# Modeling and Solving the Train Pathing Problem

**Yusin Lee**
**Department of Civil Engineering, National Cheng Kung University**
**Tainan 701, Taiwan**

**and**

**Chuen-Yih Chen**
**Department of Aviation and Maritime Management, Chang Jung Christian University**
**Tainan 711, Taiwan**

**Abstract**

In a railroad system, train pathing is concerned with the assignment of trains to links and tracks, and train timetabling allocates time slots to trains. In this paper, we present an optimization heuristic to solve the train pathing and timetabling problem. This heuristic allows the dwell time of trains in a station or link to be dependent on the assigned tracks. It also allows the minimum clearance time between the trains to depend on their relative status. The heuristic generates a number of alternative paths for each train service in the initialization phase. Then it uses a neighborhood search approach to find good feasible combinations of these paths. A linear program is developed to evaluate the quality of each combination that is encountered. Numerical examples are provided.

**Keywords:** train, optimization, scheduling, pathing, timetabling

## 1. INTRODUCTION

The path of a train service is the ordered set of rail segments assigned to the train trip, and the train timetable specifies the time when the train uses each rail segment. The former represents the spatial attribute of the movement of the train in time-space, while the latter represents the time attribute. These two attributes are deeply inter-related and should be considered simultaneously when developing operating plans for train systems.

Developing train paths and timetables for a rail system is a complicated task. The physical rail facility is shared by multiple trains, and the fact that trains are confined to tracks means that detailed planning is necessary to avoid conflicts and enhance efficiency. Due to its importance, related topics have attracted considerable attention in the literature. Most published results deal with the train timetabling problem (TTP), which attempts to develop timetables for train systems without considering the exact path of individual trains. Early work by Frank[1] analyzed the TTP mathematically and proposed solution methods. Optimization models for the TTP are used in a number of papers, for example [2-10]. In addition, Cordeau et al.[11] has presented a survey of relevant optimization models. Due to the complexity of the TTP, most contributions are limited to simplified models or small instances. In particular, most papers focus on single track rail lines or one-way tracks. Some results take into consideration the track capacity constraint, which requires that the meeting and overtaking between trains to only occur within stations. Station capacity is also considered by some models.

There have been few studies concerning the train pathing problem (which assigns trains to tracks and determines their times as well). Carey[12] proposed a mixed integer program to solve for the paths of trains in a one-way-track system. The numerical example provided in Carey's paper has 10 nodes, 28 links, and 10 train services and requires significant amount of time to solve. In another article, Carey[13] has extended the model from one-way to two-way-tracks. The resulting model is also a mixed integer program, which the author reasoned is easier to solve than that of his earlier model[12], but this newer study did not provide testing results. Other works have considered the problems of routing trains through stations, e.g. [14-16]. The problems studied in these papers are different from that of ours in nature, scope and scale.

This research was motivated by the operations of the Taiwan Railways Administration (TRA). In a highly developed and densely populated country like Taiwan, rail systems are mainly multi-track networks. Lines between stations as well as within stations can be bi-directional. Inter-station distances are short; headways are tight with a train every few minutes. At stations there is often a choice of up to four tracks with platform access at which a train can stop, sometimes together with additional tracks that do not have platform access. The tracks can be one-way or two-way. In such a busy and complicated system, conflicts between trains are widespread and interdependent, and many factors have to be considered in detail in the optimization model to achieve useful results.

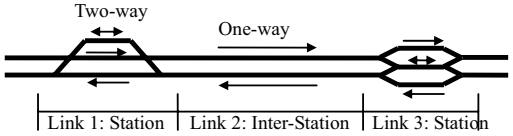
In this paper, we propose a heuristic to solve for a set of train paths and a timetable for such a complicated system as described above. The heuristic includes a number of realistic factors that are important in practice and have not been covered in previous papers. For any given link, the same train can have different dwell times depending on the assigned track, and different trains can have different track preferences. The minimum clearance time (headway) between consecutive trains also depends on the relative status of the trains as well as the track layout, i.e., whether the two trains travel in the same direction, if they use the same track, or if the two different tracks they use cross each other. The rule that only one train can occupy one block at any time is also observed. The optimization objective is to generate a timetable as close as possible to the given ideal timetable.

This paper is divided into five sections. Following this introduction, section two defines the problem studied in this research. Section three explains the solution heuristic in detail. Section four presents computation results, followed by conclusions in the final section.

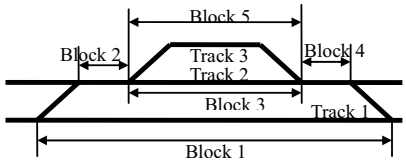
## 2. PROBLEM DESCRIPTION

In our model, we view the railroad as a collection of links,

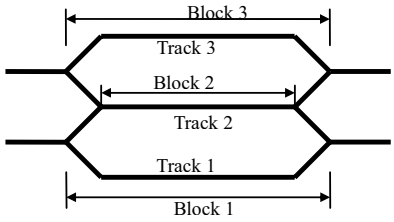
tracks, and blocks as illustrated in Figure 1. A link is either a segment of railroad between consecutive stations or the segment of the railroad within a station. The links are connected together serially as illustrated in Figure 1(a). Each link can have one or more parallel tracks as illustrated in Figures 1(b) to 1(e). Therefore, when a train service uses a link, it will use exactly one of the tracks in the link. Tracks are composed of one or more blocks connected serially. Within a station, a block represents a stretch of a track. For example, a block could be the segment of a track with access to a platform for passenger boarding. Between consecutive stations, a block corresponds to a section of a track, which allows no more than one train at any time. In either case, only one train can occupy a block at any point in time. If a train service uses a particular track, it will use all its blocks in a serial manner. Sometimes, two tracks cross each other. When this happens, we view the crossing point as one block which is shared by these two tracks. As illustrated in Figure 1 (b), track 2 is composed of blocks 2, 3, and 4, while track 3 is composed of blocks 2, 5, and 4. These two tracks cross at the shared blocks 2 and 4. Figure 1 (c) shows a station that also has three tracks, but none of the tracks share the same blocks. Stations of these configurations are common in the TRA system. A few rail segments in Figure 1 (c) do not belong to any block because they play no role in the heuristic we propose. Figure 1 (d) shows a 4-track station without shared blocks. Finally, Figure 1 (e) shows a stretch of railroad between the stations. This stretch of railroad has two tracks, divided into two blocks each.



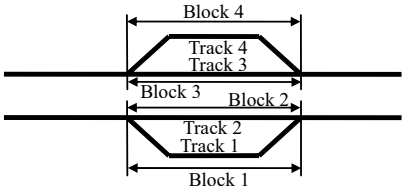

(a) A link is a segment of the railroad.



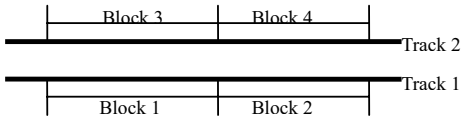

(b) A station where tracks 2 and 3 share blocks.




(c) A station where no tracks share blocks.




(d) A 4-track station without shared blocks.




(e) A stretch of railroad between stations.

Figure 1. Illustration of links, tracks, and blocks in a railroad system.

A service is defined as a train trip that travels from its origination station to its destination station, passing through a number of links. We assume that the locomotive for each service is given and fixed, but that different services can use different locomotives, resulting in different tracking power and other properties. Every service is given a target departure time, which is defined as when (in the resulting timetable) a service should plan to become ready for its first station in the trip. Besides, each service has an associated minimum and maximum dwell time at each block it might use, which is usually derived from its tracking power as well as other properties. For a block that is part of a station, the dwell time is the stopping time of the service at that station. Otherwise, the dwell time is the travel time of the train on that block. Because different trains have different travel speeds and tracking power, and different services have different stopping time at stations, the dwell times can differ between services, even for the same block. The dwell time of a train on a track is the sum of the dwell times of the train on all the blocks of that track. Note that parallel tracks of the same link can have different dwell times for the same train due to different physical conditions (e.g., speed limits imposed by turnouts) of the blocks that make up these tracks. Therefore, the dwell time of a train on a link depends on the track it uses.

A train path is an ordered set of tracks that a service can take on its trip. The ordered set of links that each service uses is given and fixed. However, because each link can contain multiple parallel tracks, it is possible for one service to have multiple alternate train paths, all using the same set of links. Each train path can also be assigned a departure time at its first station. Based on this departure time, and assuming that the dwell times of the train at each block in the path are the average of its minimum and maximum dwell times at that block, the entering and exiting time of any train path at each of its blocks can be easily calculated. These times are referred to as *pseudo times* to distinguish them from the final timetable. The departure time of a train path at its first station is referred to as *pseudo target departure time*.

The minimum clearance time between two consecutive trains that use the same link can depend on the track assignment. Suppose two trains enter the station shown in Figure 1 (c) from the left hand side on the upper line. If they both are assigned to the same track (e.g. track 3), then an appropriate headway at the track should be arranged in the timetable. If they are assigned to different tracks, there are usually no restrictions to regulate the times they enter or leave the station. (However, they still have to be properly separated on the approaching block, which they both use before entering the station.) It is more complicated when tracks cross each other. Consider the station layout in Figure 1 (b), and suppose that one train enters track 2 of the station from the left hand side on the upper line, and a second train enters track 3 from the right hand side on the lower line. Although the two trains travel in different directions, approach the station from different tracks, and use different tracks in the station, they still have to be correctly time-separated because tracks 2 and 3 cross each other. If the second train is assigned to track 1 instead, there will be no such requirement.

The required clearance time between trains that use the same

track can also depend on the relative status of the two trains. In particular, track 3 in Figure 1 (b) and track 2 in Figure 1 (c) are usually used by trains of both directions. In practice, the minimum headway between the consecutive trains that use these tracks depends on whether the two trains travel in the same direction or not.

Planners often have priorities when selecting tracks for services. For example, a higher priority service might prefer to use tracks that have fewer switches, or a cargo train might prefer to avoid tracks with platform access. For this purpose, we define a cost for every possible track of every service, with more preferred tracks having smaller costs. The cost of a path is the summation of the costs associated with all tracks that path uses. Other methods to define the costs are possible and can be easily incorporated into this model if needed.

When more than one train uses the same track, they have to do so one at a time. The relative order among the train services at the tracks is critical to train timetables. Consider two train paths *i* and *j* of different services, and consider two tracks *a* and *b* being used by both paths. The order in which *i* and *j* use track *a*, can be different from that of track *b* when they meet (if *i* and *j* travel in opposite directions) or overtake (if *i* and *j* travel in the same direction). If *a* and *b* are consecutive tracks, and the order in which *i* and *j* is using them is different, then *i* and *j* will conflict. For example, consider the railroad layout in Figure 2. There are four tracks belonging to the three links in the figure, named *a* to *d*. Suppose that there are two train paths *i* and *j* of two different services that use all three links. Since tracks *a* and *b* are consecutive, *i* and *j* should use the two tracks in the same order, i.e., if path *i* uses track *a* before path *j* does, then path *i* should also use track *b* before path *j* does, and vice versa. If *i* and *j* maintains the same order on all consecutive pairs of tracks they share, the two paths will not conflict. However, tracks *a* and *d* are not consecutive, therefore the orders *i* and *j* using these two tracks cannot determine if the pair will conflict. In principal, two train paths can reverse orders multiple times when trains re-overtake, but this is prohibited in practice, and we define this case as a conflict as well. If two paths do not conflict, the two paths are *compatible*.

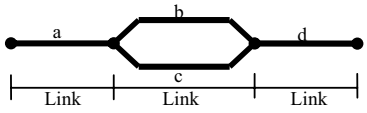



Figure 2. A railroad with 3 links and 4 tracks.

The train pathing problem studied in this research can be described as follows. Given a railroad system and a set of services, the problem aims to solve for a timetable as well as a track assignment plan for these services. Because the dwell times of individual trains as well as the required clearance time between consecutive trains depends on the track assignment, train paths have to be considered simultaneously as timetables are developed to obtain realistic results.

## 3. THE HEURISTIC

Consider a set of train paths *F*. If *F* contains exactly one path for each service, and the paths are mutually compatible, *F* is a *feasible path set*. Within a feasible path set, the exact paths of all services are known. Therefore, for any block, the services that use it are also known, and the order in which these services use this block can be determined according to the services' pseudo time. Based on this information, one can solve for the optimum schedule that corresponds to a given feasible path set with a linear programming (LP) model. Now we introduce the model, starting from a definition of the symbols used in this paper. Given information like sets and parameters are represented by upper case letters, and decision variables are in lower case letters.

$P$ the set of all services
$B$ the set of all blocks
$I$ the set of blocks that are crossing points
$G_j$ target departure time for service $j$
$D_i$ the maximum allowed delay of any service at block $i$
$Trip_j$ the minimum time for service $j$ to finish its trip
$T_{ij}^{\min}$ the minimum dwell time of service $j$ at block $i$
$T_{ij}^{\max}$ the maximum dwell time of service $j$ at block $i$
$B_j$ the set of all blocks used by service $j$
$B_j^0$ the first block used by service $j$
$B_j^F$ the final block used by service $j$
$P^-(i,j)$ the service that uses block $i$ immediately before service $j$ does
$B_{ij}^-$ the block that service $j$ uses right before it uses block $i$
$W_j$ the weight of service $j$, which represents the relative importance of the service as conceived by the planner.
$C_{ijk}$ the minimum clearance time between services $j$ and $k$ when they occupy block $i$ consecutively
$delay_j$ the total delay time of service $j$
$a_{ij}$ the scheduled time service $j$ enters block $i$
$d_{ij}$ the scheduled time service $j$ leaves block $i$
$y_{ij}$ the delay of service $j$ at block $i$
$r_j$ the absolute value of the difference between $G_j$ and the scheduled departure time of service $j$

The LP model is listed below. The objective function (1) aims to minimize the weighted sum of anomalies in time that occurred with all services, including the delays at each block, and the deviation of the scheduled departure times from the target departure times.

$$\text{Minimize} \quad \sum_{j\in P} W_j\left(r_j + 0.005d\text{elay}_j\right) \tag{1}$$

The model has nine constraints as listed below. Constraint (2) ensures that every train $j$ occupies block $i$ for at least $T_{ij}^{\min}$ time, except for the blocks that are crossing points.

$$d_{ij} \ge a_{ij} + T_{ij}^{\min} \qquad \forall i \in B_j \setminus I \qquad \forall j \in P \tag{2}$$

Constraints (3) and (4) are also related to the length of the time a train occupies a block. According to constraint (3), if service $j$ occupies block $i$ for more than $T_{ij}^{\max}$ time, the excess time will be regarded as a delay, which is limited by constraint (4) to no more than $D_i$.

$$d_{ij} \le a_{ij} + y_{ij} + T_{ij}^{\max} \qquad \forall i \in B_j \setminus I \qquad \forall j \in P \tag{3}$$

$$y_{ij} \le D_i \qquad \forall i \in B_j \setminus I \qquad \forall j \in P \tag{4}$$

Constraints (5) and (6) together makes $r_j$ equal to the absolute value of the difference between $G_j$ and the scheduled arrival time of service $j$ at its initial station.

$$r_j \ge a_{B_j^0 j} - G_j \qquad \forall j \in P \tag{5}$$

$$r_j \geq -a_{B^0_j j} + G_j \quad \forall j \in P \tag{6}$$

Constraint (7) ensures that there is at least $C_{i,P^-(i,j),j}$ clearance time at block *i*, where *j* is a service that uses block *i*, and $P^-(i,j)$ is the service that uses the block immediately before *j* does. The clearance time needed can depend on whether the block resides in a station, or is located between stations, or is a crossing point between tracks.

$$a_{ij} - d_{i,P^-(i,j)} \geq C_{i,P^-(i,j),j} \quad \forall i \in B_j \quad \forall j \in P \tag{7}$$

Constraint (8) ensures that any service will enter the next block as soon as it leaves the previous block. This constraint prevents any train from disappearing for any length of time.

$$d_{B^-_{ij},j} = a_{ij} \quad \forall i \in B_j \quad \forall j \in P \tag{8}$$

Constraint (9) makes the variable $delay_j$ equal to the total delay time of service *j*.

$$d_{B^F_j j} - a_{B^0_j j} = delay_j + Trip_j \quad \forall j \in P \tag{9}$$

Finally, constraint (10) limits the delay term $y_{ij}$ to non-negative values.

$$y_{ij} \geq 0 \quad \forall i \in B_j \quad \forall j \in P \tag{10}$$

This LP model has several important properties. First, because the choice of tracks at each link of each train service is given in this model, the dwell time of each individual train at each link is known and can be reflected explicitly in constraints (2) to (4). Therefore, the requirement that the dwell times of trains should depend on the tracks assigned can be correctly considered. Second, the crossing points are modeled as blocks, enabling constraint (7) to maintain proper headway between crossing trains even when they use entirely different tracks. Third, because constraint (7) is based on blocks instead of links, and because the train paths are known, the requirement that headways between trains depend on the trains' relative status can be explicitly included by using the correct $C_{i,P^-(i,j),j}$ value in constraint (7). Finally, the model does not contain any integer variables, thus it can be solved with the highly efficient simplex algorithm using commercial software (for example, CPLEX).

We are now ready to present the proposed heuristic by piecing together the components introduced above. The basic concept is as follows. First, we generate a number of possible alternate train paths for each service as well as an initial feasible path set. At the beginning of each iteration, the heuristic randomly replaces one of the paths in the current feasible path set *F* with another train path selected from the alternate paths, to reach another slightly different feasible path set *F'*. The new feasible path set *F'* can be evaluated with the LP model above according to the paths it contains. The objective function value of the LP model (regarded as a large number if the model is infeasible), together with the costs of each path in *F'*, is regarded as the weight of the set *F'*, which represents its quality. Then, a threshold accepting rule[17] accepts or rejects *F'* based on its quality. If *F'* is accepted, it will replace *F*. Otherwise, *F'* will be abandoned. In either case, the heuristic proceeds to the next iteration and repeats. When the heuristic ends, the best feasible solution ever encountered is used as the final output. Details of the heuristic are provided below.

The initialization phase of the heuristic involves generating the initial feasible path set and generating a number of alternate paths for each service. The initial feasible path set is generated by the following method. First, generate one path for each service, and then arrange the paths consecutively so that one service will depart its origination station after the previous service has arrived at its destination station. It is unlikely that the set generated by this method will have good quality, but it is always feasible. After creating the initial feasible path set, the heuristic generates a number of alternate paths for each service. The set of links used by each service is fixed, but the tracks within the links are selected randomly when the paths are generated. Let $p_s$ be the set of alternate paths of service *s*. The pseudo target departure times of the paths in $p_s$ are evenly spaced and distributed around the target departure time of *s*. Computational experiments indicate that the CPU time as well as the quality of the final solution is insensitive to the size of $p_s$, should $p_s$ remain within the range of 100 to 1500.

Following the initialization phase, the heuristic attempts to improve *F* through iterations. Recall that the initialization phase generated a set of alternate paths $p_s$ for each train service. At the beginning of each iteration, the heuristic generates a new set *F'* by slightly altering *F*. To do this, it first randomly selects one service *s* and replaces its path in *F* with another path in $p_s$. This is done by repeatedly selecting alternate train paths from $p_s$ at random until a path that is compatible with all other paths in *F* is found. After obtaining a new feasible set *F'*, the new set is evaluated to determine its quality. The quality of a feasible set is represented by its weight, which is the sum of the objective function value of the corresponding LP model and the costs of each path in the set. For every block in the railway, one can determine the set of services that use the block according to the paths in *F'*, and can also determine the order by which they use the block based on the paths' pseudo time. This information is sufficient for developing the LP model corresponding to *F'*. Solving the LP models consumes the most CPU time among all components of the heuristic.

A threshold accepting rule determines if *F'* should be accepted according to its weight. The heuristic maintains a threshold value *T*, which starts at one-fifth of the weight of the initial set, then gradually decreases along the process. If the weight of *F'* is less than that of *F*, or if it is higher yet still within the threshold *T*, the new set *F'* replaces *F*. Otherwise, *F'* is abandoned and the heuristic continues on to find the next possible set. The heuristic also keeps record of the best set found. The *T* value decreases by 1% whenever the best solution is not improved for 30 consecutive iterations. Also, if it happens that the weight of the best known solution drops below 5*T*, *T* is lowered to 20% of that weight. The heuristic terminates when *T* drops below 1.0, or when the best solution cannot be improved for 300 consecutive iterations after a minimum of 5000 iterations are completed. One can also set an upper bound on the total number of iterations if desired. Figure 3 presents a simple flowchart of the entire solution process.

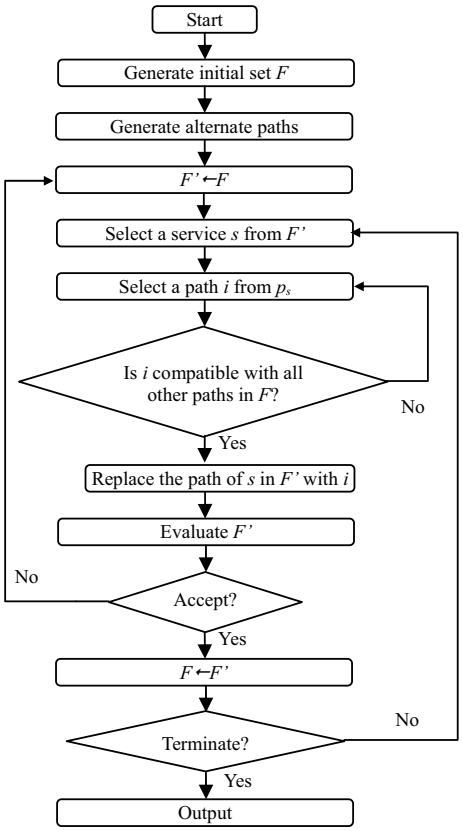



Figure 3. The solution heuristic flowchart.

## 4. COMPUTATIONAL RESULTS

This section presents some computational experiences. The two examples are prepared based on a stretch of railroad operated by TRA. The total length is 189.2 Km and has 40 stations. The stations have four types of track layouts, including the three types shown in Figure 1 and a simple, 2-track type. The railroad is double-tracked between all stations unless stated otherwise. Most of the tracks between stations have only one block, but some are composed of two blocks, allowing two trains of the same direction to pass between the two stations simultaneously. Details of the railroad configuration are presented in Table 1. Both examples are solved with a C++ language code using the CPLEX 9.0 callable library. They are tested on a computer equipped with a Pentium Pro central processing unit running at 3.2 GHz clock speed and 1 gigabytes of random access memory. The operating system is Windows XP.

Table 1. Railroad configuration used in the examples.

| Station ID | Tracks in station | Distance to next station (km) | Blocks between next station |
|---|---|---|---|
| 1 | 2 | 6.6 | 2 |
| 2 | 2 | 4.6 | 1 |
| 3 | 2 | 3.5 | 1 |
| 4 | 3 | 3.4 | 1 |
| 5 | 2 | 3.8 | 1 |
| 6 | 2 | 4.3 | 1 |
| 7 | 3 | 5.8 | 1 |
| 8 | 3 | 8.2 | 2 |
| 9 | 2 | 4.7 | 1 |
| 10 | 2 | 4.8 | 1 |
| 11 | 3 | 7.7 | 2 |
| 12 | 3 | 3.9 | 1 |
| 13 | 2 | 4.6 | 1 |
| 14 | 2 | 5.8 | 1 |
| 15 | 3 | 6.7 | 1 |
| 16 | 2 | 2.6 | 1 |
| 17 | 3 | 6.5 | 1 |
| 18 | 2 | 2.7 | 1 |
| 19 | 2 | 6.1 | 1 |
| 20 | 2 | 7.7 | 2 |
| 21 | 3 | 3.4 | 1 |
| 22 | 2 | 3.8 | 1 |
| 23 | 3 | 5.5 | 1 |
| 24 | 2 | 2.3 | 1 |
| 25 | 2 | 4.6 | 1 |
| 26 | 2 | 7.5 | 2 |
| 27 | 3 | 5.0 | 1 |
| 28 | 2 | 3.6 | 1 |
| 29 | 2 | 2.8 | 1 |
| 30 | 3 | 7.6 | 2 |
| 31 | 2 | 4.0 | 1 |
| 32 | 2 | 2.9 | 1 |
| 33 | 2 | 3.0 | 1 |
| 34 | 2 | 7.3 | 2 |
| 35 | 4 | 4.0 | 1 |
| 36 | 2 | 4.1 | 1 |
| 37 | 3 | 7.0 | 2 |
| 38 | 3 | 3.4 | 1 |
| 39 | 2 | 3.4 | 1 |
| 40 | -- | -- | -- |

The two examples have 20 services each, with equal number of them starting from either end. All services cover the entire railroad, and their target departing times are separated by one hour starting from 6 am. Not all trains stop at all stations, and the stopping patterns differ as well. For both examples, the LP model has approximately 6000 variables and 10000 constraints. Figure 4 displays the resulting train diagram for Example 1. The train diagram represents the trajectory of trains in time-space, where the horizontal axis represents time, and the vertical axis is associated with space. The horizontal lines mark the locations of the stations. The CPU time taken to solve this example is 3373 seconds.

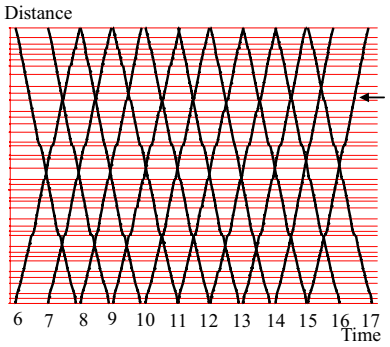



Figure 4. The train diagram for Example 1.

Example 2 differs from the previous example in that one section of the railroad between two stations is assumed to be single-tracked. The train diagram is shown in Figure 5, with the single-tracked segment pointed out. The same segment is also marked in Figure 4. By comparing the two figures one can see the effect of the difference in track configuration. In Figure 4 some trains meet at the marked segment, during which the trains use both tracks at the same time. All these meetings are

non-existent in Figure 5. Example 2 was solved in 3662 seconds. In both examples all trains departed at their target departure times.

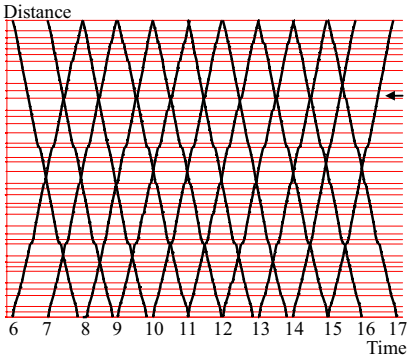



Figure 5. The train diagram for Example 2.

## 5. CONCLUSION

In this research, we developed a heuristic to solve the train pathing problem. In the initialization phase, the heuristic generates a number of alternate paths for each train service. The iteration phase that follows uses a neighborhood search process to search for a good combination of these paths. The heuristic uses a linear program to evaluate the quality of every feasible path set encountered, and uses a threshold accepting rule to determine if the new set should replace the current set. Several important practical requirements are explicitly modeled, including different train dwelling times on different tracks of the same link, correct separation of trains using tracks that cross each other, and the dependence of a minimum clearance time between two consecutive trains on their relationship. The model also arranges appropriate clearance time for trains using crossing tracks. Examples of up to 40 stations and 20 train services are solved.

## 6. REFERENCES


[1] O. Frank, "Two-way Traffic on a Single Line of Railway", ***Operations Research*** 14, 1965, pp.801-811.

[2] R. Sauder and W. Westerman, "Computer Aided Train Dispatching: Decision Support Through Optimization," ***Interfaces*** 13, 1983, pp.24-37.

[3] E. Petersen, A. Taylor, and C. Martland, "An Introduction to Computer Aided Train Dispatch", ***Journal of Advanced Transportation*** 20, 1986, pp.63-72.

[4] D. Jovanovic and P. Harker, "A Decision Support System for Train Dispatching: An Optimization-Based Methodology", ***Journal Transportation Research Forum*** 31, 1990, pp.25-37.

[5] D. Kraay, P. Harker, and B. Chen, "Optimal Pacing of Trains in Freight Railroads: Model Formulation and Solution", ***Operations Research*** 39, 1991, pp.82-99.

[6] D. Kraay and P. Harker, "Real-Time Scheduling of Freight Railroads", ***Transportation Research B*** 29, 1995, pp.213-229.

[7] M. Carey and D. Lockwood, "A Model, Algorithms and Strategy for Train Pathing", ***Journal of Operational Research Society*** 46, 1995, pp.988-1005.

[8] A. Higgins, E. Kozan, and L. Ferreira, "Optimal Scheduling of Trains on a Single Line Track", ***Transportation Research B*** 30, 1996, pp.147-161.

[9] X. Zhou and M. Zhong, "Bicriteria Train Scheduling for High-Speed Passenger Railroad Planning Applications", ***European Journal of Operational Research*** 167, 2005, pp.752-771.

[10] A. Caprara, M. Monaci, P. Toth, and P. Goida, "A Lagrangian Heuristic Algorithm for a Real-World Train Timetabling Problem", ***Discrete Applied Mathematics*** 154, 2006, pp.738-753.

[11] J.F. Cordeau, P. Toth, and D. Vigo, "A Survey of Optimization Models for Train Routing and Scheduling", ***Transportation Science*** 32, 1998, pp.380-404.

[12] M. Carey, "A Model and Strategy for Train Pathing With Choice of Lines, Platforms and Routes", ***Transportation Research B*** 28, 1994, pp.333-353.

[13] M. Carey, "Extending a Train Pathing Model From One-Way to Two-Way Track", ***Transportation Research B*** 28, 1994, pp.395-400.

[14] L. Kroon, H. Romeijn, and P. Zwaneveld, "Routing Trains Through Railway Stations: Complexity Issues", ***European Journal of Operational Research*** 98, 1997, pp.485-498.

[15] P. Zwaneveld, L. Kroon, and S. Hoesel, "Routing Trains Through a Railway Station Based on a Node Packing Model", *European Journal of Operational Research* 128, 2001, pp.14-33.

[16] M. Carey and S. Carville, "Scheduling and Platforming Trains at Busy Complex Stations", *Transportation Research A* 37, 2003, pp.195-224.

[17] G. Deck and T. Scheuer, "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing", *Journal of Computational Physics* 90, 1990, pp.161-175.