Adv Syst Sci Appl 2022; **02**:46–61
Published online at https://ijassa.ipu.ru.

Graph Methods for Improving the Invalid Solution of the Locomotive Assignment Problem under Time Constraints

Liudmila Zhilyakova[1*], Vasily Koreshkov[1]

[1] *V.A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia*

Abstract: Finding the optimal assignment of locomotives to fulfill freight train transportation schedule under time constraints is a complex problem, even on a linear section of the railway. In previous studies, a graph model was proposed that allows solving the assignment problem using a static graph algorithm. It was proved that finding the optimal assignment without constraints is equivalent to finding the minimal path cover of a specific acyclic graph. However, due to time constraints, the optimal solution may not be valid. This paper presents methods for modifying the found solution in order for it to satisfy the time constraints on locomotives. Two operations on path in the minimal path cover of a graph are introduced: crossover and changeover. They allow to rebuild the found solution without spoiling the admissible sequences of transportation, while the invalid ones are corrected.

Keywords: graph models, layered digraphs, minimal path cover with constraints, optimal assignment problem

1. INTRODUCTION

A wide range of methods and algorithms are used to solve problems of assigning locomotive and scheduling freight rail transportation. The papers [1–6] present different methods of solving the optimization problem for rail transportation. The paper [1] discusses graph methods and algorithms; in [2] a multi-commodity network and column generation model are considered; [3, 4] use dynamic programming methods; in [5] a mixed integer programming formulation of the problem is presented; in [6] genetic algorithms are applied. The article [7] presents an overview of the multimodal freight transportation optimization models. A model of the related problem, crew scheduling in railway transportation for a real world case, is introduced in [8]. This article proposes a two-step algorithm and investigates its efficiency.

The present research is a continuation of the study [9] where the problem of optimal locomotives assignment to fulfill the freight traffic schedule on a linear railway section is solved using a static graph algorithm. This model, in turn, originates in studies [10, 11]. These papers present the concepts and definitions that formed the basis of the graph model constructed in [9].

If there are time constraints on locomotives (early completion, late start), the first stage of the search for a solution coincides with the solution of the problem without constraints. A special acyclic graph is constructed and the minimal path cover is sought for it. Any isolated vertex is also considered a path, and therefore such a cover always exists.

Methods for constructing the minimal path cover of a directed acyclic graph are described in [12–15]. In these papers, it is shown that the problem of finding such a cover of an acyclic

*Corresponding author: zhilyakova.ludmila@gmail.com

digraph is equivalent to finding the maximum matching in a bipartite graph that is constructed from the given acyclic graph.

An acyclic graph is constructed as follows. Its vertices correspond to transportation that must be carried out at some time moments; the arcs connecting two vertices indicate that two transportations can be performed sequentially, one after the other by the same locomotive. The minimal path cover of such an acyclic graph is a solution to the original problem: it specifies locomotive assignments to trains over the entire planning horizon.

In a problem with constraints, the resulting solution may not be valid: some locomotives may not be available for some of the assigned transportations. Therefore, the solution found in the previous step is checked for validity.

In [9], methods for correcting an invalid solution are proposed, taking into account the time constraints. However, some cases where these methods do not allow finding the existing optimal solution remained beyond the scope of consideration. These cases are rare exceptions, but they also need to be handled. In this paper, we propose two operations that change the found minimal path cover of the acyclic graph. The first one is called "crossover" by analogy with the genetic recombination, following the terminology adopted in genetic algorithms [16, 17]. The second is called "changeover". They make it possible to construct new minimal covers corresponding to valid solutions to the problem with time constraints.

2. GRAPH MODEL OF THE PROBLEM

2.1. Basic Definitions

Let us consider freight transportation on a linear railway section with *k* stations. The schedule for the transportation of freight trains is given as follows. For each of the n trains, the station and time of departure and the station of arrival are specified. It is believed that locomotives carry trains at a constant speed. For each of the m locomotives, the start station and time constraints (late start, early end) are specified.

The set of stations $S = \{S_0, \ldots, S_k\}$ is linearly ordered by their arrangement.

Time moments t_i correspond to the departure and arrival of trains. The lengths of the intervals between moments may vary. The set of time moments is denoted as $\{t_0, \ldots, t_{fin}\}$ where t_{fin} is a given *planning horizon*.

The set of transportations (*train trajectories*) is denoted as $Tr = \{Tr_1, \ldots, Tr_n\}$.

The set of locomotives is denoted as $L = \{L_1, \ldots, L_m\}$.

We will assume that the velocities of all locomotives are the same and equal to v. The trains also move at speed v.

The schedule can be represented as a graph of the transportations on a plane where the abscissa axis indicates a section of the railway with stations $\{S_0, \ldots, S_k\}$, and the ordinate axis indicates time interval $[t_0, t_{fin}]$. Then train trajectories are represented as segments with angles to the y-axis v and $-v$. Their ends have coordinates $(t^i_{j_1}, S^i_{k_1})$ and $(t^i_{l_2}, S^i_{p_2})$, where indices 1 and 2 denote the start and destination points, i is the index of transportation Th_i. The initial positions of the locomotives are denoted by points on this plane (Fig. 2.1). Given locomotive L_i has coordinates (t_0, S_q) if it is available starting from time moment t_0. Otherwise it has time constraint "on the left" and its coordinates are (t_r, S_q) where $r > 0$ (locomotives $L_{10} - L_{12}$ in Fig. 2.1).

We consider a linear section of the railway, however, there may be several tracks on this section. Thus, several parallel transportations are allowed, like Tr_4 and Tr_5 in Fig. 2.1.

2.2. Reachability Graph

Consider two transportations Tr_i and Tr_j. They characterize by two pairs of coordinates: $(t^i_{h_1}, S^i_{k_1})$, $(t^i_{l_2}, S^i_{p_2})$ and $(t^j_{h_1}, S^j_{k_1})$, $(t^j_{l_2}, S^j_{p_2})$. Suppose that transportation Tr_i ends before

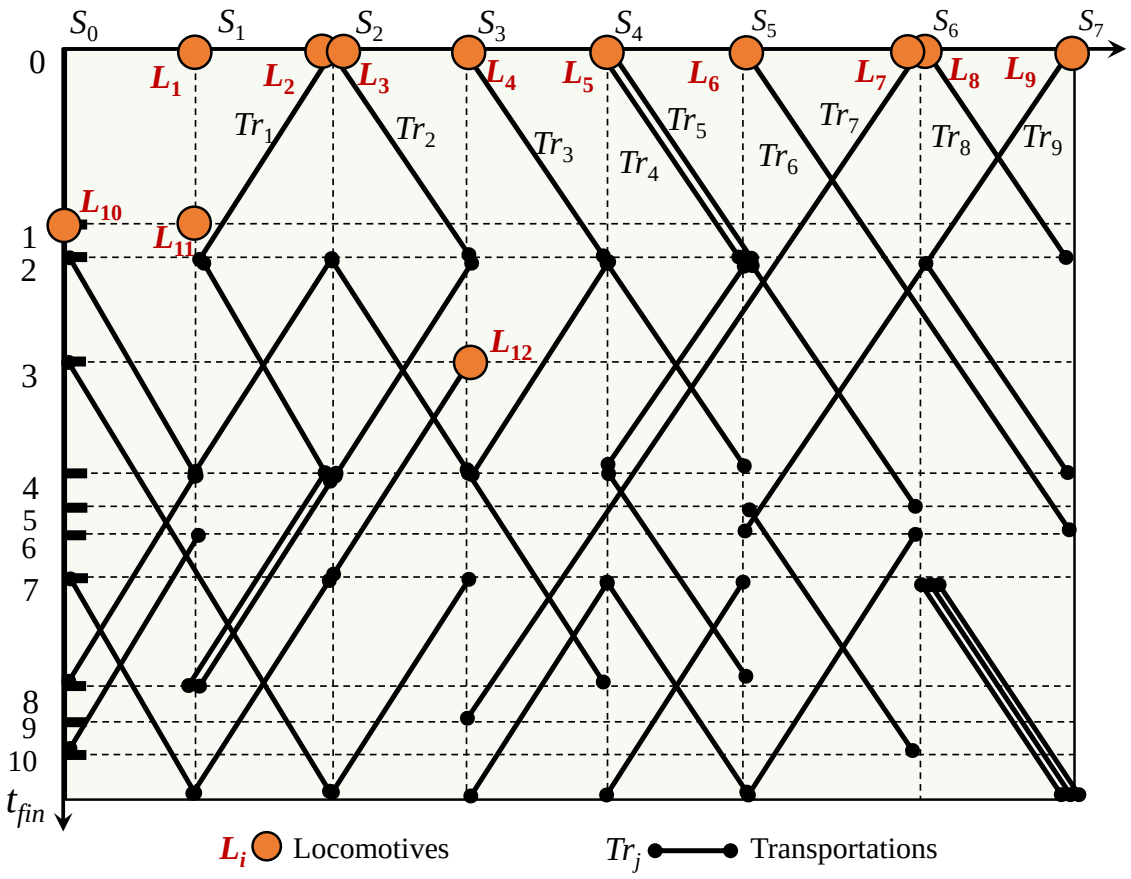

Fig. 2.1. Geometric transportation model. All locomotives and the earliest transportations are labeled

transportation Tr_j starts: $t^i_{l_2} < t^j_{h_1}$. These transportations can be carried out by the same locomotive if their coordinates meet the condition:

$$S^j_{k_1} - S^i_{p_2} \le v(t^j_{h_1} - t^i_{l_2}); \tag{2.1}$$

$$S^j_{k_1} - S^i_{p_2} \ge -v(t^j_{h_1} - t^i_{l_2}). \tag{2.2}$$

In general case, a transportation can be carried out after transportation Tr_i if it is inside the "cone of reachability" from the right end of transportation Tr_i.

$$S - S^i_{p_2} \le v(t - t^i_{l_2});$$
$$S - S^i_{p_2} \ge -v(t - t^i_{l_2}).$$

Since all coordinates of transportations are known, it is possible to construct a *reachability graph*. Its vertices are transportations, and arcs denote the reachability relation of the subsequent transportation from the previous (Fig. 2.2).

2.3. Transformations of Reachability Graph

It was shown in [9] that the graph in the form represented in Fig. 2.2 is difficult to treat "as is." For a problem without time constraints, it can be used to find a solution. However, in a problem with time constraints, the existing solution may not be found on such a graph.

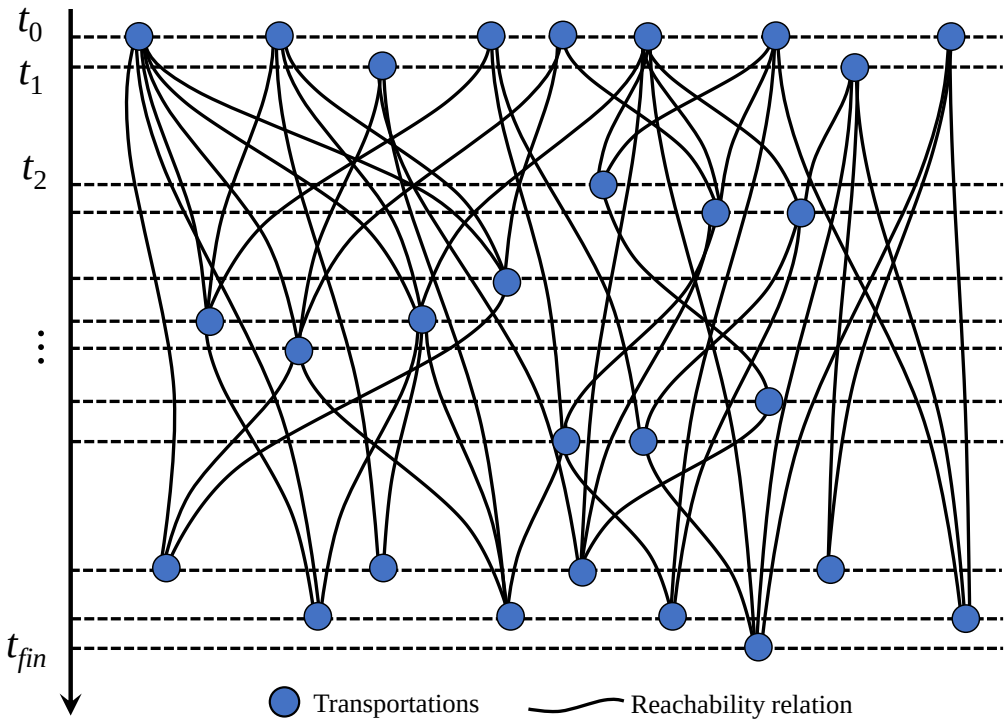

Fig. 2.2. Reachability Graph. Vertices represent transportations, arcs denote the reachability relation. The time axis is directed from top to bottom. The time moments of departure and arrival of trains are marked.

Several sequential transformations were proposed, making it possible to obtain a regular structure graph. The conversion sequence is shown in Fig. 2.3. Fig. 2.3(a) contains the same graph as in Fig. 2.2. Dotted lines represent all potentially possible transportation sequences.

2.3.1. Layer-by-layer structure. We define the layered structure as follows. The first layer contains those transportation j for which there are no transportations i satisfying inequalities (2.1) and (2.2). In other words, they cannot be transported after any other transportation by the same locomotive.

The second layer is formed only by those transportations that are reachable from the first layer's transportations, and from no others. That is, they have parents from only one layer.

Transportation of the third layer has parents in the first two layers (necessarily in both). Then the process is repeated. The result of such transformation (with some more modifications) is demonstrated in Fig. 2.3(b). Note that the time sequence of transportations given in Fig. 2.3(a) is violated.

2.3.2. Elimination of transitive closure. In [9], it was proved that the arcs of transitive closure passing through one or several layers can be eliminated in a number of cases. All valid cases are described. In Fig. 2.2 and Fig. 2.3(a), most of these arcs are already removed in order to simplify the picture. Note that not all arcs connecting vertices from non-adjacent layers can be deleted.

2.3.3. Cloning vertices to all the layers. In a problem without constraints, such a procedure is superfluous. However, in a problem with time constraints, when the found solution is invalid and needs to be corrected, the clone vertices allow finding those solutions in

which locomotives with time constraints can stop at an intermediate layer (or start from an intermediate layer). On intermediate layers, clone vertices are built so that each pair of adjacent layers can be considered separately. In Fig. 2.3(c) cloned vertices and new arcs are indicated in red.

Fig. 2.3(d) demonstrates one of the solutions: minimal path cover with nine paths. It is easy to prove that the number of paths in the cover cannot be less than the number of vertices in the graph's maximum layer (layer with a maximum number of vertices).

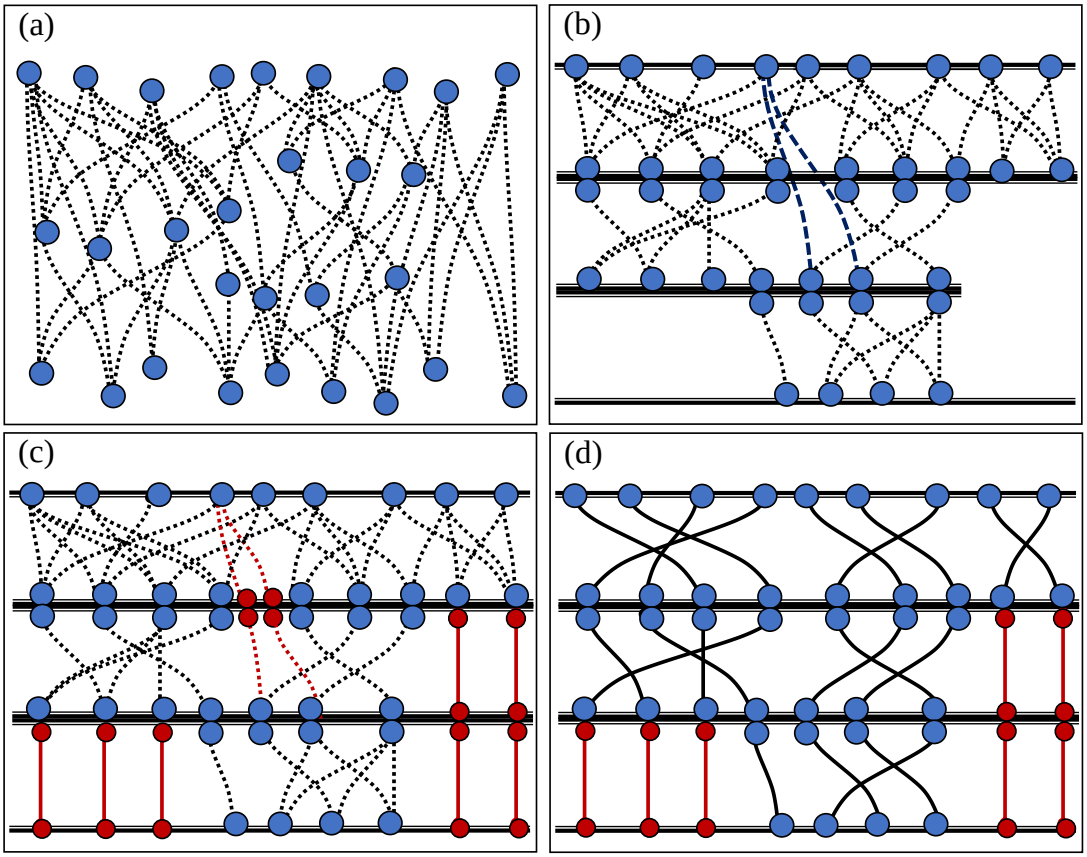

Fig. 2.3. Sequential Transformations of Reachability Graph. (a) Graph coinciding with that shown in Fig. 2.2; dotted lines are all possible interchanging of locomotives between transportations; (b) Layer-by-layer graph organization; blue dashed lines are arcs between non-adjacent layers; (c) Cloning vertices to the intermediate and bottom layers; (d) The minimal cover found

3. SOLVING THE PROBLEM WITHOUT TIME CONSTRAINTS

In [9], it is proved that solving the problem without time constraints on locomotives is equivalent to finding the minimal path cover of an acyclic graph.

3.1. Finding the minimal path cover

The minimal path cover of a directed acyclic graph can be found as the maximum matching in a corresponding bipartite graph where each "intermediate" vertex of an original graph (a vertex that has an ancestor and a child) is divided into two halves. One half has only input arcs, the other only output arcs [12–15]. This is demonstrated schematically in Fig. 2.3 (b)-(d).

The layered structure of a reachability graph shows that the vertices or half-vertices that form two adjacent layers can be considered as independent components of a bipartite graph. In this case, the maximum matching can be found between each pair of layers separately. This decomposition significantly reduces the size of the problem.

After finding the maximum matching, back-gluing the half-vertices restores the solution: the minimal path cover.

3.2. Formal definitions

Let $G = (X, A)$ be a directed graph, where X is a set of vertices, A is a set of arcs.

Definition 3.1:
The sequence of vertices and arcs $(x_{i_0}, a_{i_0}, x_{i_1}, a_{i_1}, ..., x_{i_{k-1}}, a_{i_{k-1}}, x_{i_k})$ *in graph* G *is called a path. Here* $a_{i_j} = (x_{i_{j-1}}, x_{i_j}), j = \overline{1,k}$.

Definition 3.2:
A simple path is a sequence of vertices and arcs $(x_{i_0}, a_{i_0}, x_{i_1}, a_{i_1}, ..., x_{i_{k-1}}, a_{i_{k-1}}, x_{i_k})$ *such that all vertices in it are distinct.*

Remark 3.1:
Note that in a directed acyclic graph, any path is simple.

Paths in the reachability graph will be denoted as $\{Path_1, ..., Path_k\}$.
$Path_i = (x_{i_0}, a_{i_0}, x_{i_1}, a_{i_1}, ..., x_{i_{k-1}}, a_{i_{k-1}}, x_{i_k}) = (X_i, A_i)$.

Definition 3.3:
A set of paths $P = \{Path_1, ..., Path_k\}$, *where* $Path_i = (X_i, A_i)$, *is called a path cover of a directed graph, if* $X_i \cap X_j = \emptyset$ *for* $i \neq j$ *and* $\cup_{i=1}^{k} X_i = X$; *accordingly,* $A_i \cap A_j = \emptyset$ *for* $i \neq j$ *and* $\cup_{i=1}^{k} A_i \subseteq A$. *The cardinality of a path cover* $k = |P|$ *is called the cover size.*

Since any isolated vertex is a reduced case of a path, a path cover always exists.

Definition 3.4:
A path cover $P = \{Path_1, ..., Path_k\}$ *is minimal if there is no cover of size* l, *such that* $l < k$.

3.3. Optimal Solution

The number of paths in the minimal cover determines the minimum number of locomotives for the performing of all transportations. To solve the assignment problem, one more matching must be performed – between the found paths and the available locomotives. If there is a matching that involves all paths, then the unconstrained problem has a solution.

Let the solution of the unconstrained problem exist. We can assign at least one locomotive to each path (i.e. consequence of transportations). The question arises whether the found solution can be corrected if it does not satisfy the time constraints imposed on some locomotives. Such a solution will be called *invalid*.

4. CORRECTING INVALID SOLUTION

4.1. Crossover and changeover operations. Distance between paths

To describe the procedure for correcting the found minimal cover, consider the illustration in Fig. 4.4. Fig. 4.4(a) represents the locomotive assignment to the transportation sequences denoted by paths in the cover.

Consider the fragment of the cover with three paths and three locomotives with indices 5, 6, and 7. Let locomotive 6 have a time constraint, and the last transportation in the path cannot be performed (Fig. 4.4(b)).

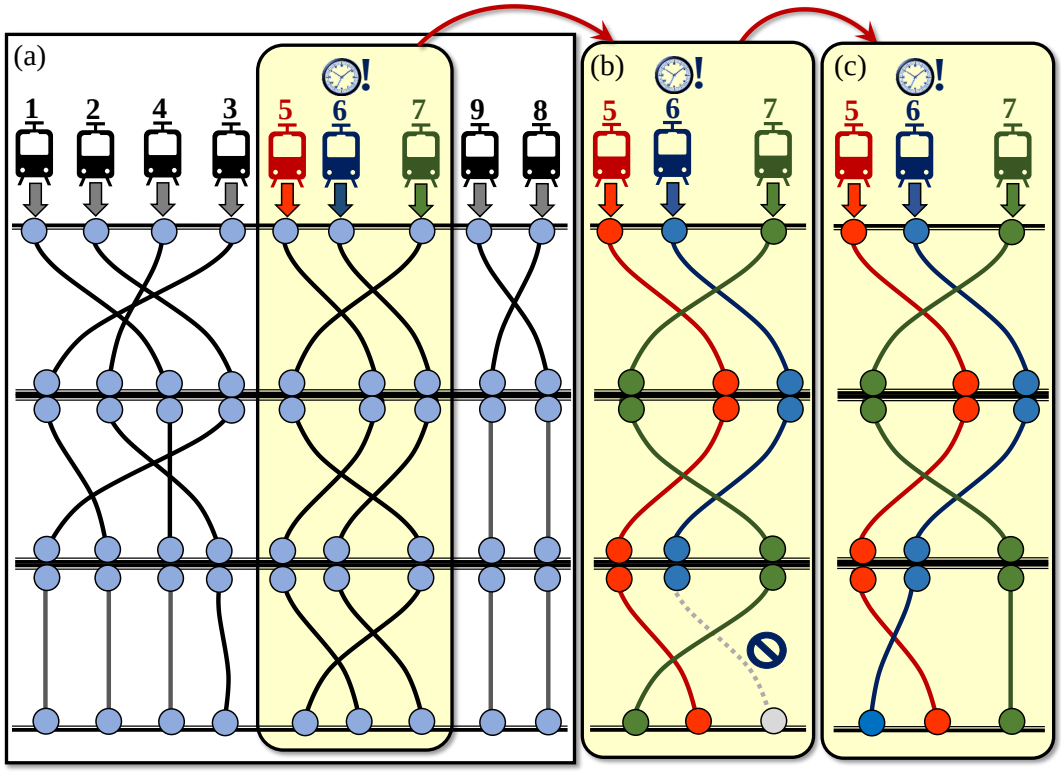

Fig. 4.4. Correction of the Invalid Minimal Path Cover. (a) Minimal path cover, coinciding with Fig. 2.3(d), and the locomotive assignment for each sequence of transportations; (b) The highlighted fragment of Fig. 4.4(a): locomotive 6 has a time constraint and cannot perform the last transportation; (c) The two paths exchange sections between the last two layers. Obtaining the valid cover

In [9], a method is described that allows changing the assignments of locomotives without changing the paths if such a new assignment exists. So, let us assume that it is impossible to obtain a valid cover by changing the locomotives' assignment. In this instance, this means that locomotive 6 has no alternative connection to the first vertices of the other paths.

The only way to correct the solution is to change the cover locally so that two paths swap the end sections. Then the invalid path becomes valid. And the second path will remain valid if its locomotive has no constraints (Fig. 4.4(c)).

We introduce the procedures that will allow to change paths while maintaining minimal cover. First, let us introduce the notions of free arcs and vertices.

Definition 4.1:
Arcs and clone vertices of a reachability graph that do not participate in the current minimal path cover are called free.

Definition 4.2:
A path consisting of free vertices and free arcs is called free.

Remark 4.1:
Obviously, the vertices corresponding to the transportations cannot be free, they always belong to the minimal cover.

4.1.1. Crossover. Following the terminology adopted in genetic algorithms [16], we will call the exchange of path sections a *crossover*.

We introduce the concepts of *simple* and *complex crossover*.

Definition 4.3:
A simple crossover, or 2-crossover, is the exchange of two paths by homologous sections using free arcs. Homologous sections contain the same number of vertices located in the same layers. Crossover can only be implemented between two adjacent layers.

A simple crossover between paths is feasible if and only if there is at least a pair of adjacent layers in which the vertices of two paths form a biclique in a bipartite graph. Then two free arcs are involved, which switches the paths' ends.

Let us consider the matching formed by the arcs between adjacent layers as a permutation on a set of paths. The crossover of the paths corresponds to a *transposition*, or a permutation in which a pair of elements swaps (Fig. 4.5 (a),(b)).

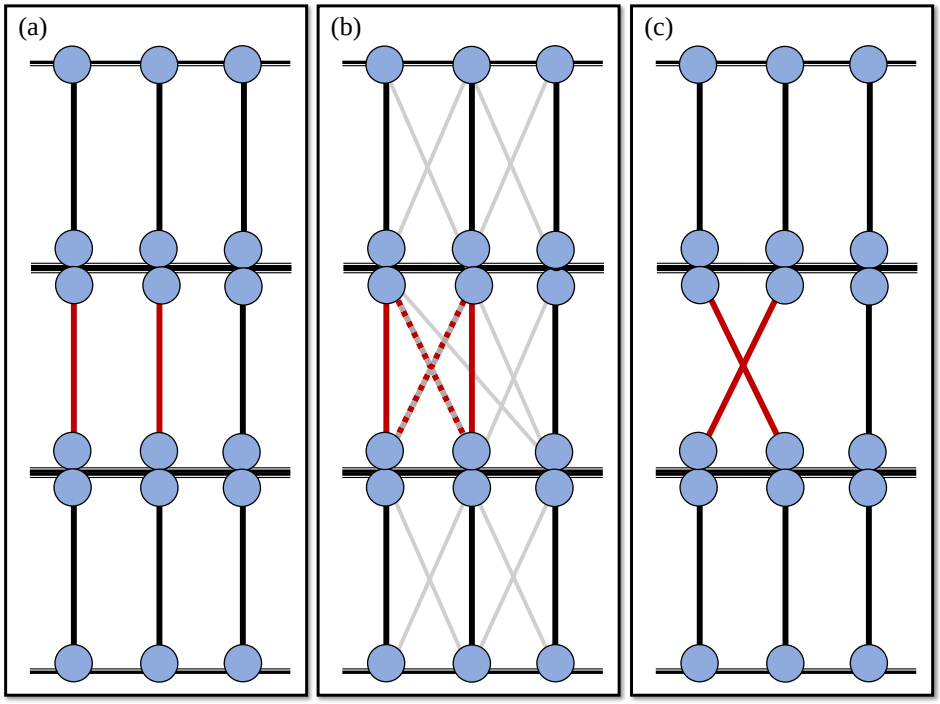

Fig. 4.5. Simple crossover: (a) the minimal cover; (b) the minimal cover and the free arcs, indicated by grey lines; the selected biclique is highlighted in red; (c) the simple crossover and the new minimal cover

The *cycle of length* k is a permutation of a subset of a set X: $\{x_1, x_2, \ldots, x_k\} \subseteq X$ mapping x_k into x_1, x_i into x_{i+1}, $(i < k)$.

The transposition is a cycle of length 2.

Definition 4.4:
We define a cyclic exchange involving $k \geq 3$ paths as a complex crossover. A complex crossover of the order k, or k-crossover, is a cyclic exchange of k paths by homologous segments.

An example of a 3-crossover is shown in Fig. 4.6.

Any permutation can be uniquely decomposed into a composition of disjoint cycles of length $k \geq 2$, and the expression of the permutation is unique up to the order of the cycles. Thus, the two introduced operations make it possible to construct any arbitrarily given permutation.

Theorem 4.1:
The k-crossover $(k \geq 2)$ operation converts a valid path with a locomotive without constraints into a valid path.

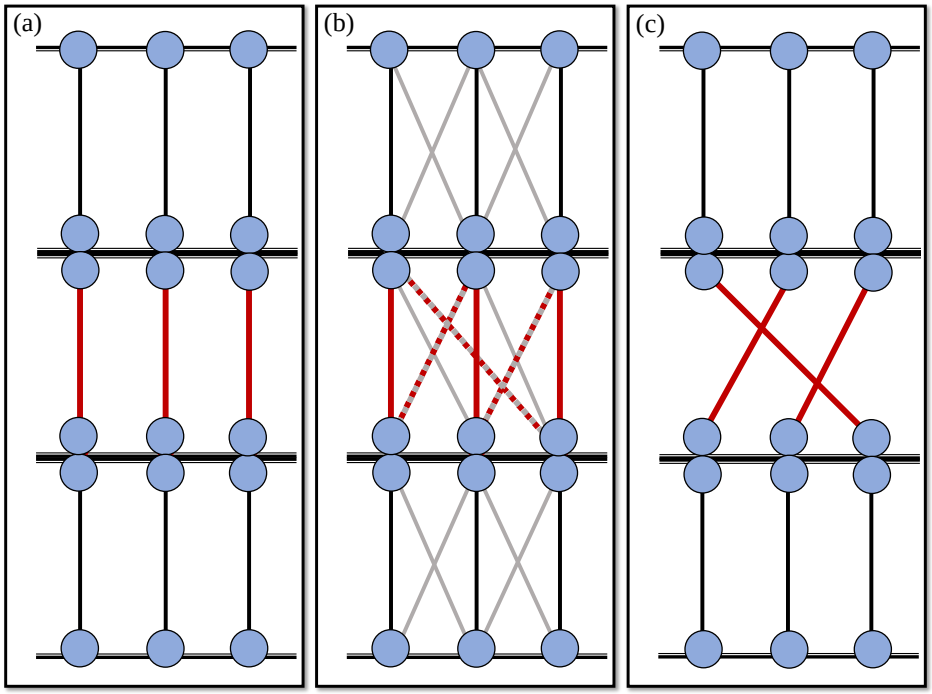

Fig. 4.6. Complex crossover: (a) the minimal cover; (b) the minimal cover and the free arcs, indicated by grey lines; the selected arcs are highlighted in red; (c) the complex crossover and the new minimal cover

Proof
If the crossover is possible, this means that the initial graph contains all the arcs necessary to form new paths. The arc in the graph denotes the reachability of the next transportation from the end of the previous one. Thus, the new path section is reachable for the locomotive as well as the previous path section. And since the locomotive has no time constraints, the new path will be entirely valid. □

4.1.2. Changeover. In a crossover, two or more paths exchange their parts, while the number of paths in the cover always remains unchanged. There are other situations where the end of an invalid path needs to be switched from one vertex to another. Usually, the new valid end is an isolated vertex at the bottom level (clone or original) or a path consisting of clone vertices (like the red paths in Fig. 2.3(c),(d)).

Definition 4.5:
Changeover of a path $Path_i$ is switching the end of a path to a free vertex or a free path using a free arc (Definitions 4.1, 4.2). In some cases a switch can be made to the original isolated vertex if it is a path.

Unlike crossover, which is a binary or, in general, n-ary operation, the changeover operation is unary.

Fig. 4.7 demonstrates an example of a changeover. In the reachability graph in Fig. 4.7(a), the colors of vertices indicate the availability of locomotives. The blue locomotive has a time constraint and cannot be assigned to the transportation of the bottom level. In this case, the vertices of the previous level with blue parts are cloned (Fig. 4.7(b)). In Fig. 4.7(c), the invalid cover is presented; Fig. 4.7(d) demonstrates the changeover operation.

It often happens that a changeover makes free a vertex that can become a part of another path, i.e., the corresponding transportation can be carried out by another locomotive. In

this case, one more switch occurs, which we will call a *co-changeover*. An example of co-changeover operations is shown in Fig. 4.7(e).

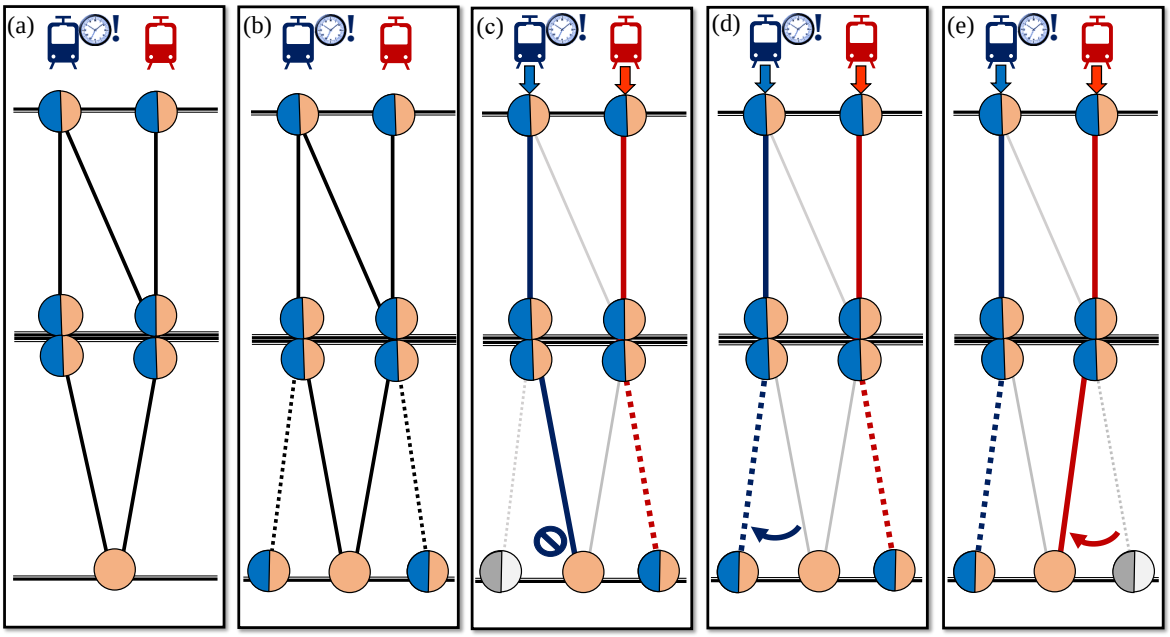

Fig. 4.7. Changeover and co-changeover: (a) the initial reachability graph; colors of vertices correspond to available locomotives; blue locomotive has a time constraint; (b) cloning vertices to the bottom level; (c) the invalid cover; (d) changeover from an invalid vertex to a valid clone vertex; (e) co-changeover from the clone vertex to the real transportation

The Figure 4.7 shows a changeover and a co-changeover that do not change the number of paths in the cover. However, there may be situations where the changeover operation generates a new path that requires one more locomotive. Such a situation is demonstrated in Fig. 4.8. Note that if there were no time constraints, two locomotives would be sufficient to perform all the transportations.

Remark 4.2:
The examples in Figures 4.7 and 4.8 consider the case of early termination of locomotives. The second possible case is a late start. Without loss of generality, we will consider time constraints corresponding to early termination. The problem with a late start is dual and is solved in a similar way.

Remark 4.3:
As in the case of k-crossover, k-co-changeover is possible, when there is a cyclic replacement of the ends of the paths with the release of one clone or original vertex or a path.

4.1.3. Distance between paths. Let us number the paths from left to right according to their beginnings in the graph.

The distance $d(Path_i, Path_j)$ from the path $Path_i$ to the path $Path_j$ is a vector of length 2: $d(Path_i, Path_j) = (d_1, d_2)$. The first component d_1 is equal to the minimum number of crossover operations that must be performed in order for the path $Path_i$ to get the end of the path $Path_j$. The second component takes the value 0 or 1, depending on whether a changeover operation is needed to reach the path $Path_j$. If $Path_j$ is unreachable from $Path_i$, we will assume that $d(Path_i, Path_j) = (\infty, \infty)$.

There cannot be more than one changeover operations when determining the distance, since a changeover can only be used at the final stage.

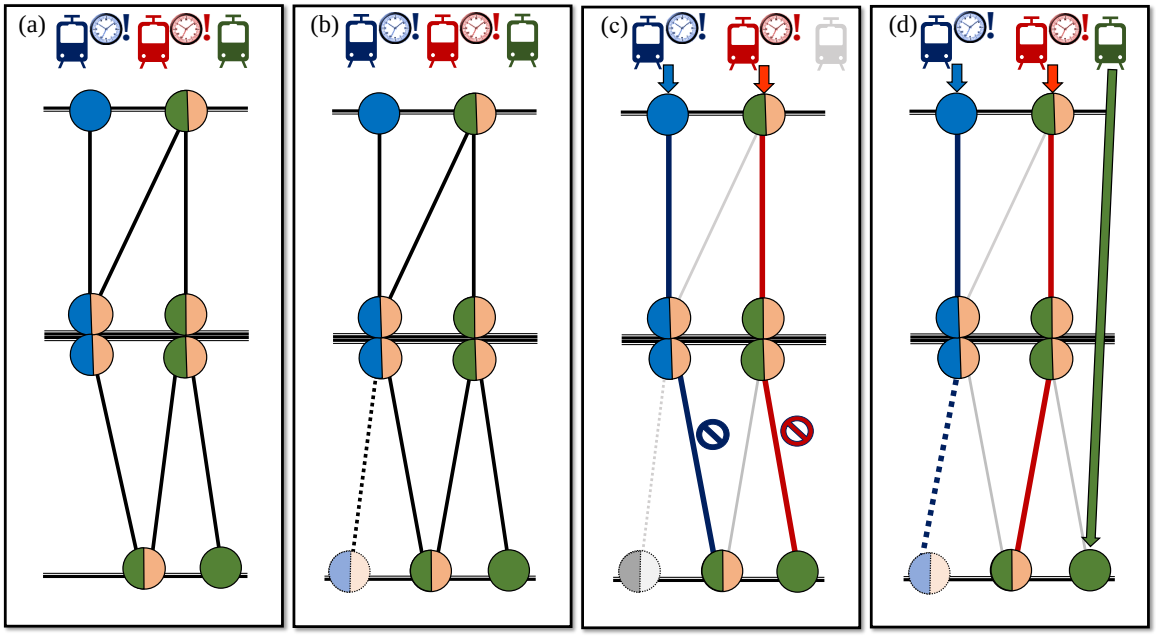

Fig. 4.8. Changeover and co-changeover: (a) the initial reachability graph; colors of vertices correspond to available locomotives; the red and blue locomotives have time constraints; (b) cloning vertex to the bottom level; (c) the invalid cover; (d) changeover (blue) from an invalid vertex to a valid clone vertex, co-changeover (red), and assigning a new locomotive (green)

Generally, the distance defined in this way is non-symmetric: the equality $d(Path_i, Path_j) = d(Path_j, Path_i)$ may not hold. It depends on the topology of the graph. Fig. 4.9 (b) demonstrates the symmetric distance between paths $Path_1$ and $Path_2$; Fig. 4.9 (c) and (d) show that $d(Path_1, Path_3) \neq d(Path_3, Path_1)$, since $d(Path_1, Path_3) = (1, 0)$ and $d(Path_3, Path_1) = (2, 0)$.

Remark 4.4:
It should be taken into account when measuring the distance that each crossover is performed between a pair of adjacent layers. All permutations within a pair of layers are considered a single complex crossover (Fig. 4.9 (c)).

4.2. Changing locomotive assignment

By constructing a layer-by-layer structure, we transformed the graph so that every two consecutive layers represent an independent bipartite graph. When searching for the minimal cover, the maximum matching is searched locally between each pair of layers.

We will assume that the problem without time constraints has a solution and any consequence of transportations can be performed by at least one locomotive.

Before applying the algorithm presented below, let us try to correct not the cover, but the assignments of the locomotives. This procedure is described in detail in [9]. The idea is shown in Fig. 4.10.

We construct a three-layer graph as follows. The mid-level vertices are the condensation of the paths that make up the minimum cover of the reachability graph.

Top and bottom layers contain all available locomotives. The arcs between top and middle layers are all possible assignments of locomotives to the initial vertices of the paths. The arcs between middle and bottom layers indicate the correspondence of locomotives and the ends of each path. So, if the same locomotive is connected to the middle vertex from the upper and

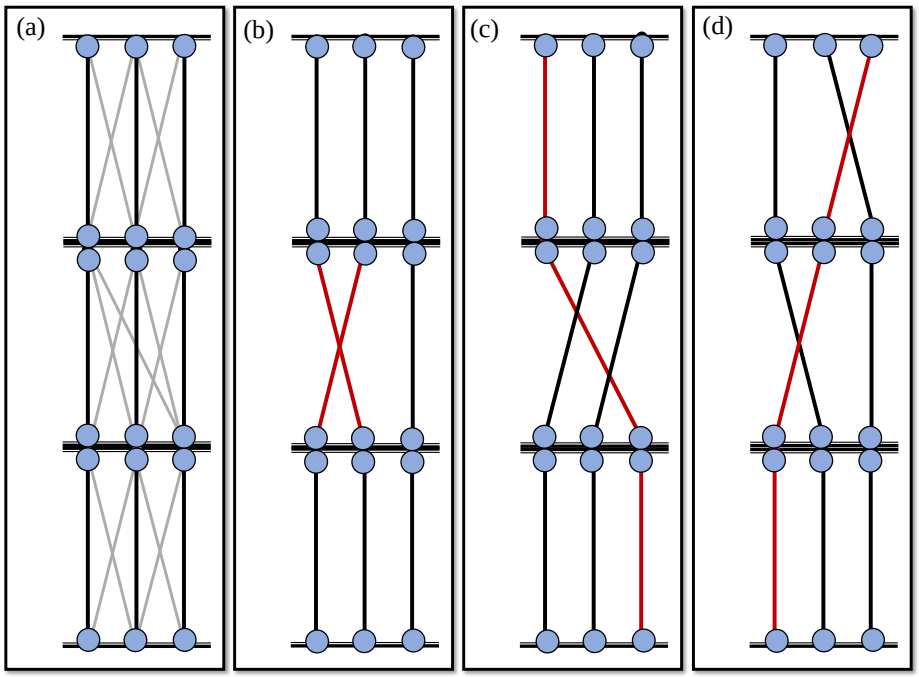

Fig. 4.9. Distance between paths: (a) the initial reachability graph coinciding with the graphs in Fig. 4.5, Fig. 4.6; (b) $d(Path_1, Path_2) = d(Path_2, Path_1) = (1, 0)$; (c) $d(Path_1, Path_3) = (1, 0)$; (d) $d(Path_3, Path_1) = (2, 0)$

lower layers, then it can carry out the entire sequence of transportations. In terms of a three-layer graph, this means that we should remove arcs that are not symmetrical with respect to the middle layer, since they connect the beginning and end of the track with different locomotives.

Fig. 4.10(a) demonstrates the principle of construction of a three-layer graph. Fig. 4.10(b) contains a reduced graph from Fig. 4.10(a), on which only arcs are left that are symmetrical with respect to the middle layer. As it is easy to see, blue locomotive is not connected with any path. This cover cannot be corrected by reassigning locomotives.

Fig. 4.10(c) considers another case. Here the invalid assignment of locomotives can be corrected. The reduced symmetric graph immediately offers a solution – Fig. 4.10(d).

The situation in Fig. 4.10(a),(b) will be considered again after the description of the algorithm of correcting the solution Fig. 4.11.

4.3. Algorithm for improving invalid paths in the minimal cover

1. Find the minimal path cover of a layered reachability graph.
2. Construct a three-layer graph locomotives-paths-locomotives (Fig. 4.10).
3. Find an invalid path $Path_i$ in this graph (the path associated with disjoint sets of locomotives of the upper and lower layers).
4. Find the locomotive vertex L_s in the upper layer that is not assigned to any valid path and has an arc to the path $Path_i$. Since it is assumed that the problem has a solution for locomotives without constraints, such a locomotive L_s necessarily exists.
5. Find a valid path $Path_j$ that has a connection with the locomotive L_s at the lower level. If there are more than one of such paths, rank the set of candidates in order of increasing the second coordinate of the distance vector – first all paths with $d_2(Path_i, Path_j) = 0$, then with $d_2(Path_i, Path_j) = 1$. Ranking on the second coordinate is done to avoid the

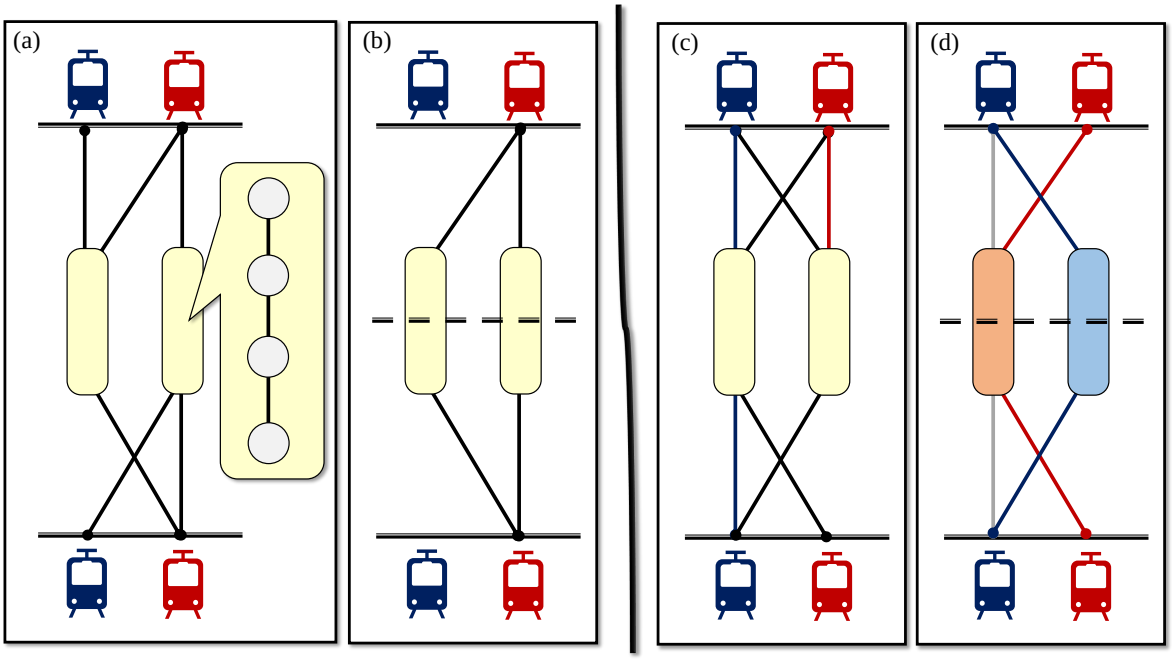

Fig. 4.10. Construction of a three-layer graph and two situations of possible locomotive assignments: (a) the three-layer graph; the middle layer is the condensation of the paths; (b) arcs symmetrical with respect to the middle layer; solution does not exist; (c) another possible situation of matching paths and locomotives; invalid assignment of the red locomotive; (d) the symmetric arcs and the found solution

changeover operation, since it may increase the number of paths in the minimal cover (Fig. 4.8).
Take as $Path_j$ the first path from the list.

6. If $d_1(Path_i, Path_j) = 1$ and $d_2(Path_i, Path_j) = 0$, then a single k-crossover is needed.
   If crossover is simple $(k = 2)$.
   - If $Path_j$ assigned to a locomotive with constraints and the crossover makes this path invalid then delete $Path_j$ from the list; if there are other crossover candidates, go to step 5, else to step 3 (in this case, there is no valid cover, however, we will improve the cover as long as possible).
   - Otherwise, perform a 2-crossover between paths $Path_i, Path_j$ on corresponding layers.
   - If there are other invalid paths, go to step 3; otherwise a valid cover is found, exit.

   If $k > 2$.
   - If all the paths included in the k crossover, except $Path_i$, have locomotives without constraints, make a k-crossover; if there are other invalid paths, go to step 3; otherwise, a valid cover is found, exit.
   - If some of the paths included in the k-crossover, other than $Path_i$, assigned to locomotives with constraints, and the crossover makes at least one of the new paths $Path_j$ invalid, remove $Path_j$ from the list of candidates for exchange; if there are other candidates for exchange, go to step 5.
   - If we have iterated over all candidate paths, go to step 3.
7. If $1 < d_1(Path_i, Path_j) < \infty$ and $d_2(Path_i, Path_j) = 0$
   Find a path connecting the beginning of $Path_i$ with the end part of $Path_j$. Carry out d_1 successive crossovers so that each of them contains one arc from the found path by performing step 6 d_1 times.

If there are other invalid paths, go to step 3; otherwise, a valid cover is found, exit.

8. If $d_2(Path_i, Path_j) = 1$.
   - Find a path connecting the beginning of the path $Path_i$ with the final part of path $Path_j$. Perform d_1 crossovers in accordance with steps 6 and 7.
   - Make the changeover.
   - Check if co-changeover exists.
   - If co-changeover exists, make it. If there are other invalid paths, go to step 3; otherwise a valid cover is found, exit.
   - If no co-changeover exists, check if there is an available locomotive for the path resulting from the changeover.
   - If the locomotive exists, assign it. If there are other invalid paths, go to step 3; otherwise a valid cover is found, exit.
9. There is no complete valid cover. The found cover has been improved as much as possible. Exit.

If an invalid path $Path_i$ can be corrected by swapping homologous sections of one or more paths, it will be corrected. The same applies to the situation when the ends of the paths are switched to other sections (changeover and co-changeover). If such a procedure cannot be done, the algorithm switches to another invalid path. After correcting all the other invalid paths, the algorithm will return to $Path_i$ again and will try to make a crossover or changeover on a new set of paths. This procedure is repeated until all possibilities to correct the paths are exhausted.

Fig. 4.11 illustrates the operation of the algorithm in the case of $d_1 = 1, d_2 = 0, k = 2$. We find a biclique in a bipartite graph between a pair of adjacent layers and perform a crossover. In Fig. 4.11(c) the crossover has been performed, in Fig. 4.11(d) the resulting valid cover is shown.

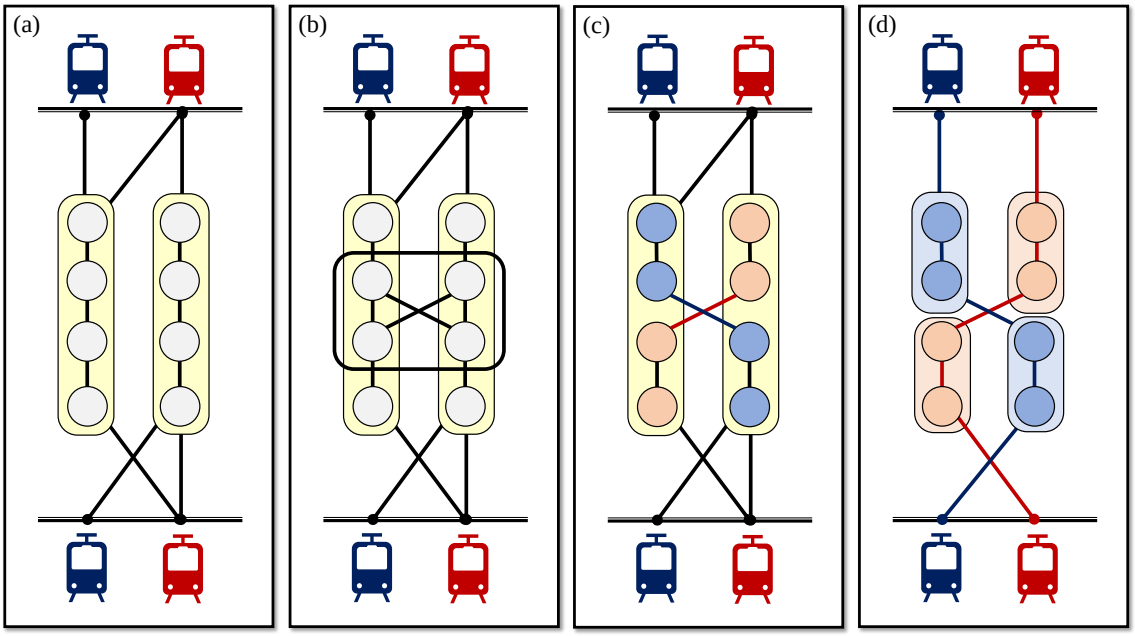

Fig. 4.11. Construction of a valid cover: (a) a three-layer graph for which there is no valid cover; (b) the selected biclique; (c) crossover; (d) valid cover; paths in it have exchanged homologous sections

Remark 4.5:
Since the algorithm certainly does not corrupt any path, it converges. Either all the paths will

be corrected, or there will be a set of invalid paths for which, under the given constraints on locomotives, there is no solution.

Remark 4.6:
There should not be too many locomotives with time constraints to have a high probability of obtaining a valid cover. If there are many of them, the planning horizon should be adjusted.

5. CONCLUSIONS

A model of transportation on a linear section of the railway is represented as an acyclic graph. Vertices in this graph correspond to transportations, and arcs indicate the possibility to perform one transportation after another by the same locomotive. Such a graph was called a reachability graph. The problem of constructing optimal locomotive assignment without time constraints is equivalent to the problem of finding the minimal path cover of an acyclic graph. And this problem, in turn, is reduced to finding a perfect matching in a bipartite graph.

In the absence of time constraints on locomotives, any minimal path cover of a given acyclic graph is an optimal solution to the original problem.

Due to time constraints, some transportation sequences cannot be completed. Under such conditions, the problem becomes much more complicated. Dynamic algorithms are often unable to find an existing solution [10]. Heuristic algorithm [9] improves the invalid solution (finds more assignments), but also does not always find an optimal solution when it exists. To simplify the finding an optimal solution under time constraints, a topological sorting of the graph and a number of other transformations are performed. As a result, the structure of a reachability graph becomes regular and the dimension of the problem is reduced.

To correct an invalid solution under time constraints, two operations are introduced: crossover and changeover. A crossover can be *simple* when a pair of paths exchange homologous sections, and *complex* when a permutation occurs between several paths. In crossover, two or more paths are exchanged with homologous sections containing the ends of these paths. A changeover switches the end of an invalid path to another free vertex or free path. The co-changeover operation allows in some cases to avoid the increase in the number of paths in the minimal cover, which occurs due to changeover.

The algorithms is proposed that convert an invalid cover into a valid one. This algorithm always finds a solution if it exists. However, during its implementation, it is necessary to check a large number of conditions.

ACKNOWLEDGEMENTS

This work was partly supported by the Russian Foundation for Basic Research (project no. 20-07-00190a).

REFERENCES

1. Gainanov D. N., Kibzun A. I., Rasskazova V. A. (2018). The Decomposition Problem for the Set of Paths in a Directed Graph and Its Application. *Autom. Remote Control*, 79, 2217–2236.
2. Jaumard B., Tian H. (2016). Multi-Column Generation Model for the Locomotive Assignment Problem. *Proc. of 16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'16)*, 6, 1 – 13.
3. Zinder, Y., Lazarev, A.A., and Musatova, E.G. (2020). Rescheduling Traffic on a Partially Blocked Segment of Railway with a Siding. *Autom. Remote Control*, **81**(6), 955–966.

4. Lazarev A. A., Musatova E. G., Tarasov I. A. (2016). Two-directional traffic scheduling problem solution for a single-track railway with siding. *Autom. Remote Control*, **77**(12), 2118–2131.
5. Ahuja R.K., Liu J., Orlin J.B., Sharma D. and Shughart L.A. (2005). Solving real-life locomotive scheduling problems. *Transport. Sci.*, **39**(4), 503 – 517.
6. Noori, S., Ghannadpour, S.F. (2012). Locomotive assignment problem with train precedence using genetic algorithm. *J Ind Eng Int*, **8**(9), 1 – 13.
7. Archetti C., Peirano L., Speranza M. G. (2022). Optimization in multimodal freight transportation problems: A Survey. *European Journal of Operational Research*, **299**(1), 1 – 20.
8. Hanczar P., Zandi A. (2021). A novel model and solution algorithm to improve crew scheduling in railway transportation: A real world case study. *Computers & Industrial Engineering*, 154, 107 – 132.
9. Zhilyakova L. Yu., Kuznetsov N. A. (2021). Graph Methods for Solving the Unconstrained and Constrained Optimal Assignment Problem for Locomotives on a Single-Line Railway Section. *Autom. Remote Control*, **82**(5), 780 – 797.
10. Zhilyakova L. Yu., Kuznetsov N. A., Matyukhin V. G., Shabunin A. B., and Takmazian A. K. (2019). Locomotive Assignment Graph Model for Freight Traffic on Linear Section of Railway. The Problem of Finding a Maximal Independent Schedule Coverage. *Autom. Remote Control*, **80**(5), 946 – 963.
11. Matyukhin V. G., Shabunin A. B., Kuznetsov N. A., and Takmazian A. K. (2017). Rail transport control by combinatorial optimization approach. *11th IEEE International Conference on Application of Information and Communication Technologies*, 1, 419 – 422.
12. Boesch F.T., Gimpel J.F. (1977). Covering the points of digraph with point-disjoint paths and its application to code optimization. *J. Associat. Comput. Machin.*, **24**(2), 192 – 198.
13. Noorvash Sh. (1975). Covering the vertices of a graph by vertex-disjoint paths. *Pacific J. Math.*, **58**(1), 159 – 168.
14. Jackson B. and Ordaz O. (1990). Chvátal–Erdös conditions for paths and cycles in graphs and digraphs. A survey. *Discret. Math.*, **84**(3), 241 – 254.
15. Chvátal V. and Erdös P. (1972). A note on Hamilonian circuits. *Discret. Math.*, 2, 111 – 113.
16. Sivanandam S., Deepa S. (2008). Genetic Algorithms. *Introduction to Genetic Algorithms*. Springer, Berlin, Heidelberg. P. 15 – 37.
17. Mirjalili S. (2019). Genetic Algorithm. *Evolutionary Algorithms and Neural Networks. Studies in Computational Intelligence*, vol 780. Springer, Cham. 43 – 55.

