Dynamic Bus Scheduling

TIMETABLE GENERATION ALGORITHM

A timetable document is consisted of a database id, a bus line id, a list of timetable entries which contain details about the followed bus route and the number of passengers, and the travel requests which are served by the specific timetable.

```
timetable_document: {
    _id, line_id,
    timetable_entries: [{
        starting_bus_stop: {_id, osm_id, name, point: {longitude, latitude}},
        ending_bus_stop: {_id, osm_id, name, point: {longitude, latitude}},
        departure_datetime, arrival_datetime, total_time,
        number_of_onboarding_passengers, number_of_deboarding_passengers,
        number_of_current_passengers
    }],
    travel_requests: [{
        _id, travel_request_id, client_id, bus_line_id,
        starting_bus_stop: {_id, osm_id, name, point: {longitude, latitude}},
        ending_bus_stop: {_id, osm_id, name, point: {longitude, latitude}},
        departure_datetime, arrival_datetime,
        starting_timetable_entry_index, ending_timetable_entry_index
    }]
}
```

The timetable generation algorithm requires the following **inputs**:

- Bus line (or id): The bus line where the generated timetables correspond to.
- Starting and ending datetime of timetables: The datetime period where the generated timetables correspond to.
- Starting and ending departure datetime of travel requests: Timetables are generated taking into consideration travel requests, with departure datetimes which correspond to the provided datetime period.
- Maximum bus capacity: The timetable generation algorithm ensures that there is no generated timetable, where the number of current passengers exceeds the maximum bus vehicle capacity.

- Minimum number of passengers in timetable: The timetable generation algorithm makes sure that there is no generated timetable, where the total number of served travel requests is lower than the minimum allowed number of passengers. As a consequence, this parameter could be used in order to increase or decrease the number of generated timetables.
- Average waiting time threshold: Used in order to set a lower bound for the timetable generation algorithm. If the average waiting time for the passengers of all timetables is lower than this value, then the algorithm stops operating.

The timetable generation algorithm includes the following **steps of execution**:

1. A connection is established, between the Look Ahead component and the System Database, and the list of bus stops which corresponds to the provided bus line is retrieved.
2. The Look Ahead component retrieves, from the System Database, a list of travel requests with departure datetimes between the provided starting and ending values. These travel requests are evaluated, in the next steps of the algorithm, in order to generate the timetables of the system.
3. A request is sent, from the Look Ahead to the Route Generator, so as the less time-consuming bus route between the bus stops of the bus line to be identified, while taking into consideration the current levels of traffic density. The response of the Route Generator includes details about starting and ending bus stops, covered distance, required traveling time, intermediate nodes, and edges which connect them:

```
route_generator_response: [{
    starting_bus_stop: {
        _id, osm_id, name, point: {longitude, latitude}
    },
    ending_bus_stop: {
        _id, osm_id, name, point: {longitude, latitude}
    },
    route: {
        total_distance, total_time, node_osm_ids, points, edges,
        distances_from_starting_node, times_from_starting_node,
        distances_from_previous_node, times_from_previous_node
    }
}]
```

4. Based on the response of the Route Generator and assuming that there is only one bus vehicle available, the Look Ahead generates some initial timetables which cover the whole datetime period between the provided starting and ending values. Initially, the lists of travel requests of these timetables are empty, since travel requests have not been partitioned yet, and the departure and arrival date-times of the timetable entries are based exclusively on the details of the route. In the next steps of the algorithm, these timetables are used for initializing the of travel requests.

5. (Clustering of Travel Requests): Each one of the retrieved travel requests is corresponded to the timetable which produces the minimum waiting time for the passenger, which is calculated by comparing the departure datetime of the travel request with departure datetime of the corresponding timetable entry.

6. (Handling of Under-crowded Timetables): After the clustering step, there might be timetables where the number of travel requests is lower than the minimum allowed number of passengers in timetable. This is usual during night hours, where transportation demand is not so high. These timetables are removed from the list of generated timetables, and each one of their travel requests is corresponded to one of the remaining timetables, taking into consideration the waiting time of the passenger.

7. (Handling of Over-crowded Timetables): In a similar way, there might be timetables where the number of current passengers is higher than the maximum bus vehicle capacity, which indicates that each one of these timetables cannot be served by one bus vehicle. For this reason, each one of these timetables is divided into two timetables and the corresponding travel requests are partitioned, taking into consideration the waiting time of each passenger. The whole procedure is repeated until there is no timetable where the number of passengers exceeds the maximum bus vehicle capacity.

8. (Adjusting Departure and Arrival Datetimes of Timetables): At this point of processing, the departure and arrival datetimes for each timetable entry are re-estimated, taking into consideration the departure datetimes of the corresponding travel requests and the required traveling time between bus stops. In each timetable and for each travel request, the ideal departure datetimes from all bus stops (not only the bus stop from where the passenger desires to depart) are estimated. Then, the ideal departure datetimes of the timetable, for each bus stop, correspond to the mean values of the ideal departure datetimes of the travel requests. Finally, starting from the initial bus stop and combining the ideal departure datetimes of each bus stop and the required traveling time between bus stops, included in the response of the Route Generator, the departure and arrival datetimes of the timetable entries are finalized.

9. (Handling of Timetables with Average Waiting Time Above Threshold): For each timetable, the average waiting time of passengers is calculated. If the average waiting time is higher than the provided average waiting time threshold, then the algorithm investigates the opportunity of splitting the timetable into two timetables. So, the travel requests of the initial timetable are partitioned, based on the waiting time of the passenger, and the departure and arrival datetimes of both timetables are re-estimated based on the departure datetimes of the partitioned travel requests. If the number of travel requests of each timetable is higher than the minimum allowed number of passengers in timetable, and the average waiting time of each one of the produced timetables is lower than the corresponding waiting time of the initial timetable, then the new timetables are added to the list of generated timetables, and the initial one is removed. Otherwise, the initial timetable stays as it was before splitting. The whole procedure is repeated until there is no timetable with average waiting time above threshold, or no timetable can be divided.

10. The average waiting time of passengers in all timetables is calculated.

11. Steps 5 to 10 are repeated, as long as the average waiting time of passengers in all timetables decreases.

12. The number of bus vehicles, required in order to serve the generated timetables, is calculated.
13. The generated timetable documents are stored to the corresponding collection of the System Database.

The timetable generation algorithm produces the following **outputs**:

- The generated timetable documents.
- The number of bus vehicles, required in order to serve the generated timetables.
- The average waiting time of passengers in all timetables.

The timetable generation algorithm can be described by the following **pseudocode**:

Timetable Generation Algorithm Pseudocode

```
procedure generate_timetables_for_bus_line(bus_line, timetables_starting_datetime,
                                           timetables_ending_datetime,
                                           travel_requests_min_departure_datetime,
                                           travel_requests_max_departure_datetime):
    bus_stops = get_bus_stops(bus_line)
    travel_requests = get_travel_requests(
        bus_line,
        travel_requests_min_departure_datetime,
        travel_requests_max_departure_datetime
    )
    route_generator_response = get_less_time_consuming_route(bus_stops)
    current_timetables = generate_initial_timetables(route_generator_response)
    average_waiting_time_of_current_timetables = Infinity

    while True:
        new_timetables = generate_new_timetables_based_on_travel_requests(
            current_timetables,
            travel_requests
        )
        average_waiting_time_of_new_timetables = calculate_average_waiting_time(new_timetables)

        if average_waiting_time_of_new_timetables < average_waiting_time_of_current_timetables:
            current_timetables = new_timetables
            average_waiting_time_of_current_timetables = average_waiting_time_of_new_timetables
        else:
            break

    number_of_bus_vehicles = calculate_number_of_bus_vehicles(current_timetables)
    output = {
        current_timetables,
        average_waiting_time_of_current_timetables,
        number_of_bus_vehicles
    }
    return output
end procedure


procedure generate_new_timetables_based_on_travel_requests(current_timetables, travel_requests):
    new_timetables = generate_empty_timetables()
    new_timetables.set(timetable_entries) = current_timetables.get(timetable_entries)
```

```
    correspond_travel_requests_to_timetables(travel_requests, new_timetables)
    calculate_number_of_passengers_of_timetables(new_timetables)
    adjust_departure_and_arrival_datetimes_of_timetables(new_timetables)

    handle_undercrowded_timetables(new_timetables)
    calculate_number_of_passengers_of_timetables(new_timetables)
    adjust_departure_and_arrival_datetimes_of_timetables(new_timetables)

    handle_overcrowded_timetables(new_timetables)
    calculate_number_of_passengers_of_timetables(new_timetables)
    adjust_departure_and_arrival_datetimes_of_timetables(new_timetables)

    handle_timetables_with_average_waiting_time_above_threshold(new_timetables)
    calculate_number_of_passengers_of_timetables(new_timetables)
    adjust_departure_and_arrival_datetimes_of_timetables(new_timetables)

    return new_timetables
end procedure
```
