A heuristic approach to improve the robustness of a railway timetable in a bottleneck area

Inneke Van Hoeck*[1] and Pieter Vansteenwegen[1]

[1]KU Leuven Institute for Mobility - CIB, KU Leuven,
Celestijnenlaan 300, Box 2422, 3001 Leuven, Belgium

Short summary

In this paper, a heuristic is presented to improve the robustness of a given railway timetable in a bottleneck area. The timetable can be adapted by adjusting the timing and routing of trains, but cannot deviate too much from the current timetable. Robustness is measured with an objective function that considers the buffer times between train pairs. The developed algorithm updates the routing and the timing in separate steps. It is applied to a bottleneck area of the Belgian network. The results show that the objective can be improved by about 10% when alternative routing options are considered. Additional experiments with smaller instances indicate that the heuristic is capable of finding near-optimal solutions.
Keywords: Operations research, Public transport, Railway transport, Timetable robustness

1 Introduction

Railway networks are often heavily used. In nearly saturated parts of the network, limited capacity is one of the main reasons for delay propagation (Burggraeve & Vansteenwegen, 2017). Often, delay propagation is closely related to timetable robustness (Dewilde et al., 2014). Robustness is a concept that has many definitions and interpretations, an extensive survey on this topic can be found in Lusby et al. (2018). The aim of this work is to adjust a given timetable to improve its robustness. Our objective function is based on the work of Dewilde et al. (2014) and looks for a good spreading in time of the trains by considering the buffer times between trains. Both the timing and the routing of the trains can be adjusted. However, the new timetable cannot deviate too much from the current timetable. This is required by the Belgian railway companies to ensure the practical applicability. Additional constraints for the timing of the trains are included to guarantee this. Several methods to improve the robustness of a timetable by making relatively small changes have been presented in literature. Some examples can be found in Andersson et al. (2015), Jovanović et al. (2017) and Högdahl et al. (2019). They all present methods to optimize the allocation of time supplements in a given timetable. These methods do not allow alternative routing options, as will be considered in this work.

Thus, in this paper it is examined how the robustness of a given timetable can be improved by changing both the timing and routing of trains, while not deviating too much from the original timetable. The timetable is considered on a microscopic level. A heuristic to solve this problem is presented and applied to a case study for a part of the Belgian railway network.

2 Problem description

Denote the set of all considered trains as T. A path p is associated with each train to indicate the route that it takes through the network. We refer to a *trainroute* (t, p) when talking about a train t that uses path p. A path is a sequence of infrastructure resources that are used by the train. The set of resources that is used by a trainroute (t, p) is denoted by $R_{(t,p)}$. Each resource r in a trainroute (t, p) will be blocked by the train during a certain period of time, meaning that no other train is allowed to use the resource. The blocking times are determined according to the blocking time theory. Thus, for each $r \in R_{(t,p)}$ the *reserve* and *release* time indicate the time when the blocking time of the resource starts and stops, respectively denoted by $res^r_{(t,p)}$ and $rel^r_{(t,p)}$. These

times are determined relative to the start time of the train t, denoted by S_t. This is the time when a train enters the considered zone. In this work, alternative routing options for trains are considered. Thus, for each train t there are multiple possible paths $p \in P_t$ that can be selected. The point where a train enters and leaves the considered zone is assumed to be fixed for all paths. The reserve and release times for the resources in an alternative path are estimated based on the timing for the path that is used in the current timetable.

The absolute reserve and release times for a resource r in a trainroute (t, p) can be written as:

$$RES^r_{(t,p)} = S_t + res^r_{(t,p)}, \tag{1}$$

$$REL^r_{(t,p)} = S_t + rel^r_{(t,p)}. \tag{2}$$

The *buffer time*, between a pair of trainroutes (t, p) and (t', p') is defined as

$$B_{(t,p),(t',p')} = \min_{r \in R_{(t,p)} \cap R_{(t',p')}} b^r_{(t,p),(t',p')} \tag{3}$$

where $b^r_{(t,p),(t',p')}$ is the minimum time span between the two trainroutes on resource r. It is defined as:

$$b^r_{(t,p),(t',p')} = \begin{cases} \min(RES' - REL, RES + H - REL') & \text{if } t \prec t', \\ \min(RES - REL', RES' + H - REL) & \text{if } t' \prec t. \end{cases} \tag{4}$$

Here, the notation of the absolute reserve and release times is simplified by omitting the explicit reference to the trainroute and resource. Additionally, $t \prec t'$ means that trainroute (t, p) uses resource r before trainroute (t', p') does. The parameter H is the time period that is considered, 1 hour in this case. Note that this definition takes into account that the timetable is cyclic.

Similar to the work of Dewilde et al. (2014), a cost $c_{(t,p),(t',p')}$ is associated with each buffer time as follows:

$$c_{(t,p),(t',p')} = \begin{cases} 100 & \text{if } B \leq 0, \\ \frac{-1}{10}B + 10 & \text{if } 0 < B \leq 60, \\ \frac{-1}{30}B + 6 & \text{if } 60 < B \leq 120, \\ \frac{-1}{390}B + \frac{90}{39} & \text{if } 120 < B \leq 900, \\ 0 & \text{else,} \end{cases} \tag{5}$$

where $B_{(t,p),(t',p')}$ is simply denoted by B. This is a piecewise linear, monotone decreasing function. A small buffer time will thus correspond to a large cost. The objective is to minimize the sum of all costs:

$$\text{minimize} \sum c_{(t,p),(t',p')}. \tag{6}$$

With this objective in mind, let us look more closely at the definition of the cost. A cost of 100 is given when the buffer time is smaller than or equal to 0, i.e. when there is a conflict between the two trains. When the buffer time is larger than 900 seconds, the cost is set to 0 based on the idea that these trains are far enough apart such that they do not influence each other (Dewilde et al., 2014). The slope of the functions that are used for the different intervals for B becomes less steep when B becomes larger. This is done to include the following idea: improving a buffer time from 30 seconds to 1 minute is much more valuable than increasing a buffer time of 10 minutes with 30 seconds. Note that the same principle can be obtained by defining the cost as $1/B$. However, we opt for a piecewise linear function because this allows a MILP formulation of the problem.

In conclusion, the aim of this work is to obtain a new feasible timetable such that the value of the objective function (6) is minimal. A timetable is completely defined when the start time S_t and the selected path $p \in P_t$ is known for each train t. As stated in section 1, the new timetable cannot deviate too much from the current timetable to ensure the practical applicability. Additional timing constraints are imposed to achieve this, but they are not discussed further in this short paper.

3 Heuristic approach

Solving the problem described in section 2 with an exact model is not possible within a reasonable amount of time for the instances considered in the case study (see section 4). Therefore, a heuristic

approach is developed. Updating the timing and the routing is done in separate steps. In the remainder of this work, a *route selection* refers to the set of trainroutes (t, p) that are used in the timetable. In a single iteration of the algorithm, a new solution is determined by first calling the `routing update` to find a new route selection, while the timing is fixed, followed by the `timing update` function to change the start times. These functions are now discussed in more detail.

Routing update

The `routing update` is a function that takes the current start times as input. The aim of this function is to find a feasible route selection such that the timetable defined by this route selection and the current start times has the lowest possible cost. Note that the complete impact of the route selection on the cost of the timetable is only known after the `timing update` is applied. However, observations show that timetables with a low cost after only applying the `routing update` lead to timetables with a low cost after the `timing update` is also applied.

The crucial concept in the `routing update` is the *cost graph*. Each trainroute (t, p) is a node in this graph. An edge is added between two nodes if the corresponding trainroutes can be simultaneously included in the route selection. Two conditions must hold for this. First, there cannot be a conflict between the trainroutes. Second, the additional timing constraints must hold, if there are such constraints defined between the trainroutes. The cost $c_{(t,p),(t',p')}$ is associated with the edge between two trainroutes. Denote the set of edges in the cost graph as E. The trainroute combinations that are not connected by an edge are included in the complement set E^C.

A feasible route selection corresponds to a clique of the cost graph where the number of nodes in the clique is equal to the number of trains, since a route must be selected for each train. As stated before, we aim to find the route selection that leads to the timetable with minimal cost. This is done by solving the following ILP model. The decision variables are:

$x_{(t,p)} = 1$ if the node (t, p) is included in the clique, 0 otherwise,
$y_{(t,p),(t',p')} = 1$ if the edge $((t, p), (t', p'))$ is included in the subgraph induced by the clique, 0 otherwise.

The objective function is given by:

$$\text{minimize} \sum_{((t,p),(t',p'))\in E} c_{(t,p),(t',p')} \cdot y_{(t,p),(t',p')}. \tag{7}$$

For each train, exactly one route must be included in the clique, thus:

$$\sum_{p\in P_t} x_{(t,p)} = 1 \qquad \forall t \in T. \tag{8}$$

If two nodes are not connected in the cost graph, then at most one of them can be included in the clique:

$$x_{(t,p)} + x_{(t',p')} \leq 1 \qquad \forall((t, p), (t', p')) \in E^C. \tag{9}$$

An edge is included in the subgraph induced by the clique if both its incident nodes are included in the clique. The following constraints hold $\forall((t, p), (t', p')) \in E$:

$$y_{(t,p),(t',p')} \leq x_{(t,p)}, \tag{10}$$
$$y_{(t,p),(t',p')} \leq x_{(t',p')}, \tag{11}$$
$$y_{(t,p),(t',p')} \geq x_{(t,p)} + x_{(t',p')} - 1. \tag{12}$$

The trainroutes that are included in the clique give the new route selection.

Timing update

In the `timing update` function, the current route selection and start times are given as input. The aim is to adjust the timing of the trains to decrease the cost of the timetable. The function keeps iterating until no decrease larger than a certain threshold can be found or when a maximum computation time, for example 1 second, is reached. In a single iteration, first the `possible improvements` function is applied to create a list of possible timings that can be changed. The `select improvements` function is then used to determine which of these possible changes will

actually be applied. These functions are now discussed in more detail.

Because the route selection is fixed in the entire `timing update` function, we simplify the notation by omitting the reference to the path p and refer to train t instead of trainroute (t, p). For each train t the total cost related to this train in the current timetable is defined as follows:

$$C_t = \sum_{t' \in T \setminus t} c_{t,t'}. \tag{13}$$

The current start times are used to calculate this cost. Now, the `possible improvements` function runs over all trains. For each train t, the aim is to find the optimal start time $\hat{s}_t$, while the start times of the other trains remain fixed. The definition of the cost related to a train can be extended to be dependent on the start time of that train. Thus C_t^s is calculated as in expression 13, but the start time of train t is now equal to s, instead of its current start time. We look for the value s that gives the minimal related cost C_t^s. Discrete times, with a time step of 6 seconds (0.1 minutes), are considered for s. For a possible start time s it is first verified if this start time leads to a feasible timetable, i.e. there are no conflicts, and if the additional timing constraints are satisfied. If this is the case, the cost C_t^s is determined. The time that leads to the minimum cost is denoted by $\hat{s}_t$. The improvement that can be made to the objective by changing the start time of train t is defined as $C_t - C_t^{\hat{s}}$. If this value is greater than a certain threshold, for example 0.1, then t is added to the list of possible improvements.

Not all of these possible changes can be applied at the same time. When two trains have an impact on each other, meaning that they have common resources or have a timing constraint defined between them, the calculated improvements are no longer correct if their timing is updated at the same time. Therefore, the `select improvements` function determines a subset of the possible improvements such that none of the selected trains have an impact on each other. The total improvement in the objective is then given by the sum of the improvements of the separate trains. Selecting such a subset is done by creating an *improvement graph*. Each train that is included in the list of possible improvements is a node in this graph. An edge between two nodes indicates that they have an impact on each other. Thus, finding trains that can be updated at the same time corresponds to finding an independent set in the improvement graph. This is done in a greedy way as follows. Select the node in the improvement graph with the largest improvement. Remove this node and its adjacent nodes from the graph. Repeat this process until there are no nodes left in the graph. The timing of the selected trains can then be updated.

Overview of the algorithm

Algorithm 1 now presents an overview of the heuristic algorithm. The `update routing` and `update timing` functions that were discussed in the previous sections are the core of the algorithm.

Algorithm 1: Overview of the heuristic algorithm

```
Data: original_solution
Result: optimal_solution
solution ← find initial solution(original_solution);
while comp_time ≤ max time do
    new_route_selection ← routing update(start_times);
    new_start_times ← timing update(start_times, new_route_selection) ;
    if new_objective < objective then
        solution ← new_solution
        if new_objective < optimal_objective then
            optimal_solution ← new_solution
        end
    else
        solution ← generate random solution(new_solution)
    end
end
```

The algorithm starts from the original timetable. The function `find initial solution` adapts the timing of the trains to obtain a better spreading. The routes are not adjusted yet. In this function, a MILP model is used to minimize the overall maximum cost that is found in the timetable,

while satisfying the feasibility constraints. Then the `timing update` is applied to create a better spreading in time. As stated before, an iteration starts by calling the `routing update` to generate a new route selection. This route selection and the current start times are then used as input for the `timing update`. Together, the new start times and the new route selection completely define a new timetable. The objective value related to this timetable is referred to as new objective. If this new objective is better than the current objective, the current solution is updated. It is also checked if the new objective is better than the optimal solution found until now, if so, the optimal solution is updated. If the new objective is not better than the current objective, then a new random, feasible solution is generated to continue with in the next iteration. The algorithm continues until a maximum computation time is reached.

4 Results and discussion

A part of the Belgian network that is centered around the station of Halle is considered. This is located just outside of Brussels, which is the main bottleneck of the Belgian railway network. Figure 1 shows a macroscopic view of this area. A small part of it is shown in microscopic detail on figure 2 to illustrate the complexity of the network. In this case study, zone 1 and 2 indicated in figure 1 are considered. All the trains that use this part of the network during one hour of the morning peak (between 7:00 and 8:00) are selected. This leads to 35 trains for zone 1 and 40 trains for zone 2. More data and information related to this case study can be found in Uyttendaele et al. (2022). A regular laptop with an 11th Gen Intel Core i7-1185G7 @ 3.00 GHz processor, 16 GB RAM was used to run the experiments.

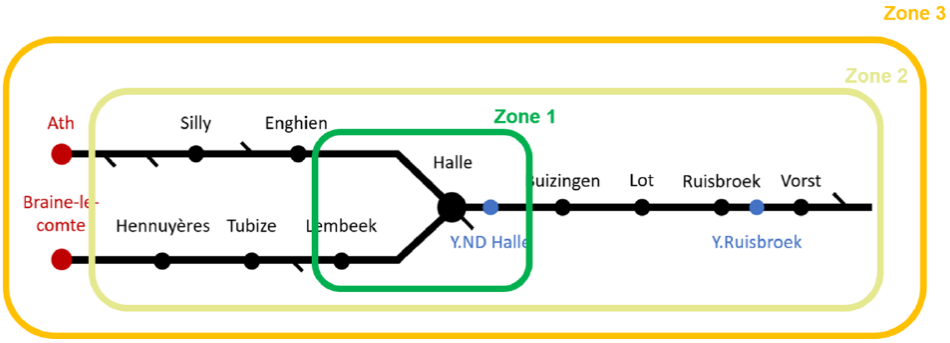

Figure 1: Macroscopic view of the considered part of the network.

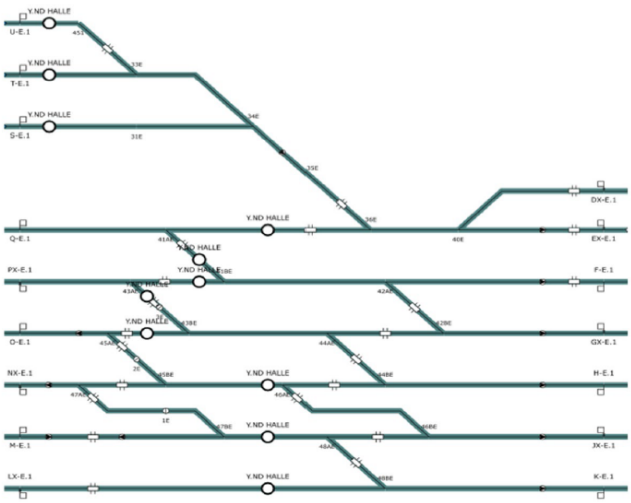

Figure 2: Microscopic view of the switch area Y.ND Halle, which corresponds to the blue dot in zone 1 indicated in figure 1.

The problem is considered for zone 1 and 2 with 5 and 10 possible paths for each train. Only the routes with the shortest duration are considered. It is unlikely that longer routes will lead to good solutions because this typically corresponds to routes with some detours that are not desirable in practice. As stated before, solving this problem with a MILP model is not possible in a reasonable amount of time. After 48 hours of running for zone 1 with 5 alternative routes, the best objective found was 68.08, but there was still an optimality gap of 25%, so no conclusions can be drawn from this. Therefore, the heuristic algorithm was developed. Because the heuristic contains randomness, it is run 10 times for each instance. Each run has a maximum allowed computation time of 30 seconds. The best obtained objective and the average objective are reported in table 1. The improvement compared to the initial objective is given between brackets. This initial objective is the objective after the `find initial solution` is applied.

Table 1: Results of the heuristic algorithm for zone 1 and 2 with 5 and 10 possible routes.

	Initial objective	**Heuristic: best (improvement)**	**Heuristic: average (improvement)**
Zone 1 - 5 routes	79.03	69.63 (−11.89%)	71.85 (−9.09%)
Zone 1 - 10 routes	79.03	69.93 (−11.51%)	71.01 (−10.15%)
Zone 2 - 5 routes	205.97	180.11 (−12.56%)	183.03 (−11.14%)
Zone 2 - 10 routes	205.97	184.15 (−10.59%)	184.43 (−10.46%)

The heuristic clearly finds solutions with a lower objective value than the initial objective. For the different instances, the best obtained objective is quite close to the average objective. This indicates that the heuristic performs consistently over the different runs. For both zones, the best solution that is found for 5 possible routes is better than for the case with 10 possible routes, although the difference for zone 1 is quite small. This is an unexpected result because the solution that is found for 5 possible routes is of course also a possible solution when 10 routes are considered. The fact that these solutions are not found by the heuristic is possibly a consequence of the larger search space in the case with 10 routes.

Because the optimal solution is not known for any of the instances, some additional experiments are conducted to gain some insight in the performance of the heuristic. For both zone 1 and 2 with 5 possible routes, 5 instances are generated by randomly selecting 10 trains to consider instead of all the trains that use the zone. For these instances, the optimal solution can be calculated by a MILP model in only a few seconds. It can then be compared with the solution found by the heuristic. Like before, the heuristic is run 10 times, with a computation time of 30 seconds for each run. The results for zone 1 and 2 are presented in table 2 and 3, respectively. The results show that the heuristic is able to find good solutions, i.e. with a small gap. For instances A and D of zone 2, there is a larger gap. These experiments show that the heuristic is indeed capable of finding near-optimal solution. This does however not guarantee that this is also the case for the larger instances of the case study.

Table 2: Results of the smaller instances for zone 1.

	Optimal objective	**Heuristic: best (gap)**	**Heuristic: average (gap)**
A	13.40	13.41 (+0.07%)	13.56 (+1.19%)
B	11.19	11.19 (+0.00%)	11.20 (+0.09%)
C	11.01	11.13 (+1.09%)	11.56 (+5.00%)
D	13.62	13.69 (+0.51%)	13.82 (+1.47%)
E	9.16	9.17 (+0.11%)	9.17 (+0.11%)

Table 3: Results of the smaller instances for zone 2.

	Optimal objective	**Heuristic: best (gap)**	**Heuristic: average (gap)**
A	5.14	6.21 (+20.82%)	6.21 (+20.82%)
B	9.64	9.66 (+0.21%)	9.66 (+0.21%)
C	15.33	15.39 (+0.39%)	15.88 (+3.59%)
D	13.05	14.09 (+7.97%)	14.75 (+13.03%)
E	11.18	11.50 (+2, 86%)	11.58 (+3.58%)

5 Conclusions

In this paper, a heuristic is presented to improve the robustness of a given timetable by adjusting the timing and routing of the trains. Robustness is measured by considering the buffer times between train pairs. The developed algorithm consists of separate steps to first update the routing and then the timing of the trains. The heuristic is applied to a bottleneck area of the Belgian network. The results for the different instances show that the value of our objective can be improved by about 10% by considering alternative routing options and not only changing the timing. Additional experiments with smaller instances show that the heuristic is capable of finding near-optimal solutions.

Acknowledgements

The authors would like to thank Infrabel, the Belgian railway infrastructure manager, and NMBS, the Belgian railway passenger operator, for this collaboration and their support.

References

Andersson, E., Peterson, A., & Törnquist Krasemann, J. (2015). Improved railway timetable robustness for reduced traffic delays - a milp approach. *Proceedings of the 6th International Seminar on Railway Operations Modelling and Analysis (RailTokyo2015)*.

Burggraeve, S., & Vansteenwegen, P. (2017). Robust routing and timetabling in complex railway stations. *Transportation Research Part B*, *101*, 228–244.

Dewilde, T., Sels, P., Cattrysse, D., & Vansteenwegen, P. (2014). Improving the robustness in railway station areas. *European Journal of Operational Research*, *235*, 276–286.

Högdahl, J., Bohlin, M., & Fröidh, O. (2019). A combined simulation-optimization approach for minimizing travel time and delays in railway timetables. *Transportation Research Part B*, *126*, 192–212.

Jovanović, P., Kecman, P., Bojović, N., & Mandić, D. (2017). Optimal allocation of buffer times to increase train schedule robustness. *European Journal of Operational Research*, *256(1)*, 44–54.

Lusby, R., Larsen, J., & Bull, S. (2018). A survey on robustness in railway planning. *European Journal of Operational Research*, *266(1)*, 1–15.

Uyttendaele, J., Van Hoeck, I., Besinović, N., & Vansteenwegen, P. (2022). Timetable compression using max-plus automata applied to large railway networks. *TOP*, 1–26.