Computers & Industrial Engineering 66 (2013) 791–799

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A graph theory-based approach to route location in railway interlocking ☆

Dong Wang, Xiangxian Chen *, Hai Huang

Department of Instrumental Science and Engineering, Zhejiang University, Hangzhou, China

ARTICLE INFO

Article history:
Received 20 April 2012
Received in revised form 18 July 2013
Accepted 22 September 2013
Available online 29 September 2013

Keywords:
Railway interlocking
Graph theory
Route information
Station layout description
Matrix
Simple path

ABSTRACT

The location and verification of routes is an important component of railway interlocking logic design. The interlocking routes are typically listed manually by experienced signaling engineers, and this task is both time consuming and expensive. In this paper, an automatic, graph theory-based approach to route location and verification is presented. In this new approach, a component-based model is used to represent the topology of the station layout, and a modified matrix algorithm based on graph theory is used to locate all of the routes in a given station. This algorithm exhibits superior performance in the location and verification of routes and is universally applicable, irrespective of the station layout. When a station is modified, the designers can simply update the topological data for the station, and the new route information can be obtained automatically.

1. Introduction

A railway interlocking system is a type of railway monitoring and control system that is used in railway stations to ensure train operation safety (IEEE Std 1474.1, 1999). The interlocking system prevents trains from colliding (in both side and rear impacts) by monitoring and managing the states of the signaling apparatus (for example, semaphores and switches) and the track occupancy.

In railway systems, route information is one of the most important components of the interlocking logic. A train is not allowed to move forward until the front route has opened (Basten, Bol, & Voorhoeve, 1995). The interlocking system chooses the appropriate route from the available list and opens the route after checking its conditions. The interlocking routes are typically listed manually by experienced signaling engineers, and this task is both time consuming and expensive. Moreover, the manual approach introduces human error into the interlocking system. A search-based method for automatic interlocking route location has been introduced by Chen, He, & Huang, 2011a, 2011b. However, this method also suffers from low efficiency, particularly for large stations.

This paper proposes a new approach to route location for a given station, consisting of a component-based description of the station layout, a modified matrix algorithm based on graph theory and a route verification method designed for railway interlocking. This approach has the advantages of efficiency and universality. When a station is modified, the designers can simply update the topological data for the station, and the new route information can be computed using the automatic method. Furthermore, the route location efficiency is enhanced by the modified matrix algorithm.

2. Station layout description

In this section, a topology graph (TG) is introduced to describe the layout of a given station based on component models. Once the topology graph is constructed, all of the route information for this station can be obtained using graph theoretic methods.

2.1. Component-based topology model

The signal layout of a railway station typically consists of four types of elements: semaphores, switches, straight track sections and point sections (Chen et al., 2011b). The definitions of these elements are as follows:

i. Semaphores, also known as signals (Hansen, 1998), are placed at track segment boundaries to control the movement of trains originating from the guard direction of the

☆ The manuscript was handled by the Area Editor Qiuhong Zhao.

* Corresponding author. Present address: Department of Instrumentation Science and Engineering, Zhejiang University, ZheDa Rd. 38, Hangzhou, China. Tel.: +86 571 87953941.

E-mail address: xxchen99@gmail.com (X. Chen).

0360-8352/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2013.09.019

semaphores. A red signal typically indicates that forward movement is forbidden, while a green or yellow signal indicates that forward movement is permitted.

ii. Straight track sections are track sections that do not include switches.

iii. In this paper, point sections are defined as sections that include switches. There are two types of point sections, single-switch and multi-switch point sections, as shown in Fig. 1.

iv. A switch is the mobile portion of a point section and has two possible positions, known as the normal and reverse positions. In the normal position, trains are allowed to travel along the straight path in the point section, while in the reverse position, the branch path is allowed. The connectivity between a specified point section and its adjacent sections can be changed by flipping the switch of the point section.

Each of these components is represented as a node in the TG, and the relationships between components can be described by two types of links: connection links (CLs) and reference links (RLs).

CL edges are used to describe the connectivity relationships between semaphores, switches and straight track sections. A CL edge between two nodes indicates that these two devices are adjacent. For example, s10, w2 and t3 are all adjacent to w1, as illustrated in Fig. 2. CL edges are represented by solid lines in the TG and occur between nodes if one of the following conditions holds:

i. Two straight track sections are directly connected and there is no semaphore between them. In this case, there is a CL edge between the two straight track sections.

ii. A semaphore is present between two straight track sections. In this case, there are CL edges between the semaphore and the two straight track sections, for example, the CL edge between s9 and t3, and the CL edge between s9 and t6 in Fig. 2.

iii. A semaphore is adjacent to a point section. In this case, there is a CL edge between the semaphore and the appropriate switch in the point section, e.g., the CL edge between s10 and w1 in Fig. 2.

iv. Two point sections are directly connected. In this case, there is a CL edge between the appropriate switches in each point section, e.g., the CL edge between w1 and w2 in Fig. 2.

v. Two switches are directly connected within a single point section. In this case, there is a CL edge between the two switches, e.g., switch1 and switch2 in Fig. 1.

Reference links are used to describe the reference relationships between switches and point sections. An RL edge is always present between a switch and the point section controlled by the switch. In the TG, a switch can be connected with only one point section, but a point section may be connected with more than one switch. RL edges in the TG are represented by dashed lines.

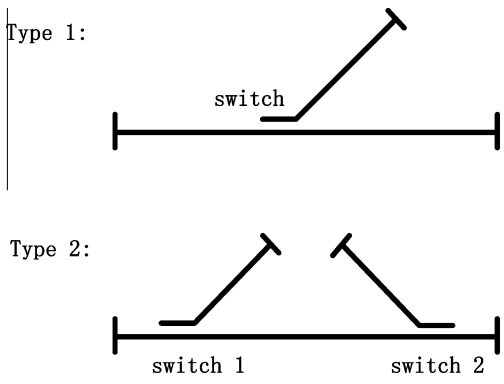

Fig. 1. Two types of point sections.

Consider the station shown in Fig. 2 as an example. By combining the two types of links, all of the station topology information can be described by a single TG, as shown in Fig. 3.

2.2. Component data structures

For computer analysis and manipulation of the topology graph, the data representation of the components is introduced in this section. Different types of components are represented by different data structures. The entirety of the station topology information can be stored in four data lists: the lists of semaphore, switch, straight track section and point section components. The data structure representing each type of component is described below.

2.2.1. Semaphore component

A semaphore component has four attributes: the semaphore ID, westbound connected component, eastbound connected component and guard direction. The semaphore guard direction has two possible values, E and W. A guard direction of E indicates that trains moving from west to east may not pass through the semaphore when it is red, while a guard direction of W indicates a similar restriction for trains moving from east to west. Consider semaphore s1, shown in Fig. 2, as an example. This semaphore is stored as [s1, null, t1, E]. The "null" value of the westbound connected component indicates that s1 has no connected component in this direction.

2.2.2. Switch component

A switch component has six attributes: the switch ID, westbound connected component through the straight track, eastbound connected component through the straight track, connected component through the diverted track, branching direction and referenced point section. The branching direction has two possible values, E and W. A branching direction of E indicates that the diverted track runs from west to east, while a branching direction of W indicates that the diverted track runs from east to west. Consider switch w1, shown in Fig. 2, as an example. This switch is stored as [w1, s10, t3, w2, E, d1].

2.2.3. Straight track section component

A straight track section component has three attributes: the section ID, westbound connected component and eastbound connected component. Consider straight track section t3, shown in Fig. 2, as an example. This section is stored as [t3, w1, s9].

2.2.4. Point section component

A point section component is a section including one or more switches and therefore has the following three attributes: the section ID, number of referenced switches and referenced switch set. Consider point section d3, shown in Fig. 2, as an example. This point section is stored as [d3, 1, w3].

2.3. Simplification of the topology graph

The scale of the topology graph is the most important factor in determining the efficiency of the graph computation and analysis. To simplify the subsequent matrix calculations (illustrated in Sections 3 and 4), it is necessary to reduce the scale of the topology graph. The simplifications in the TG do not alter the basic topology of the signaling elements and cause no loss of route information.

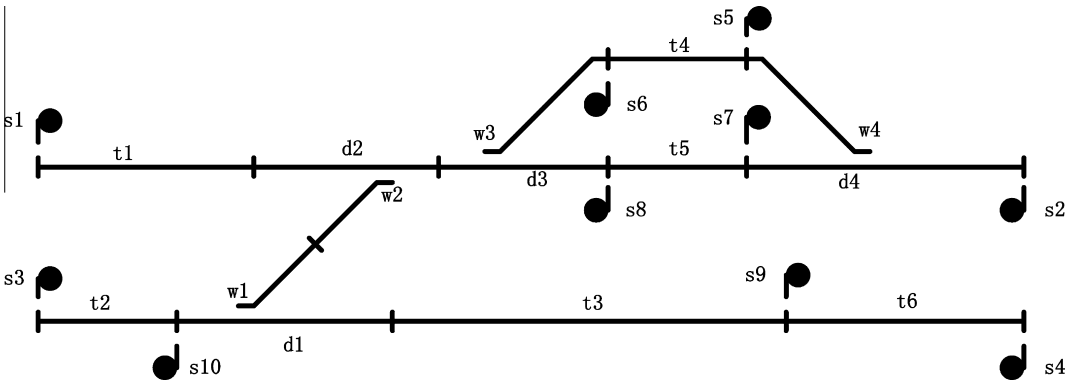

Fig. 2. Signal plane layout of a railway station.

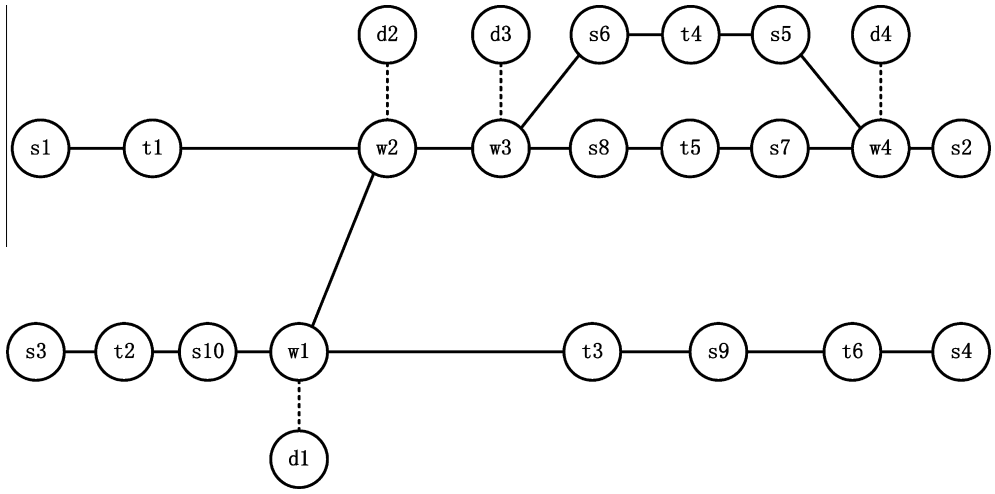

Fig. 3. Component-based topology graph of the station shown in Fig. 2.

2.3.1. Exclusion of straight track sections

A route in the interlocking system can be represented as a sequence of components from the starting semaphore to the ending semaphore. A straight track section component in a TG is simply a link between its westbound connected component and eastbound connected component. No additional signaling device is present within the straight track section. The removal of straight track sections from a TG therefore reduces the spatial complexity of the graph without interfering with route searching or identification. The following steps are performed to remove the straight track section component T [T, X, Y]:

i. Identify component X (the westbound connected component of T), and then identify the attribute of X whose value is equal to T (the eastbound connected component of X, unless X is a switch with a branching direction of E; in this case, the desired attribute may be the component that is connected through the diverted track). Replace the value of this attribute by Y. Take t3 [t3, w1, s9] in Fig. 2 as an example, where X = w1 [w1, s10, t3, w2, E, d1], Y = s9 [s9, t3, t6]. After we replace the value 't3' in X(w1) by Y(s9), X is changed to w1 [w1, s10, **s9**, w2, E, d1].
ii. Identify component Y (the eastbound connected component of T), and then identify the attribute of Y whose value is equal to T (the westbound connected component of Y, unless Y is a switch with a branching direction of W; in this case, the desired attribute may be the component that is connected through the diverted track). Replace the value of this attribute by X. For the t3 example above, after we replace the value 't3' in Y(s9) by X(w1), Y is changed to s9 [s9, **w1**, t6].
iii. Remove straight track section component T.

2.3.2. Exclusion of point sections

The point sections and switches are necessary components in the description of the station topology. However, switches and point sections can substitute for one another in the expression of the route information. Here, we select switches as the basic elements in the route descriptions. The point sections and adjacent edges can therefore be ignored in the route calculation algorithm. Once the route information has been obtained for a given station, we can easily retrieve the point sections through which each route passes by checking the reference links of the switches along the route.

The simplified version of the station topology graph (STG) in Fig. 3 is shown in Fig. 4. We can see that the number of nodes in STG is reduced from 24 to 14 compared with TG in Fig. 3. For the actual railway station layouts, more than half of the signaling elements are straight sections and point sections. So exclusion of them can largely reduce the dimension of the matrices proposed in the following sections.

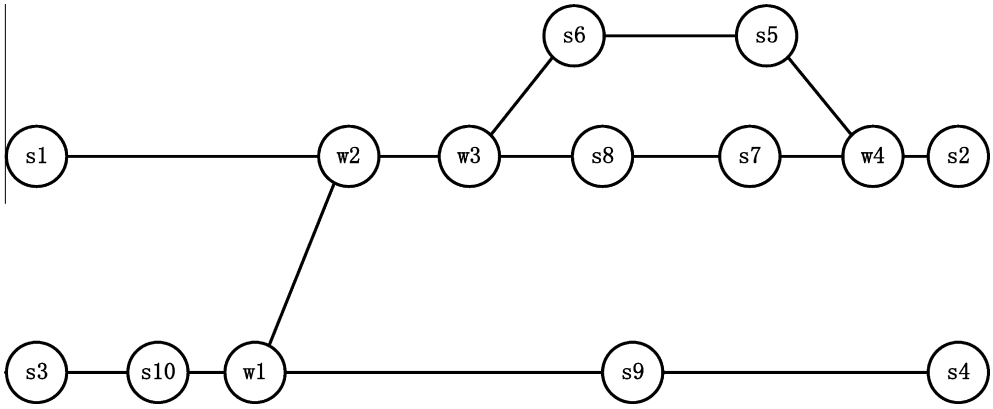

Fig. 4. Simplified station topology graph (STG) corresponding to Fig. 3.

3. Algorithm for finding simple paths in a graph

From the simplified station topology graph (STG) obtained in the previous section, the route information for the interlocking system can be generated using graph theoretic techniques.

In graph theory, a simple path refers to a path along which no vertex is traversed more than once (Hart, Nilsson, & Raphael, 1968; Ore, 1962). A non-oriented graph can be represented in matrix form, and all of the simple paths in the graph can then be obtained using matrix operations (Danielson, 1968;Duckham, 2003; Fisher & Wing, 1996; Ponstein, 1966; Sedgewick, 2003). Routes in the interlocking system have the same characteristics as simple paths, and methods designed for the location of simple paths in graphs can therefore be applied to the problem of route generation based on the STG.

Given an m-node graph G, $A = [a(i, j)]$ is the $m \times m$ adjacency matrix of G, where $a(i, j) = 1$ if an edge occurs between nodes i and j and is equal to 0 otherwise. Another $m \times m$ matrix, $B = [b(i, j)]$, can also be constructed, where $b(i, j) = j$ if an edge exists between node i and j and is equal to 0 otherwise. In addition, suppose that we have a matrix $P_n = [p_n(i, j)]$, where $p_n(i, j)$ is a polynomial used to describe the set of internal node products of all of the simple paths of length n between nodes i and j. Each term in $p_n(i, j)$ is an internal node product indicating a simple path of length n between nodes i and j. If the set of paths is ϕ, then $p_n(i, j) = 0$. In addition, we set $p_n(i, j)$ equal to 0 if $i = j$ because closed routes should not occur in railway interlocking systems. Consider the graph shown in Fig. 5 as an example. There are two distinct simple paths with lengths of 3 from node a to node e, a–b–c–e and a–b–d–e. From node a to node e, the set of paths represented by internal nodes is {bc, bd}. Therefore, $p_3(a, e) = bc + bd$ in the corresponding matrix P_3. Considering the product $BP_n = [c_{n+1}(i, j)]$, we have $c_{n+1}(i, j) = \sum_k b(i, k)p_n(k, j)$ according to the matrix operation rules. We see that $b(i, k) \neq 0$ when an edge connects node i to node k, and $p_n(k, j) \neq 0$ when a simple path of length n exists between nodes k and j. Therefore, $c_{n+1}(i, j) \neq 0$ indicates that there is at least one possible simple path of length $(n + 1)$ that begins at node i and ends at node j. In fact, the nonzero terms $c_{n+1}(i, j)$ simply describe the set of internal nodes of all possible simple paths of length $(n + 1)$ between nodes i and j. A value of $c_{n+1}(i, j) = 0$ indicates that no simple paths of length $(n + 1)$ exist between nodes i and j. As the paths represented by $c_{n+1}(i, j)$ are constructed by adding the head node i to the simple paths represented by $p_n(k, j)$, the product paths may not be simple unless node i is required not to be among the terms $p_n(k, j)$. The simple paths (including simple circuits) can be extracted by replacing i with 0 in the ith row of matrix BP_n. The main diagonal term $c_{n+1}(i, i)$ in BP_n represents the closed paths (circuits) through node i, and the circuits can be excluded from the set of simple paths of length $(n + 1)$ by setting main diagonal terms of BP_n equal to 0. These two operations are combined into one operation known as *Simple*(). That is, $P_{n+1} = Simple(BP_n)$. The matrix $P_2 = Simple(BA)$ must be considered as a special case because P_1 does not exist according to the definition of P_n. Consider the graph shown in Fig. 5 as an example, the matrices A, B, BA and P_2 are shown in Fig. 6. It is clear that to obtain all of the simple paths, we can simply iterate using the formula $P_{n+1} = Simple(BP_n)$ until $P_m = [0]$. When $P_m = [0]$ for all $q > m - 1$, P_q is also equal to [0].

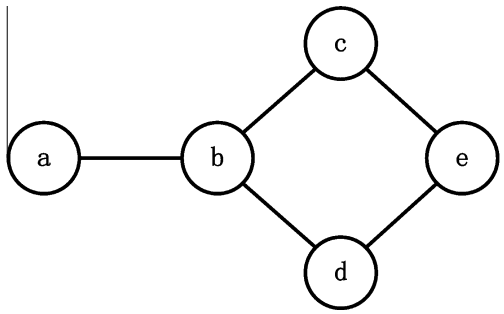

Fig. 5. A 5-node non-oriented graph.

4. Modified algorithm to obtain the route information

Because the switches have specified branching directions and the semaphores have specified guard directions, not all of the simple paths in an STG represent valid routes. The connectivity between certain components must be analyzed, and the simple paths algorithm requires modification.

4.1. The connectivity of switches

Each switch component has three connecting points, which are not equivalent in their connectivity. Consider the switch w, shown in Fig. 7, as an example. Depending on the position of the switch (normal or reverse), a train can move from point X to either point Y or point Z and from either point Y or point Z to point X through w. Consider the paths Y-w-Z and Z-w-Y. These paths are invalid regardless of the position of the switch, and all of the simple paths including these sub-paths are invalid routes. Positions similar to point X are defined as switch root positions, denoted by ***root***, and positions similar to point Z are defined as straight branch positions of the switch, denoted by ***bn***. Positions similar to point Y are defined as diverted positions of the switch and denoted by ***br***. The

A

	a	b	c	d	e
a	0	1	0	0	0
b	1	0	1	1	0
c	0	1	0	0	1
d	0	1	0	0	1
e	0	0	1	1	0

B

	a	b	c	d	e
a	0	b	0	0	0
b	a	0	c	d	0
c	0	b	0	0	e
d	0	b	0	0	e
e	0	0	c	d	0

BA

	a	b	c	d	e
a	b	0	b	b	0
b	0	a+c+d	0	0	c+d
c	b	0	b+e	b+e	0
d	b	0	b+e	b+e	0
e	0	c+d	0	0	c+d

P_2

	a	b	c	d	e
a	0	0	b	b	0
b	0	0	0	0	c+d
c	b	0	0	b+e	0
d	b	0	b+e	0	0
e	0	c+d	0	0	0

Fig. 6. Matrices A, B, BA and P_2 corresponding to the graph in Fig. 5.

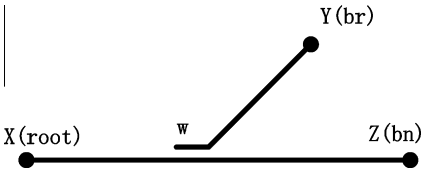

Fig. 7. Plan of a switch.

bn and ***br*** positions are equivalent in their connectivity and can therefore be known as ***b*** positions collectively (see Table 1).

To test the validity of the paths, two flags associated with switches are introduced. One flag is referred to as ***flag_in***; if a path enters a switch from the ***root*** position, then ***flag_in*** = 1, and if a path enters a switch from the ***b*** position, then ***flag_in*** = −1. The other flag is denoted by ***flag_out***; if a path leaves a switch at the ***b*** position, then ***flag_out*** = 1, and if a path leaves a switch at the ***root*** position, then ***flag_out*** = −1. The values of these flags are listed in Table 2. After the path has crossed the switch, a new flag indicating the validity of the path is obtained by multiplying ***flag_in*** and ***flag_out***. The product flag is denoted by ***flag_v***. A value of 1 for ***flag_v*** indicates that the path is valid, while a value of −1 indicates that the path is invalid.

4.2. Improvement of the simple paths algorithm

To reduce the number of invalid routes among the simple paths obtained using the algorithm introduced in Section 3, we make several modifications to the algorithm.

Table 1
Validity of paths through switches.

	Leave position	
	root	***b***
Enter position		
root	Invalid	Valid
b	Valid	Invalid

Table 2
Two types of flags associated with switches.

	root position	***b*** position
flag_in	+1	−1
flag_out	−1	+1

4.2.1. Definitions of the modified matrices

For the extraction of the correct routes from the set of simple paths given by the matrix P_n (see Section 3), several additional matrices and formulae must be defined.

Based on the matrix B introduced in Section 3, the additional matrices B^{out} and $B^{in,out}$ are introduced in this section. The matrix B^{out} is constructed by adding the ***flag_out*** flag to matrix B. For every row of matrix B^{out} that is marked with the switch ID row header, each component that is connected at the ***root*** position is multiplied by −1, based on the matrix B. The matrix $B^{in,out}$ is constructed by adding the ***flag_in*** flag to the matrix B^{out}. For every column of $B^{in,out}$ that is marked with the switch ID column header, each component that is connected at a ***b*** position is multiplied by −1, based on the matrix B^{out}. Consider the TG shown in Fig. 8 as an example. The corresponding B, B^{out} and $B^{in,out}$ matrices are provided in Fig. 9.

Similarly, based on the matrix A introduced in Section 3, the matrix A^{out} can be constructed by adding the ***flag_out*** flag to matrix A. For every row of the matrix A^{out} that is marked with the switch ID row header, each component connected at the ***root*** position is multiplied by −1, based on the matrix A. The A and A^{out} matrices corresponding to the TG in Fig. 8 are shown in Fig. 10.

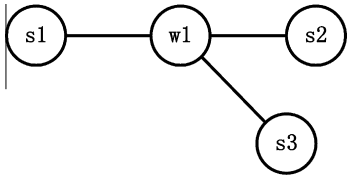

Fig. 8. A simple TG containing one switch.

$$B=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & 0 & 0 & w1\\ 0 & 0 & 0 & w1\\ 0 & 0 & 0 & w1\\ s1 & s2 & s3 & 0\end{bmatrix}\end{array}$$

$$B^{out}=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & 0 & 0 & w1\\ 0 & 0 & 0 & w1\\ 0 & 0 & 0 & w1\\ -s1 & s2 & s3 & 0\end{bmatrix}\end{array}$$

$$B^{in,out}=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & 0 & 0 & w1\\ 0 & 0 & 0 & -w1\\ 0 & 0 & 0 & -w1\\ -s1 & s2 & s3 & 0\end{bmatrix}\end{array}$$

Fig. 9. Matrices B, B^{out} and $B^{in,out}$ corresponding to the TG shown in Fig. 8.

$$A=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1\\ 1 & 1 & 1 & 0\end{bmatrix}\end{array}$$

$$A^{out}=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1\\ -1 & 1 & 1 & 0\end{bmatrix}\end{array}$$

Fig. 10. Matrices A and A^{out} corresponding to the TG shown in Fig. 8.

Given the simple paths matrix P_n, the following additional matrices are defined:

$$P^{*}_{n+1} = Simple(BP^{0}_{n}) \tag{4.1}$$

$$P^{out+v}_{n+1} = Simple(B^{out}P^{0}_{n}) \tag{4.2}$$

$$P^{out-v}_{n+1} = Simple(B^{in,out}P^{out}_{n}) \tag{4.3}$$

$$P^{0}_{n} = P^{*}_{n} - \left(\frac{P^{out+v}_{n} - P^{out-v}_{n}}{2}\right) \tag{4.4}$$

$$P^{out}_{n} = \frac{P^{out+v}_{n} + P^{out-v}_{n}}{2} \tag{4.5}$$

The '+' and '−' in the equations are defined for two matrices of the same dimensions. '+' is the operation of adding two matrices by adding the corresponding elements together and '−' is the operation of subtracting one matrix from another by subtracting corresponding elements. For example, matrices $A = [a(i, j)]$and $B = [b(i, j)]$ have the same dimensions, then $C = [c(i, j)] = A + B$, where $c(i, j) = a(i, j) + b(i, j)$, and $D = [d(i, j)] = A - B$, where $d(i, j) = a(i, j) - b(i, j)$. The '/' in the equations is the scalar division. For example, $A/2 = [0.5{\cdot}a(i, j)]$. The elements of the matrices P^{out+v}_{n} and P^{out-v}_{n} both indicate the set of simple paths of length n. However, P^{out-v}_{n} includes the ***flag_v*** information, while P^{out+v}_{n} does not. That is, all of the elements corresponding to invalid simple paths have different signs in P^{out+v}_{n} and P^{out-v}_{n}, and the sum of the corresponding elements in the two matrices is equal to 0. Furthermore, the ***flag_out*** information is also contained in the matrices P^{out+v}_{n} and P^{out-v}_{n}; the matrix P^{out}_{n} is given by $(P^{out+v}_{n} + P^{out-v}_{n})/2$. In the matrix P^{out}_{n}, all of the simple paths through switches are valid, and the elements of P^{out}_{n} also carry the ***flag_out*** attribute. A new matrix P^{out-v}_{n+1} containing both ***flag_v*** and ***flag_out*** can be obtained by multiplying P^{out}_{n} by $B^{in,out}$ from the left and appending the *Simple*() operation. The matrix P^{0}_{n} is defined as the matrix that removes the ***flag_out*** attribute from the matrix P^{out}_{n}, and the matrix P^{*}_{n} is another matrix that does not include the ***flag_out*** attribute and is defined as $Simple(BP^{0}_{n-1})$. It is readily apparent that the matrix P^{0}_{n} is equal to $P^{*}_{n} - (P^{out+v}_{n} - P^{out-v}_{n})/2$. The matrices P^{0}_{2}, P^{out+v}_{2} and P^{out-v}_{2} are given by the following three special formulae:

$$P^{0}_{2} = Simple(BA) \tag{4.6}$$

$$P^{out+v}_{2} = Simple(B^{out}A) \tag{4.7}$$

$$P^{out-v}_{2} = Simple(B^{in,out}A^{out}) \tag{4.8}$$

The product matrices P^{0}_{2}, P^{out+v}_{2}, P^{out-v}_{2} and P^{out}_{2} corresponding to the matrices in Figs. 8–10 are shown in Fig. 11. Observe that there is no invalid simple path in the matrix P^{out}_{2}.

4.2.2. Route computation

In this section, the station shown in Fig. 2 is adopted as an example to illustrate the route information computation process.

Step 1. Describe the station plane layout using a component-based topology model and simplify it (to obtain the STG shown in Fig. 4).

Step 2. Represent the k-node STG in matrix form. Generate the following $k \times k$ matrices: B, A, B^{out}, $B^{in,out}$ and A^{out}. In this case, $k = 14$.

Step 3. Perform iterative computations using formulae (4.1)–(4.8) in Section 4.2.1 to obtain the matrices P^{*}_{n}, P^{out+v}_{n}, P^{out-v}_{n}, P^{0}_{n} and P^{out}_{n} until $n = m$, where for any $q > m - 1$, $P^{out}_{q} = [0]$. It can be seen readily that $m \leqslant Length_{max} < k$, where $Length_{max}$ is the length of the longest simple path in the STG. In this case, we can see that P^{out}_{9} is equal to [0], and the iterative computation therefore terminates at $n = 9$. Save all of the P^{out}_{n} matrices generated by the iterative computation. The time complexity of the computation is $k \times k$.

$$P^{out+v}_{2}=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & w1 & w1 & 0\\ w1 & 0 & w1 & 0\\ w1 & w1 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\end{array}\qquad P^{out-v}_{2}=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & w1 & w1 & 0\\ w1 & 0 & -w1 & 0\\ w1 & -w1 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\end{array}$$

$$P^{0}_{2}=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & w1 & w1 & 0\\ w1 & 0 & w1 & 0\\ w1 & w1 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\end{array}\qquad P^{out}_{2}=\begin{array}{c} \\ s1\\ s2\\ s3\\ w1\end{array}\begin{array}{c}\begin{array}{cccc} s1 & s2 & s3 & w1\end{array}\\ \begin{bmatrix} 0 & w1 & w1 & 0\\ w1 & 0 & 0 & 0\\ w1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\end{array}$$

Fig. 11. Matrices P^{0}_{2}, P^{out+v}_{2}, P^{out-v}_{2} and P^{out}_{2} corresponding to the TG shown in Fig. 8.

Step 4. Identify the available routes in each of the P_n^{out} matrices saved in step 3. The matrix P_3^{out} shown in Fig. 12 is used as an example to illustrate the identification process. Semaphores s1, s3, s5, s7 and s9 all have a guard direction of E (east), and semaphores s2, s4, s6, s8 and s10 all have a guard direction of W (west). Each group of semaphores (east- and west-directed) can be used to generate a sub matrix of P_n^{out}; these sub matrices (shown in Fig. 12) are denoted by Re and Rw. All of the semaphores with a guard direction of E are contained in the matrix Re, and all of the semaphores with a guard direction of W are contained in the matrix Rw. If the semaphores in these sub matrices are arranged in east-to-west or west–to-east order, then it is easiest to obtain the route information from the top right or bottom left elements of the sub matrices, using the diagonal terms as a reference. The circles in Fig. 12 show all of the routes of length 3. One eastbound route, [s3-s10-w1-s9], and three westbound routes, [s2-w4-s5-s6], [s2-w4-s7-s8] and [s4-s9-w1-s10], are obtained from Re and Rw.

In the station shown in Fig. 2, there are a total of twelve routes. The route information is listed in Table 3.

4.3. Efficiency analysis

The time complexity and the efficiency of the route computation algorithm is discussed in this section.

The time complexity of each iterative computation is $O(k^2)$, which is equal to the complexity of $k \times k$ matrix multiplication, where k is the number of nodes in the STG (or the size of the STG). The maximum number of iterations is k because for $q > k$, any path in P_q^{out} contains cycle(s), which are invalid in the interlocking system. The total time complexity $T(k)$ of the algorithm is therefore given by the complexity of each iterative computation multiplied by the number of iterations, that is, $T(k) = O(k^2) * O(k) = O(k^3)$ in principle. To improve the convergence of the iterative algorithm, the algorithm has considered the connectivity of switches and removed most invalid paths from the matrices for each iteration. The number of iterations is thereby decreased, and the experiments in our study demonstrate that the iteration count is essentially proportional to the maximum route length in the STG. For the actual railway line layouts in China, the maximum route length is roughly constant, independent of the size of the STG, and the total time complexity is therefore $T(k) = O(k^2) * O(1) = O(k^2)$ in practice.

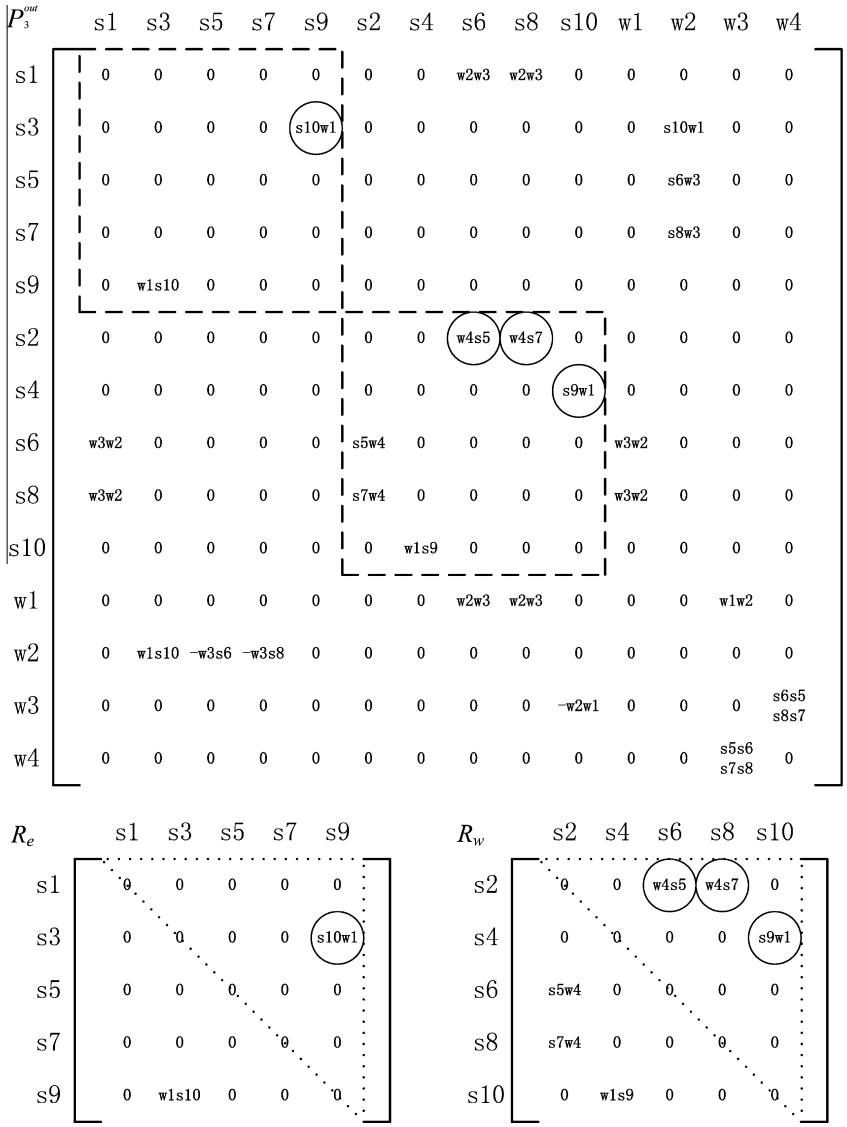

Fig. 12. The matrix P_3^{out} and its two sub matrices, R_e and R_w.

Table 3
Route information for the station shown in Fig. 2.

SN.	Route	Direction	Length	Referenced matrix
1	s3s10w1s9	E	3	P_3^{out}
2	s2w4s5s6	W	3	P_3^{out}
3	s2w4s7s8	W	3	P_3^{out}
4	s4s9w1s10	W	3	P_3^{out}
5	s1w2w3s6s5	E	4	P_4^{out}
6	s1w2w3s8s7	E	4	P_4^{out}
7	s6w3w2w1s10	W	4	P_4^{out}
8	s8w3w2w1s10	W	4	P_4^{out}
9	s3s10w1w2w3s6s5	E	6	P_6^{out}
10	s3s10w1w2w3s8s7	E	6	P_6^{out}
11	s2w4s5s6w3w2w1s10	W	7	P_7^{out}
12	s2w4s7s8w3w2w1s10	W	7	P_7^{out}

The search-based algorithm (Chen et al., 2011b) is also based on graph theory. As reported by Chen et al., 2011b, the time complexity of the search process for one semaphore is the number of possible routes multiplied by the maximum route length. The total time complexity is therefore the product of the number of possible routes, the maximum route length and the number of semaphores. The maximum route length is also roughly constant in practice. The number of possible routes is proportional to the square of the graph size (k^2), and the number of semaphores is proportional to the graph size (k). In practice, the total time complexity is therefore $O(k^3)$. However, as the size of the topology graph becomes large, especially when multiple cycles and switches are included, the lengths of the paths in the search may become extremely long. In this case, the time consumption of the search routine cannot be estimated beforehand, and the overall efficiency is difficult to ensure. However, in the algorithm proposed this paper, the time consumption of each phase is estimable.

The algorithm in this paper therefore exhibits superior performance compared to search-based algorithms in terms of efficiency.

5. Route verification

To open a particular route from the route list in Table 3, the interlocking system must first perform several verifications. More specifically, a route must meet the following necessary conditions to be opened:

i. No conflicting route is opened.
ii. The switches (if they exist) in the target route must be in the correct positions.
iii. The sections in the target route must be clear (not occupied).

It is straightforward to verify whether the sections of a route are clear. Once the route information is generated as in Table 3, the primary tasks of the interlocking system before opening a given route are conflict detection and verification of the switch positions.

5.1. Conflict detection

For a given railway station, all of the route information can be generated using the algorithm introduced in Section 4. Certain routes cannot be opened at the same time. It is clear that routes sharing the same component nodes are in conflict. Consider the routes listed in Table 3 as an example. The 1st and 8th routes are in conflict as they both contain s10 and w1 in their route information. The two routes therefore cannot be opened at the same time. Normally, the conflict information between routes is included in the interlocking logic, which is edited by experienced signaling engineers (Ho, Mao, Yuan, & Liu, 2002). In this section, an automatic conflict detection method is introduced. This method can be used in the automatic development of interlocking logic.

To identify all of the conflicts between routes in a station, we construct an $n \times d$ matrix $R = [r(i, j)]$ based on the route information, where n is equal to the number of routes, and d is equal to the number of components in the STG, as in Fig. 4. Here i is the route index, and j is the component ID. For the route information listed in Table 3, n is equal to 12, and d is equal to 14. In this matrix, $r(i, j) = 1$ if the component j is included in route i, and $r(i, j) = 0$ otherwise. The conflict matrix $C = RR^T$ is defined as an $n \times n$ matrix with $c(i,j) = \sum_k r(i,k)r(j,k)$. Both i and j in $c(i, j)$ indicate the route indexes. The $c(i, j)$ element indicates the number of common components in routes i and j. In other words, $c(i, j) > 0$ indicates that route i is in conflict with route j, and $c(i, j) = 0$ indicates that route i is not in conflict with route j. If $i = j$, then let $c(i, j)$ be equal to zero. Using the conflict matrix C, it is straightforward to obtain all of the conflict information between the routes.

The conflict detection algorithm for target route k proceeds as follows:

i. Calculate the conflict matrix C using the operation $R \times R^T$.
ii. Locate all of the nonzero terms in row k of matrix C, and check whether the conflicting routes indicated by these nonzero terms are open. If any one of the conflicting routes is opened, then the target route k cannot be opened.

The matrices R and C corresponding to the routes in Table 3 are shown in Figs. 13 and 14. Take $c(2, 9)$ in the matrix C as an example, $c(2, 9) = 2$ means the 2nd and 9th routes ([s2w4s5s6] and [s3s10w1w2w3s6s5]) in Table 3 are in conflict as they contain two common elements (s5 and s6).

5.2. Switch position verification

The verification of the switch position is based on the data representation of the switch component introduced in Section 2. The position of a switch in a route being considered for opening is correct if and only if the route passes through the switch in the straight direction and the switch is in the normal position or the route passes through the switch in the diverted direction and the switch is in the reverse position. The actual positions of the switches are recorded by the interlocking system in real time. In

R

	s1	s3	s5	s7	s9	s2	s4	s6	s8	s10	w1	w2	w3	w4
1	0	1	0	0	1	0	0	0	0	1	1	0	0	0
2	0	0	1	0	0	1	0	1	0	0	0	0	0	1
3	0	0	0	1	0	1	0	0	1	0	0	0	0	1
4	0	0	0	0	1	0	1	0	0	1	1	0	0	0
5	1	0	1	0	0	0	0	1	0	0	0	1	1	0
6	1	0	0	1	0	0	0	0	1	0	0	1	1	0
7	0	0	0	0	0	0	0	1	0	1	1	1	1	0
8	0	0	0	0	0	0	0	0	1	1	1	1	1	0
9	0	1	1	0	0	0	0	1	0	1	1	1	1	0
10	0	1	0	1	0	0	0	0	1	1	1	1	1	0
11	0	0	1	0	0	1	0	1	0	1	1	1	1	1
12	0	0	0	1	0	1	0	0	1	1	1	1	1	1

Fig. 13. The matrix R corresponding to the routes in Table 3.

C

	1	2	3	4	5	6	7	8	9	10	11	12
1	4	0	0	3	0	0	2	2	3	3	2	2
2	0	4	2	0	2	0	1	0	2	0	4	2
3	0	2	4	0	0	2	0	1	0	2	2	4
4	3	0	0	4	0	0	2	2	2	2	2	2
5	0	2	0	0	5	3	3	2	4	2	4	2
6	0	0	2	0	3	5	2	3	2	4	2	4
7	2	1	0	2	3	2	5	4	5	4	5	4
8	2	0	1	2	2	3	4	5	4	5	4	5
9	3	2	0	2	4	2	5	4	7	5	6	4
10	3	0	2	2	2	4	4	5	5	7	4	6
11	2	4	2	2	4	2	5	4	6	4	8	6
12	2	2	4	2	2	4	4	5	4	6	6	8

Fig. 14. The matrix C corresponding to the routes in Table 3.

this section, the 1st route [s3-s10-w1-s9] listed in Table 3 is taken as an example to illustrate the switch position verification process.

Step 1. Traverse all of the components in the route and search for the switch components. Switch w1 is found in this case.
Step 2. Analyze the data structure of the switch (w1) found in step 1. If one of the adjacent components (s10 or s9) in the route component sequence [s3-s10-w1-s9] is the component that is connected through the diverted track of the switch, then the correct position of the switch (w1) is the reverse position. If neither of the adjacent components (s10 or s9) is the component that is connected through the diverted track of the switch, the correct position of the switch (w1) is the normal position. In this case, neither s10 nor s9 is connected through the diverted track of w1. So the correct position of w1 is the normal position.
Step 3. Verify whether the actual position of the switch w1 is correct (using the real-time signal acquisition in the interlocking system). If the switch is not in the correct position obtained in step 2, then the target route cannot be opened.

6. Conclusion

In this paper, a new approach is introduced for obtaining route information in interlocking systems. This approach employs a component-based topology model and graph theoretic matrix method. The component-based topology model describes the station as a group of independent components, and a set of data structures is used to represent these components. To improve the route-finding efficiency given the station topology graph, a modified matrix algorithm is introduced. An automatic verification method to be applied before a route is opened is also introduced. This solution is a universal method that is applicable to variety of station layouts. For each new layout, the user can obtain the route information automatically using a single application. At this stage, the structured data of components in a station are generated from the station layout manually. This will become a bottleneck especially in the large railway station layout. Therefore, we plan to develop a graphical tool for drawing station layouts and generating structured data. The planners of railways can use this tool to edit the station layout and the structured data can be automatically generated. In addition, there is further work remaining to be done. For example, the coding and configurations of the interlocking logic software are compiled manually, leading to difficulties in updating the software when the layouts of stations change. It is therefore useful to provide designers with an integrated framework in which the route information generator and auto-coding suite for the interlocking logic are combined.

Acknowledgment

This work is partially supported by National Science and Technology Infrastructure Program of China (Grant No.: 2011BAG01B03).

References

Basten, T., Bol, R., & Voorhoeve, M. (1995). Simulating and analyzing railway interlockings in ExSpect. *IEEE Parallel and Distributed Technology, 3*, 50–62.
Chen, X., He, Y., & Huang, H. (2011a). An approach to automatic development of interlocking logic based on Statechart. *Enterprise Information Systems, 5*(3), 273–286.
Chen, X., He, Y., & Huang, H. (2011b). A component-based topology model for railway interlocking systems. *Mathematics and Computers in Simulation, 81*(9), 1892–1900.
Danielson, G. H. (1968). On finding simple paths and circuits in a graph. *Circuit Theory, IEEE Transactions, 15*(3), 294–295.
Duckham, Matt, & Kulik, Lars (2003). Simplest Paths: Automated Route Selection for Navigation. *Spatial Information Theory. Foundations of Geographic Information Science*. Berlin Heidelberg: Springer pp. 169–185.
Fisher, G. J., & Wing, O. (1996). Computer recognition and extraction of planar graphs from the incidence matrix. *IEEE Transactions on Circuit Theory, CT(13)*, 154–163.
Hansen, K. (1998). *Modeling railway interlocking systems*. Department of Computer Science, Technical University of Denmark, Citeseer.
Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics*, 100–107.
Ho, T., Mao, B., Yuan, Z., & Liu, H. (2002). Computer simulation and modeling in railway applications. *Computer Physics Communications, 143*(1), 1–10.
IEEE Std 1474.1 (1999). IEEE standard for communications-based train control (CBTC) performance and functional requirements. In *Rail vehicle interface standards committee of the IEEE vehicular technology society* (pp. 135–139).
Ore, O. (1962). *Theory of graphs* (Vol. 38). New York: Am. Math. Sot, Colloq. Pub..
Ponstein, J. (1966). Self-avoiding paths and the adjacency matrix of a graph. *SIAM Journal on Applied Mathematics*, 600–609.
Sedgewick, R. (2003). *Algorithms in Java, Part 5: Gragh algorithms*. Addison-Wesley Professional.