35th European Modeling & Simulation Symposium
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2023 The Authors.
doi: 10.46354/i3m.2023.emss.004

Improved fuzzy decision support system for human-realistic overtaking in railway traffic simulations

Tomáš Vyčítal[1,*]

[1]University of Pardubice, Studentská 95, Pardubice, 532 10, Czech Republic

[*]Corresponding author. Email address: tomas.vycital@student.upce.cz

Abstract

In a simulation model of a railway system it is a necessity, to have realistic train overtaking behavior during disturbances. Many simulation tools address this problem by using simple overtaking solutions, such as look ahead. These tend to be non-intuitive to configure and often not very human-realistic. The goal of this paper is to design a fuzzy system that can easily be used to simulate human-realistic overtaking behavior. A fuzzy system has been chosen because this approach is well-suited for human-style decision making. The presented fuzzy system considers timetables, train positions, delays, buffer times etc. as inputs and provides instructions to the simulation tool whether given train should or should not be overtaken. Finally the performance of the system is evaluated by a case study simulating a part of the railway system in the Czech Republic around the town of Pardubice.

Keywords: decision-making support, fuzzy systems, railway, simulation, transport

1. Introduction

Overtaking is a common occurrence in railway traffic and therefore it is a necessity to have realistic overtaking behavior in railway traffic simulations. A simulation ran without any overtaking would be very unrealistic and would not be able to provide any useful outputs. Similarly a simulation with poorly facilitated overtaking also wouldn't be able to provide any useful outputs. It is therefore something that must be addressed in every railway traffic simulation (excluding some more limited systems where overtaking is not possible at all or not usually performed, such as many subway systems).

Configuring overtaking behavior in a simulation tool can be rather tricky, especially for newcomers. Commonly used simulation tools provide some built-in overtaking solutions, but these are usually very simple and not very human-like. They also tend to require quite extensive knowledge of their inner workings to configure in a satisfactory way. Atop of that the configuration typically cannot be transferred between different models, because it is very specific to the model it was created for.

This paper's goal is to create a decision support system that would be able to make decisions about overtaking in a human-like manner. This should be possible without the need for model specific configuration and especially weird hacks to achieve the desired overtaking behavior. The decision support system should also be able to adjust itself not only to the model as a whole but also to the nature of the infrastructure and traffic in different parts of the model.

2. State of the art

The typical solution to the problem of overtaking in railway traffic simulations is to use look aheads. These are typically out-of-the-box built into the simulation tools and are usually very simple implementation wise and with very good performance. However, they are not very human-like and can be quite tricky to configure well. They are

also prone to misbehaving, most notably commonly making trains wait to be overtaken on main lines and then overtaking them on speed limited sidings.

They also have issues with diverse models, where the look ahead distance has to be picked more as a compromise than an actually ideal value. In such a model a part with longer stretches between stations would need longer look ahead distances to achieve the same level of realism as a part with shorter stretches between stations. In some tools this may not be possible at all, as the look ahead distance is a global setting (e.g. in OpenTrack it's configured per train category but stays the same across the model). Leading to overtaking almost never happening in some parts of the model. In other parts trains can be blocked for overtaking event though the overtaking train is still quite far away and there are other stations further ahead the line where the overtaking could be performed.

Since overtaking is so common in railway traffic, it is a well researched topic. There are many papers that touch on the subject of overtaking in from various perspectives. In Josyula et al. (2020) the authors explore the problem of overtaking in the context of train timetable rescheduling. They propose a method for finding a diverse set of possible solutions that are then presented to a dispatcher as an aid in their work. The dispatcher then picks the best solution from the set, modifies it to their liking or discards it altogether.

The solutions provided would be rather difficult to implement in a simulation tool, as they would require a whole second decision support system to be implemented. The seconds system would have to be able to evaluate and possibly even modify the solutions provided to apply them to the simulation. This would be a very complex task and would require a lot of work to implement. Given the wildly different goal of the paper, it would most likely be better to start from scratch (Josyula et al. (2020) could be handy as inspiration).

Other papers such as Obara et al. (2018), Cavone et al. (2022) and many others focus on real world applications. Some of them are easier to implement in a simulation tool than others. A subset of these was even tested in various forms of simulations, though not necessarily in off-the-shelf simulation tools that would be usable by the general public (e.g. having an at least somewhat user-friendly GUI). However almost none of them are focused on simulations specifically, mostly as dispatcher aids or automatic train control systems for real world applications. There also aren't many papers that would focus solely on the overtaking problem, most of them focus on the larger problem of train timetable rescheduling (which usually includes reordering AKA overtaking as one of it's parts).

The use of fuzzy systems in this type of problem is not very common. This used to be explored more in the past, though again with focus on real world applications rather than simulations. Examples can be a fuzzy control system for dispatching on a mining railway in Brazil explored in Vieira and Gomide (1996) or a dispatcher aid implemented with the help of a fuzzy system in Fay and Schnieder (1997), the goal here is very similar to Josyula et al. (2020). Recently the research has moved to linear programming, neural networks and the like. With papers like Zhu et al. (2020) based on Q-learning, Cavone et al. (2022) mixed integer linear programming or Obara et al. (2018) using reinforcement learning (specifically Deep Q-Network).

This being said, fuzzy systems are pretty common in railway applications, just not in the context of overtaking nor rescheduling. Fuzzy systems has, in general, not only in railway related applications, moved mostly to solving smaller technical problems. These, specifically in the scope of railways, include brake assistance, as explored in Tsuneyoshi et al. (2022), a study comparing different fuzzy inference types. Other focus on vibration mitigation, as explored in Sezer et al. (2011) and many other topics and papers.

The fuzzy system developed as part of this paper is a continuation of my prior work on Vyčítal (2022), which was more a proof of concept than a fully fledged solution, though it did work. More generally it is a continuation of my work on Vyčítal and Bažant (2020) and Vyčítal and Bažant (2021). The algorithm from Vyčítal and Bažant (2021) will serve as the basis for the fuzzy system developed in this paper, being the best performing of my work so far.

The system presented in Vyčítal and Bažant (2020) was a very simple system that was able to perform overtaking. It quite well addressed the issues of overtaking over sidings OpenTrack's built-in solution is prone to. However it had issues with trains overtaking each other repeatedly, requiring quite high thresholds to prevent this, which in turn cut significantly down on overtaking in general.

This issue was mostly mitigated in Vyčítal and Bažant (2021), however it brought the necessity of manually configuring bonuses and penalties for individual categories of trains. What also came up in the work following the paper, was the need to use more advanced approach to creating more complex decision support systems. This is where Vyčítal (2022) comes in, it explores the use of fuzzy systems for this purpose. However this was more of a proof of concept than a fully fledged solution, as it was not able to perform as well as the algorithm from Vyčítal and Bažant (2021).

3. Fuzzy logic

In fuzzy logic, discrete input values are mapped to words of linguistic variables, each with certain degree of membership. These words are then used by rules (in the form of IF-THEN statements) that execute the desired logic. The rules then map to fuzzy words of linguistic variables on the output side, again each with certain degree of membership based on the degrees of membership of the input words used by given rule. The output linguistic variables are then defuzzified, that is converted into discrete values.

A simple example relevant to the topic of this paper

could be a single rule fuzzy system that decides whether to overtake or not based on the ETA at the next station. The rule would be: IF train ahead's ETA is late and train behind's ETA is soon THEN overtake is strong yes. The input variables would be the ETAs for each train and the words "late" and "soon". The output variable would be "overtake" with the words "strong yes", "weak yes" and "no". An input of 5 min for the train ahead and 3 min for the train behind would map to the words "late" with strength 0.8 and "soon" with strength 0.2 for the train ahead and "soon" with strength 0.8 and "late" with strength 0.2 for the train behind. Executing the rule would then yield output of "yes" with strength 0.8 (another rule could also add "no" with some strength to the mix at the same time and so on). This can be then defuzzified to a discrete value of overtake equals 83 % if we define the range as 0 meaning no and 1 meaning yes with equally distributed words on the range by center of gravity defuzzification. Since the output is now a discrete value, it can be used normally outside of the fuzzy system (e.g. we can decide that any recommendation above 60 % is good enough to plan the overtaking maneuver).

4. Fuzzy system

From design perspective a decision has been made to always consider two trains in isolation and compare the differences between the two to make a decision whether to overtake or not. The main goal is to abstract away from any specifics of the model or even the trains themselves. This way, it's possible to use the same fuzzy system for any model and any number of trains. This idea has already been tested as a proof of concept in my previous work in Vyčítal and Bažant (2020) (comparing differences between trains to decide overtaking) and in Vyčítal (2022) (feeding the differences into a fuzzy system to decide overtaking) and proved to be a good approach.

There are some caveats to this approach however. The most obvious is that there are more than two trains in a typical model and overtaking can very easily affect many trains at once, both positively and negatively. This isn't so severe issue that it would prevent the use of this approach in practice, in the end approaches such as look ahead don't evaluate the effects on other trains either and are still commonly used in practice with good results. It does, however, likely limit the maximum possible potential that this approach could achieve.

Another caveat comes from system designer's perspective and that is that all inputs have to be preprocessed and normalized to be comparable and independent of the model. This can lead to a bit cryptic inputs to the fuzzy system, which are a bit harder to understand than the raw absolute values. However, considering how difficult it would be to design a fuzzy system that would work with raw absolute values and achieve the same model independence, this is a good trade-off.

4.1. Implementation

The fuzzy inference system is implemented in Mamdani EH (1974) style. On the technical side of things, TypeScript Microsoft (2023) running on Node.js Foundation and contributors (2023) is used with the @thi.ng/fuzzy library Schmidt (2020) facilitating general fuzzy logic to avoid reinventing the wheel. The simulation itself is then running in the simulation tool OpenTrack Huerlimann (2017). Communication between the simulation tool and the fuzzy system is done via custom tool utilizing OpenTrack's SOAP API Huerlimann (2020). This tool further avoids reinventing the wheel with the use of other well known libraries like Axios Zabriskie (2023) to send commands to the simulation tool, Fastify Fastify (2023) to receive messages from the simulation tool and others.

The overtaking maneuver planning is done via a series of route blocking commands send to the simulation tool. The routes are blocked for the train that is supposed to wait for the other train to overtaken it. Concretely the route through the main line track is blocked, so that the train will use some of the more distant tracks to wait for the other train to overtake it. Also all of the routes out of the station are blocked, so that the train will not be able to depart before the other train has overtaken it and the routes are unblocked again.

This method has been chosen as it allows for custom overtaking without reimplementing all of the logic that the simulation tool already has for route planning. OpenTrack continues to make all of the decisions as normal it just knows that it's not supposed to let given train on given routes. There wouldn't be much of an advantage in taking complete control over the route planning.

4.2. Input data

The input variables are based on exports from the simulation tool (static things like timetables or infrastructure) and the current state of the simulation (dynamic things like train positions or delays). These are then preprocessed into a few values that are easier to use for comparisons in the fuzzy system. Some are simply subtracted from one another, after being taken for the two trains in consideration (e.g. stop frequency). Other are combined from multiple sources to form a single value before being subtracted from one another to create the differential input value (e.g. ETA).

The core input value is the difference in ETA between the two trains in consideration. The basic idea behind the ETA is when would the train arrive at the station in question if there were no other trains on the railway. The exact formula is taken from Vyčítal and Bažant (2020):

$$e = a + \max(0, d - b) \tag{1}$$

where e is the ETA, a is the timetabled arrival of the train, d is current delay and b is the timetabled buffer time. This is then calculated for each train in consideration and the

difference between the two is taken and used as for the fuzzy variable depicted in 1.

The variable is split into 4 words: "soonish", "slightly later", "later" and "very later". The word soonish represents ETA differences where the train ahead is expected to arrive sooner, at about the same time or up to 160 s later. There generally isn't much point in differentiating between these cases as the overtaking will likely cause about the same if not higher delay and won't be worth it, at least not right now, it may be worth it later on. The next word slightly later represents the next 40 s of ETA difference where overtaking starts looking more appealing. The word later is further 40 s of ETA difference after that and is the point where overtaking is generally a good idea. With the last word very later capturing the rest being mostly a no-brainer for overtaking.

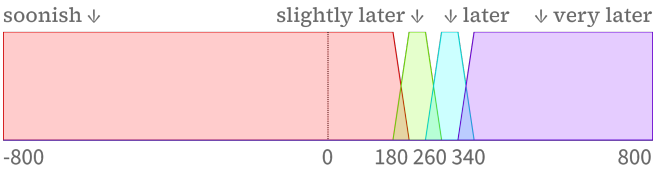

Figure 1. An input variable for the difference in ETA (in seconds) between the two trains in consideration.

Next input is the difference in stopping frequency. Stopping frequency is defined as the percentage of stops at which the train stops compared to the total number of stops on the path the train is timetabled to take. The figure 2 provides visualization of the variable. The variable is designed to provide the ability to penalize or reward overtaking based on the difference in stopping frequency of the two trains in consideration. Since there are no hard anchor points for this variable, the words simply taper of towards the middle to give a transition between the words and therefore to allow the rules to apply more strongly where the difference is higher.

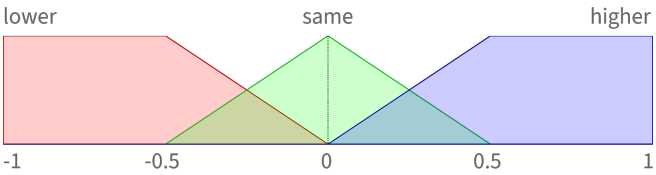

Figure 2. An input variable for the difference in stopping frequency between the two trains in consideration.

Lastly there is a variable 3 indicating whether the train ahead, that may stop to be overtaken, is timetabled to stop there anyway. This isn't actually a fuzzy variable, it's binary. Either the train is timetabled to stop there and it will be 1 or it isn't and it will be −1. They idea is to be able to take advantage of the opportunity to overtake a train that is going to stop either way and avoid the delays associated with an unplanned stop.

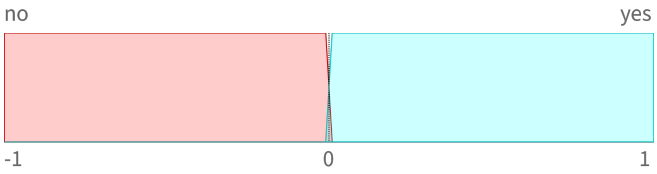

Figure 3. An input variable indicating whether the train ahead is timetabled to stop where the overtaking may happen.

Compared with my previous attempt at creating a fuzzy decision system for overtaking from Vyčítal (2022), two input variables has been completely omitted. They turned out to be anywhere between useless and detrimental to the performance of the system, depending on the rules that used them as inputs. I was simply unable to find a way to beneficially include them in the system.

The first omitted input is the difference in top speed between the two trains in consideration. This turned out to be a bad idea as the top speed of a train is not a good indicator of the train's ability to overtake. The reason is that the top speed is not the speed at which the train is currently traveling or will be traveling later on, but the maximum speed it can travel at. Trains often travel at a lower speed due to speed restrictions imposed by the infrastructure (e.g. the model used in the case study contains trains with top speeds as high as 230 km h^{-1} but no infrastructure above 160 km h^{-1}). The speed the train will actually travel at is also already taken into account in the ETA calculations as it is a part of the timetable (i.e. a train that can't go more than 100 km h^{-1} is not going to be timetabled to cover 100 km in less than 1 h).

The second omitted input is the difference in priority between the two trains in consideration. In this case, I tried considerably more approaches to including it in the system, but none of them worked well. Many were outright detrimental to the performance of the system, causing weird overtaking choices that should generally be avoided. This isn't ideal as priority is very commonly considered when deciding whether to overtake or not.

The reason for this is that the priority of a train is not necessarily indicative of how quickly will the train pass through the railway section in question. Two trains that will behave exactly the same in given section may have different priorities assigned to them (quite a few do in the case study included in this paper). This can lead to situations where lower priority train stops at a siding, lets the higher priority train pass and then continues on its way. Gaining delays in the process and forcing the higher priority train to decelerate as it slows down on the main line before entering the siding, generating delays for the higher priority train as well. After all of this, the two trains then continue to their destinations, the higher priority train now being ahead the lower priority train, both with

higher delays than before, with absolutely no benefit to either of them. Potentially also causing delays to other trains that may be following them.

The ETA input variable has also been significantly changed compared to the version in Vyčítal (2022). It turned out that having multiple words around 4 min mark is more interesting for creating rules than differentiating between inputs close to 0 min. Overtaking generally adds delays in the short term so there has to be a significant future benefit to overtake in order to make it worth it.

4.3. Output data

Since the purpose of the fuzzy system is to recommend whether to overtake or not, the output consists of a single variable indicating the strength of the recommendation to overtake. The variable can be visually seen in 4 and it consists of five words. The words are evenly spaced out to allow the rules to express recommendation to overtake and not to overtake with two levels of strength and also no preference.

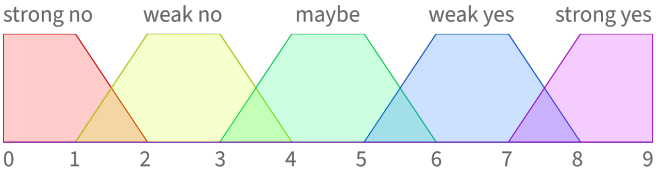

Figure 4. The output variable for the strength of the recommendation to overtake.

To use the output in the simulation tool, which expects a discrete order to overtake or not, the output is defuzzified using the center of gravity method. This yields a discrete number which is further reduced to a binary value by comparing it to a threshold of 4.5, values through 4.5 mean not to overtake and values above 4.5 mean to overtake. If the result is to overtake, the simulation tool will then be instructed to plan an overtaking maneuver. If the result is to not overtake, the simulation tool will either receive no instructions and continue as normal or it will be instructed to cancel a previously planned overtaking maneuver.

4.4. Rules

Most of the rules compare differences and more complex values created as described in the previous section. This preprocessing is much simpler and more performant when done in classic programming languages than in the fuzzy logic system as combing and compering the raw values would very quickly explode into a huge number of rules and would be difficult to be made independent from concrete model or even station. Thanks to this the rules are relatively simple and there aren't that many of them, yet they are crucial to the decision support system's performance as they combine them all together and create the final recommendation to overtake or not.

1. IF train ahead's ETA is very later THEN recommendation to overtake is strong yes WEIGHT 1
2. IF train ahead's ETA is later THEN recommendation to overtake is weak yes WEIGHT 1
3. IF train ahead's ETA is slightly later THEN recommendation to overtake is maybe WEIGHT 1
4. IF train ahead's ETA is soonish THEN recommendation to overtake is strong no WEIGHT 1
5. IF train ahead's stop frequency is lower THEN recommendation to overtake is weak no WEIGHT 2
6. IF train ahead's stop frequency is same THEN recommendation to overtake is maybe WEIGHT 1
7. IF train ahead's stop frequency is higher THEN recommendation to overtake is weak yes WEIGHT 1
8. IF train ahead stops anyway THEN recommendation to overtake is weak yes WEIGHT 1
9. IF train ahead doesn't stops anyway THEN recommendation to overtake is weak no WEIGHT 1

The backbone of the decision are rules number 1–4. They analyze the difference in ETA between the two trains in consideration. This on its own is pretty much what Vyčítal and Bažant (2020) is based on, however here it's combined with additional rules, which will be described later, instead of a single threshold to plan or cancel overtaking. Overtaking is strongly recommended if the train ahead is expected to arrive substantially later than the train which is considering overtaking. On the other hand if the train ahead is expected to arrive sooner, at the same time or only slightly later, overtaking is strongly not recommended. There are also two intermediate levels of recommendation, in case the difference in ETA is not as clear.

Rules number 5–7 analyze the stopping frequency of the trains in consideration. The reasoning behind this is that the more the train stops the more likely it is to cause a lot of delays to the trains behind it in case of further disturbances (e.g. an intercity train catching up with a commuter train and being stuck behind it running slowly or worse, stopping on red signals, because the commuter train has to stop at every stop along the way). On the other hand a train stopping often is unlikely to ever catch up with a train stopping rarely. These rules therefore discourage overtaking when the train that is being overtaken stops less often than the train that is considering overtaking and vice versa. The negative recommendation is stronger than the positive one as overtaking carelessly is quite likely to cause a lot of delays to all trains involved even those involved only indirectly.

Finally rules number 8–9 analyze whether the train ahead is going to stop regardless of whether the overtaking is planned or not. If the train ahead does stop anyway, then further recommendation to overtake is added. However if the train would otherwise pass without stopping, then a recommendation against overtaking is added. This is because if the train ahead is going to stop anyway, then it's only going to be held in the station longer, no other

delays, when compared to the timetable, are going to be caused to it. In the case that the train would otherwise pass without stopping, the time spent decelerating to stop, acceleration to get back to speed, spent waiting in the station and by running on the longer path on the siding and atop of that most likely at lower top speed are going to be added as none of that was timetabled for. It can also affect the train behind it as it itself might have to slow down to keep the safe distance and then accelerate back to speed, in worst case scenario it might encounter red signal and stop completely. For these reasons overtaking when the train that is to be overtaken stops anyway due to it's timetable is highly advantageous.

5. Evaluation

To evaluate the fuzzy system a simulation was run in the simulation tool OpenTrack with and an algorithm from my older paper Vyčítal and Bažant (2021). The algorithm from Vyčítal and Bažant (2021) was used with the same configuration as in the paper however the model has been mildly modified in the meantime and more importantly the code connecting the algorithm to the simulation tool has been significantly changed. These changes fixed some race conditions that were present in the original implementation and made the runs replicable. It nevertheless changed the results slightly, though the changes affect all runs more or less equally (i.e. rerunning the case study comparing Vyčítal and Bažant (2020) and Vyčítal and Bažant (2021) as presented in Vyčítal and Bažant (2021) yields somewhat different numbers but the same conclusions).

As mentioned before, in general terms the model is still the same as in Vyčítal and Bažant (2021), that is: The simulation model consists of a part of the Czech railway network, specifically around the town Pardubice. Overall the model contains over 50 km of tracks with multiple stations and stops. Topologically the infrastructure is configured in Y shape with a branch line splitting in Pardubice and continuing through Rosice providing inflow to and outflow from the main line. With this the fuzzy system is presented with traffic merging and splitting in different directions, there is also traffic crossing the main line from the branch line and vice versa. Both the main line and the branch lines are double-track with very dense mixed traffic. The traffic used is typical for the area, however is not strictly speaking real world as the area is currently undergoing a major reconstruction and the model represents the situation after the reconstruction.

To prevent interference with decisions made externally by the fuzzy system from this paper and the algorithm from Vyčítal and Bažant (2021), the simulation was configured with look ahead distance of 0 m. Setting it to anything else could result in deadlocks, where OpenTrack would be blocking one train from moving forward, because it would be waiting for another train, whose look ahead reached beyond the first train, to move forward. While at the same time the second train would be blocked in place by the fuzzy system's overtaking decision. Even if that wouldn't happen, it would still void the results as it would be difficult to evaluate the performance of the fuzzy system, if it was fighting with by the built-in look ahead of OpenTrack.

The delays were again configure the same as in Vyčítal and Bažant (2021): According to the SŽDC SM124 guideline SŽDC (2019) (up to 2 h on entry, average delays are bellow 10 min. for passenger traffic), SŽ (formerly SŽDC) is the local railway infrastructure manager. Exactly the same delay scenarios used for the fuzzy system from this paper were also used for the algorithm from Vyčítal and Bažant (2021).

The simulation consisted of 30 runs with the fuzzy system from this paper and another 30 runs with the algorithm from Vyčítal and Bažant (2021), each run containing 2 h of traffic.

6. Results

The high level overview of the results of the simulation is visualized in 5. The fuzzy system presented in this paper was able to match or outperform the reference algorithm from Vyčítal and Bažant (2021) in every measured category.

The reference algorithm from Vyčítal and Bažant (2021) had to be manually configured with bonuses and penalties for each train category to achieve the best results. Meanwhile the fuzzy system from this paper was able to achieve its results based solely on infrastructure exports from OpenTrack and messages sent during the simulation via OpenTrack API. Both of these were also used by the reference algorithm from Vyčítal and Bažant (2021) in addition to the manually input bonuses and penalties.

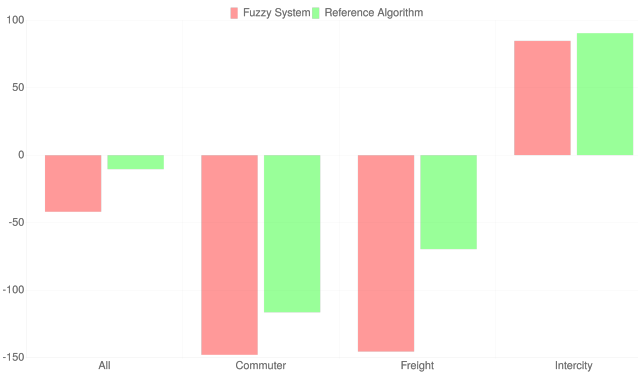

Figure 5. The average change in delay in seconds (y axis) per category of trains (x axis) across the 30 runs.

The bar chart in 5 shows the average change in delay in seconds across all of the trains and some interesting categories of trains. The change in delay is calculated as the difference between the delay at the destination and the delay at the entry to the model, where negative means decrease in delay and positive an increase in delay. The delay at the destination is capped at 0 s from the bottom,

as there is little benefit in decreasing the delay further (the train will wait for it's scheduled 0 s delayed departure at the next station anyway no matter how early it arrived).

Apart from the all category that contains all of the trains, the data is split by the type of traffic. Freight trains are grouped together in the freight category, these are quite varied in their behavior but they all have lower priorities than passenger traffic. Passenger trains that stop at most (usually all) passenger stations and stops are grouped in the commuter category. Whereas passenger trains that stop only at bigger passenger stations are grouped in the intercity category.

As mentioned before, the all category saw noticeable decrease in delays. Inspecting the individual categories reveals that intercity traffic gains delays and the other two categories decrease their delays. This makes sense as there are very small reserves timetabled for the intercity traffic while the other two categories have more generous timetables in this regard, making it easier to decrease their delays.

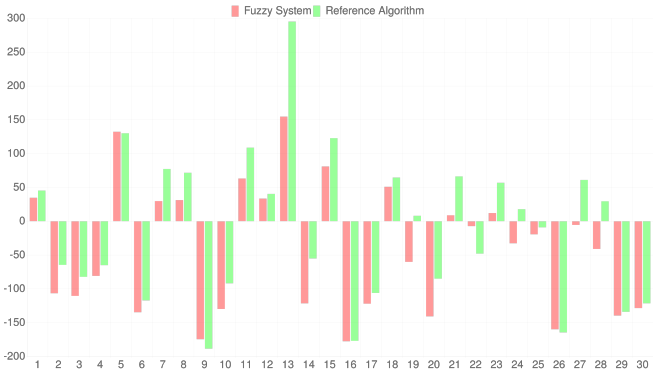

Figure 6. The confidence interval for the average change in delay in seconds across the 30 runs, run numbers are plotted on the x axis and the delay gained while passing through the model on the y axis.

When comparing the fuzzy system from this paper to the reference algorithm from Vyčítal and Bažant (2021), the fuzzy system performed better everywhere except the intercity category. The difference in intercity category is the only one that is not significant (95 % confidence level). There is quite high variability across runs (as can be seen in 6) and the difference is not big enough to confidently say that it means something. Overall the fuzzy system managed to achieve about the same delay change for intercity traffic as the reference algorithm from Vyčítal and Bažant (2021) without sabotaging the commuter and freight traffic to the same extent.

When put to absolute numbers (see 7), it can be seen that the delays are relatively high and the change in delay comparatively small. This is caused in part by the fact that the model is quite small (the main line track across the model has 50 km) and therefore even if a good overtaking decision is made, it can't do miracles, there simply isn't enough of time to take down 15 min from a train's delay.

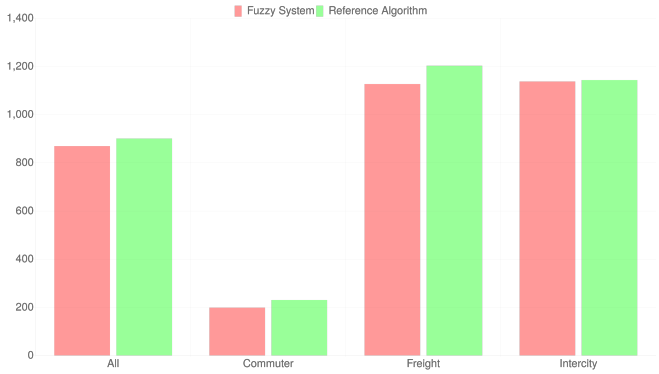

Figure 7. The average delay in seconds (y axis) per category of trains (x axis) at their destination across the 30 runs.

Another contributing factor is the high utilization of the main line, which is utilized to its full capacity. This leads to delays very easily cascading from one train to the next as there are very short separation times between trains.

7. Conclusion

The fuzzy system presented in this paper has proved to be a viable solution to the problem of overtaking in railway traffic simulations. It is able to adjust itself effortlessly to the model it is used in based solely on the infrastructure exports from OpenTrack and messages sent during the simulation via OpenTrack API. It is also an improvement when compared to the reference algorithm from Vyčítal and Bažant (2021) in both achieved results and easy of use.

The fuzzy system presented in this paper is of course not perfect nor all encompassing. Further works could focus on incorporating priority into the system as an optional configurable input (many railways use priority in various ways and here it's completely ignored). Another possible improvement could be to incorporate a safe guard against repeated overtaking, the system isn't very prone to a train getting stuck for a very long time but it happens that a train is overtaken multiple times in a row and there is no mechanism to limit that.

8. Funding

This work was supported by Internal Grant Agency of University of Pardubice in the frame of Student Grant Competition 2023.

References

Cavone, G., van den Boom, T., Blenkers, L. L., Dotoli, M., Seatzu, C., and de Schutter, B. (2022). An mpc-based rescheduling algorithm for disruptions and disturbances in large-scale railway networks. *IEEE Transactions on Automation Science and Engineering*, 19:99–112.

EH, M. (1974). Application of fuzzy algorithms for the control of dynamic plants. *Proc. IEEE*, 121(12):1583–1588.
Fastify (2023). Fastify, fast and low overhead web framework, for node.js.
Fay, A. and Schnieder, E. (1997). Fuzzy expert system for real-time dispatching of maglev train traffic. *IFAC Proceedings Volumes*, 30:681–685.
Foundation, O. and contributors, N. (2023). Node.js.
Huerlimann, D. (2017). *OpenTrack*.
Huerlimann, D. (2020). *OpenTrack*.
Josyula, S., Törnquist, J., and Lundberg, L. (2020). Parallel computing for multi-objective train rescheduling. *IEEE Transactions on Emerging Topics in Computing*, PP.
Microsoft (2023). Typescript.
Obara, M., Kashiyama, T., and Sekimoto, Y. (2018). Deep reinforcement learning approach for train rescheduling utilizing graph theory. pages 4525–4533.
Schmidt, K. (2020). @thi.ng/fuzzy. https://thi.ng/fuzzy.
Sezer, S., Çetin, c., and Atalay, A. (2011). Application of self tuning fuzzy logic control to full railway vehicle model. *Procedia CS*, 6:487–492.
SŽDC (2019). *SM124: Zjišťování kapacity dráhy*. Prague. https://thi.ng/fuzzy.
Tsuneyoshi, M., Ikeda, M., and Barolli, L. (2022). *A Brake Assisting Function for Railway Vehicles Using Fuzzy Logic: A Comparison Study for Different Fuzzy Inference Types*, pages 301–311.
Vieira, P. and Gomide, F. (1996). Computer-aided train dispatch. *Spectrum, IEEE*, 33:51 – 53.
Vyčítal, T. (2022). Fuzzy decision support system for human-realistic overtaking in railway traffic simulations. *2022 IEEE 16th International Scientific Conference on Informatics (Informatics)*, pages 345–350.
Vyčítal, T. and Bažant, M. (2020). Train overtaking at railway stations within simulation models of railway lines. *Proceedings of the 32nd European Modeling & Simulation Symposium (EMSS 2020)*, pages 28–34.
Vyčítal, T. and Bažant, M. (2021). Algorithm for train overtaking at railway stations within railway line simulation models with parameter fine-tuning. *Proceedings of the 25th international scientific conference TRANSPORT MEANS 2021*, pages 666–671.
Zabriskie, M. (2023). Axios.
Zhu, Y., Wang, H., and Goverde, R. (2020). Reinforcement learning in railway timetable rescheduling.