FIT5216: Assignment 3 – Report

Testing assertion

Assertion 9: The Line Type Agrees with the Travel Time

```
constraint forall(st1, st2 in STOP)(
                assert(line[st1, st2] = NONE <->
                (travel_time[st1,st2] = <> \/ travel_time[st1,st2] = 0),
                  "line_type and travel_time of (\(st1), \(st2)) do NOT agree!\n
line_type = \(line[st1, st2]) but travel_time = \(travel_time[st1,st2])"));
```

This assertion ensures consistency between the line type and travel time data. Specifically, if the line type is NONE, the travel time must be either absent or zero, and vice versa. To test this assertion, we create a scenario where a line is defined as NONE but has a positive travel time. For instance, in the test data *(test1.dzn)*, the travel time from A to C is 86, but the line type is NONE. Given that the line type is symmetric (as per Assertion 8), both A to C and C to A must have the line type NONE. This violates the constraint and triggers the assertion: *"line_type and travel_time of (A, C) do NOT agree! line_type = NONE but travel_time = 86"*.

Another test case might involve the line type between C and F being SING, but their travel time is absent. After correcting the violation between A and C (e.g., setting travel_time(A, C) = <>), the model raises an error for the latter: *"line_type and travel_time of (C, F) do NOT agree! line_type = SING but travel_time = <>"*. These errors confirm that the assertion correctly identifies inconsistencies, preventing the model from processing erroneous data. Without this check, the model might misinterpret routes, leading to illogical schedules where non-existent or unfeasible connections are considered valid, resulting in invalid optimisation results.

Assertion 10: Route Lengths Agree with the Definitions in Route

```
constraint forall(r in ROUTE, stopno in STOPNO)(
                assert(route[r, stopno] = dstop <-> stopno > rlength[r],
                        "\((r)) has route lengths = \(rlength[r]) but stop at
position \(stopno) = \(route[r, stopno])"));
```

This assertion checks that routes are correctly defined in terms of their lengths and the placement of dummy stops. If a route's length is less than the maximum route length, the remaining stops should be dummy stops, and these dummy stops should only appear at the end of the route. Additionally, it ensures that routes with the maximum length do not contain any dummy stops at any position.

```
rlength = [ 6, 3, 6, 6, 5 ];
route = [| A, B, C, D, E, F
         | G, B, C, dstop, K, dstop
         | F, E, D, C, B, A
         | K, J, dstop, D, H, I
         | I, H, D, E, F, A
         |];
```

To test this assertion, create a scenario where these conditions are violated. For example, consider the test data *(test2.dzn)* with three violations: firstly, Route 2 has a length of 3, but its fifth position is a non-dummy stop 'K' instead of a dummy stop. This triggers the error: *"to_enum(ROUTE,2) has route lengths = 3 but stop at position 5 = K"*. After correcting this by setting 'K' to a dummy stop, the next violation is identified in Route 4, which has a length of 6 (the maximum length) but has a dummy stop at position 3. This results in the error: *"to_enum(ROUTE,4) has route lengths = 6 but stop at position 3 = dummy"*. After correcting this by ensuring all stops in Route 4 are non-dummy, the final violation is found in Route 5, where the last stop at position 6 must be a dummy stop as its actual length is only 5. This triggers the error: *"to_enum(ROUTE,5) has route lengths = 5 but stop at position 6 = A"*. These tests confirm that the assertion correctly identifies misplaced stops, ensuring the data's integrity and preventing invalid route configurations.

Without this check, the model might generate routes with incorrectly placed stops, leading to invalid or infeasible schedules that do not conform to the expected structure, causing potential downstream processing and optimization errors.

Assertion 11: Each Two Consecutive (Non-Dummy) Stops Occurring in a Route are Connected

```
constraint forall(r in ROUTE, stopno in STOPNO1 where route[r, stopno] != dstop /\
route[r, stopno+1] != dstop)(
                assert(travel_time[route[r, stopno], route[r, stopno+1]] != <>,
                "stop \(route[r, stopno]), \(route[r, stopno+1]) are consecutive in
route \(r) but travel_time is absent!"));
```

This assertion ensures that any two consecutive non-dummy stops in a route have a valid travel time connection between them. This is crucial for maintaining the integrity of the route, ensuring that the schedule is feasible and that stops are correctly linked.

To test this assertion *(test3.dzn)*, we create a scenario where two consecutive non-dummy stops lack a valid travel time connection. For example, consider the test case where Route 1 is defined as A, B, C, D, E, F. Ideally, there should be a valid line and travel time connecting each consecutive stop. However, if the line between A and B is set to NONE and their travel time is absent, the model will raise the error: *"Stop A,*

B are consecutive in route to_enum(ROUTE,1) but travel_time is absent!". This error indicates that the assertion correctly identifies the absence of necessary travel time connections.

Without this assertion, the model might produce routes where consecutive stops are not properly connected, leading to infeasible schedules and significant errors in the overall routing logic. This would render the entire schedule invalid and unreliable, causing potential disruptions and inaccuracies in the modelled transportation system.

The importance of Assertion

Assertion 1: Minimal Wait Times are All Non-Negative

If minimal wait times are negative and this is not checked, the model may treat these negative values as if they were zero. This is because the domain of the wait array is TIME, which guarantees non-negative values. This would lead to logical inconsistencies in the schedule, as the stop time for passengers could be zero, causing the departure and arrival times at every stop to be equal. The model might exploit this by not skipping any stops to reduce skip costs and minimize delay penalties by spending minimal time at each stop. This could result in a superficially optimized schedule that doesn't reflect realistic operating conditions. Such a scenario could cause trains to collide or create unsafe conditions for passengers due to the lack of adequate waiting times at stations.

Assertion 2: Skip Cost is Non-Negative, Skip Cost for Non-Ordinary Stations is Zero

If skip costs are negative or incorrect for non-ordinary stations and this is not checked, the model may calculate incorrect costs for skipping stops. This could result in suboptimal routing decisions, where the model might prefer skipping certain stops based on erroneous cost calculations, thus distorting the optimization results. For example, if a stop has a negative skip cost, the model is more likely to skip this stop to reduce the total skip cost. This action could also prevent delays, as skipping a stop means spending less than the minimal wait time required. As a result, the model might optimize for a lower overall cost and reduced delays, but at the expense of bypassing essential stops, which could lead to inefficient and unrealistic service schedules, ultimately compromising the quality and reliability of the transportation system.

Assertion 3: Platform Numbers are Positive, Except for the Dummy Stop

If platform numbers are not positive (except for the dummy stop) and this is not checked, the model may force trains to bypass stops or attempt to assign trains to non-existent platforms. For example, if any actual stop has zero platforms, no train can stay at this stop. The 'stay' time, which is the duration from arrival to departure, would be forced to zero, leading the model to set the arrival and departure times to be the same (implying a wait time of zero) and to skip the stop. However, since only ordinary stops can be skipped, this could lead to unsatisfiable conditions for non-ordinary stops, where stopping is mandatory. This would result in infeasible schedules, causing significant disruptions and operational issues in the railway network, as trains cannot operate correctly without valid platform assignments.

Assertion 4: Wait Times, Skip Cost, and Platform Numbers for the Dummy Stop are All 0

If wait times, skip costs, and platform numbers for the dummy stop are not zero and this is not checked, the model may incorrectly treat the dummy stop as an actual stop. For instance, if the dummy stop has a non-zero wait time, the model will detect the actual stop time of the service incorrectly, leading to inaccurate scheduling. If the skip cost is non-zero, it could add extra cost, causing the model to find a suboptimal solution. This could affect overall timing and cost calculations, resulting in incorrect optimisation results. Additionally, treating the dummy stop as a regular stop might disrupt the intended use of the dummy stop, which is meant to signify the end of a route rather than an actual station. This misinterpretation could lead to further scheduling errors, such as unnecessary delays or inappropriate resource allocation, ultimately compromising the efficiency and reliability of the train schedule.

Assertion 5: Travel Time to the Dummy Stop and from a Stop to Itself is 0

If the travel time to the dummy stop or from a stop to itself is not zero and this is not checked, the model may calculate the overall service time incorrectly. For example, non-zero travel time to the dummy stop could create unnecessary delays in the schedule. This is because shorter routes are padded with dummy stops, and if there is a non-zero travel time from the previous actual station to the dummy stop, this travel time would be incorrectly included in the total service time. Additionally, non-zero travel time from a dummy stop to itself could also inflate the total running time of the service. As travel times are represented in a 2D matrix, it is crucial to maintain accurate data with such assertions to ensure that the model correctly handles travel times and avoids these erroneous calculations. Without this assertion, the model

could produce schedules with inflated service times, leading to inefficient resource utilisation and potential scheduling conflicts.

Assertion 6: Minimum Separation is Non-Negative

(skipped as suggested in the Ed forum:
https://edstem.org/au/courses/14744/discussion/1965431)

Assertion 7: Service End is No Earlier Than Service Start for Each Service

If the service end time is earlier than the service start time and this is not checked, the model may produce invalid service schedules. The model constraints require that the actual start time of any service be greater than the planned start time and the actual end time SHOULD be smaller than the planned end time, with deviations counting as delays. If the end time is incorrectly set earlier than the start time, it creates a logical inconsistency: the planned *start time* < *actual start time* < *actual end time (ideally)* < *planned end time* < *start time*. This contradiction would cause the model to incorrectly calculate delays, significantly boosting the delay objective. The model would struggle with logical reasoning, leading to unrealistic schedules where services appear to end before they begin. This would not only invalidate the optimization but also render the schedule practically unusable, causing operational issues and confusion in the service timelines.

Assertion 8: The Line Type is Symmetric

If the line type is not symmetric and this is not checked, the model may generate inconsistent routing information. For example, it would not make sense to have no route from A to B (line type NONE) while having a route from B to A. Even for single-track routes, a train should be able to go back and forth. This inconsistency could result in errors in route planning and operational inefficiencies. Ensuring symmetry in line types aids in maintaining consistent data, supporting assertions 9 and 11 by ensuring that if one direction is incorrect, both directions must be incorrect, thus preserving the logical integrity of the routing information. Without this assertion, the model might produce illogical schedules that cannot be executed in practice, leading to significant disruptions and inefficiencies in the transportation network.

Assertion 9: The Line Type Agrees with the Travel Time

If the line type does not agree with the travel time and this is not checked, the model may misinterpret the nature of certain routes. For instance, if a line type is NONE but there is a positive travel time, the model might incorrectly assume that the route is valid. Conversely, if the travel time is absent for a defined line, the latter stop could

be considered a dummy stop, causing the model to treat the train as if it can bypass the latter stop entirely. This misalignment would lead to errors in routing and timing calculations. Ensuring this constraint links the data in the line and travel_time matrices enhances the robustness of the model when handling problematic data. Without this constraint, the model could generate illogical schedules where non-existent or unfeasible connections are treated as valid, severely distorting the overall optimization results and leading to unrealistic and inefficient train operations.

Assertion 10: Route Lengths Agree with the Definitions in Route

Padding techniques help developers handle data more easily by grouping it into consistent formats. In this case, padding dummy stops to shorter routes ensures all routes have the same length. If route lengths do not agree with the definitions in the route and this is not checked, the model may incorrectly handle dummy stops. If any dummy stop appears within the actual route length, it indicates that the route length is either overestimated or an actual stop is missing. Conversely, if any actual stop appears after the defined route length, it will unnecessarily increase the total travel time. This could result in routes that include non-dummy stops after dummy stops, creating invalid and infeasible routes that do not conform to the expected structure. Ensuring that route lengths and dummy stops are correctly handled is crucial for generating valid, feasible schedules that align with the expected operational framework.

Assertion 11: Each Two Consecutive (Non-Dummy) Stops Occurring in a Route are Connected

If two consecutive non-dummy stops are not connected and this is not checked, the model may include infeasible routes in the schedule. This would result in routes where travel times between consecutive stops are absent, making it impossible to generate a valid and continuous schedule. For example, if the travel time between two consecutive stops is absent, the model would be unable to accurately calculate arrival and departure times at these stops. This would lead to significant errors in the overall routing logic, potentially rendering the entire schedule invalid and unreliable. Such inconsistencies would disrupt the continuity of service, causing operational inefficiencies and making the schedule practically unusable. Ensuring that all consecutive non-dummy stops are properly connected is essential for maintaining the integrity and feasibility of the train schedules.