     FORMAT(' NUMBER OF CARS')
01100            CALL URREAD(1,NCARS)
01200            WRITE(1,103)
01300    103     FORMAT(' AVG GROSS TONS/CAR')
01400            CALL URREAD(1,TONS)
01500            TONS=TONS*NCARS+ENGWT/2000.
01700            FINAL=25.
02100            CALL TPC(0.,0.,FINAL,A,ENGWT,ENGS,
02200           *  TONS,NCARS,TIME,MILES,SPEED,FUEL)
04000            STOP
04100            END
04200            SUBROUTINE TPC(GRADE,START,FINAL,A,ENGWT,
04300           * ENGS,TONS,CARS,TIME,MILES,SPEED,FUEL)
04400            IMPLICIT REAL(A-Z)
04500            DATA SINTV / 1. /
04600            TIME=0.
04700            FUEL=0.
04800            MILES=0.
04900            SPEED=START
04910            WRITE(2,111)
04920    111     FORMAT('  TIME    MILES   SPEED   FUEL')
05000    10      WRITE(2,110) TIME, MILES, SPEED, FUEL
05100    110     FORMAT(4F8.2)
05200            SPEED= SPEED+ SINTV
05300            RESIST=RTRAIN(TONS,CARS,SPEED,ENGS,ENGWT)/TONS
05400            RESIST=RESIST+GRADE*20.
05500            TSP= SPEED- SINTV/2.
05600            TFORCE= A/TSP
05700            IF( TFORCE .GT. .25*ENGWT) TFORCE=.25*ENGWT
05800            NFORCE= TFORCE/TONS- RESIST
05900    C
06000    C       RETURN IS NET FORCE IS NEGATIVE
06100    C
06200            IF( NFORCE .LT. 0.) RETURN
06300            ACCEL= NFORCE/100.
```

```
06400            IF( ACCEL .LT. .3) GO TO 13
06500            ACCEL= .3
06600            TFORCE= TONS* (30.+ RESIST)
06700    13      TINTV= SINTV/ACCEL
06800            TIME= TIME+ TINTV/60.
06900            DIST=TSP* TINTV/3600.
07000            FUEL= FUEL+ TFORCE* DIST* 5280.*.0324/1000000.
07100            MILES= MILES+ DIST
07200            IF( SPEED .LT. FINAL) GO TO 10
07300            WRITE(2,110) TIME,MILES,SPEED,FUEL
07400            RETURN
07500            END
07600            FUNCTION RTRAIN(TONS,CARS,SPEED,ENGS,ENGWT)
07700            ATONS= TONS/CARS
07800            AENGWT= ENGWT/2000./ENGS
07900            RTRAIN= ENGS*(1.3*AENGWT+ 116.+.03*AENGWT*SPEED
08000         *  + .288* SPEED**2)
08100            RTRAIN=RTRAIN+ (CARS+1.)*(1.5*ATONS+72.5
08200         *  + .015* ATONS* SPEED+ .055*SPEED**2)
08300            RETURN
08400            END
EOF
```