Optimal Bus Scheduling using a Distributed Game Model Approach

Perla Hajjar, Leïla Kloul, Dominique Barth

Optimal Bus Scheduling using a Distributed Game Model Approach

Perla Hajjar[1], Leïla Kloul[2], and Dominique Barth [2]

***Abstract*— The efficiency of the transport system depends on the planning and control strategies applied. The satisfaction of both the operator and the passenger in transport systems is challenging and determines the level of service of the system. In order to adapt to disruptions in the transport system which influence the traveled time by the buses, the stop skipping control strategy is adopted. The goal is to serve all the passengers waiting at stops by minimizing the total delay in a known static system. Because of the NP-Hardness of minimizing the real total delay of the system, a new delay based on the notion of balancing the load inside the buses, denoted as $load - delay$, is defined. A distributed game model is proposed to solve the delay minimization problem using stop skipping control strategy. Finally, the distributed game is solved by Linear Reward Inaction algorithm (LRI) and its results are compared with the Simulated Annealing meta-heuristic results.**

***Index Terms*— Public transport scheduling, Stop-Skipping, NP-Hardness, Game model, Delay**

I. INTRODUCTION

The public transport system is considered as the backbone of sustainable urban development since it allows more efficient movements in cities. It is the most popular form of public transport as it operates on a fixed route and serves a defined set of stops. Various factors such as dynamic changes in traffic congestion, weather conditions, and unstable demand patterns lead to uncomfortable travel time for both the passengers and the operators [1]. Thus, It is critical that bus services run on time for the convenience of passengers and to be able to provide a dependable public transportation service for them. A delay in the arrival time of a bus at a station may lead to a longer waiting time for passengers and a deterioration of the service.

To provide more flexibility, a variety of innovative transportation services, such as on-demand services, ride-sharing, and autonomous public transportation have recently appeared in urban areas. Moreover, solutions such as the increase of the frequency of the bus lines and bus control strategies, i.e dedicated bus lanes and signals, vehicle holding, stop-skipping and deadheading that have been proposed in the literature are not enough to improve the efficiency and the reliability of the bus systems [2].

*This work was not supported by any organization

[1]Perla Hajjar, is with DAVID Laboratory, Universite de Versailles Saint-Quentin-en-Yvelines/ Paris Saclay, Versailles, France in collaboration with Communauté d'Agglomération de Saint Quentin en Yvelines, Trappes, France perla.hajjar@uvsq.fr amd perla.hajjar@sqy.fr

[2]Dominique Barth is with Universite de Versailles Saint-Quentin-en-Yvelines/ Paris Saclay, Versailles, France dominique.barth@@uvsq.fr

[2]Leïla Kloul is with Universite de Versailles Saint-Quentin-en-Yvelines/ Paris Saclay, Versailles, France leila.kloul@uvsq.fr

In today's situation, buses of the same line stop at all stations forming a schedule of served stations. The stop-skipping (also known as expressing, or limited-stop service) is a control measure that allows a vehicle to skip a stop (or a series of stops) of the same line if it is running behind schedule [3] [13]. To provide a resilient and a dynamic service, we adopt the stop skipping strategy in order to minimize the time until the last passenger reaches his/her destination, which is the $delay$. Considering that we are in a static system where the demand, the number of buses, the number of stops, and the time to move on the route are known and fixed, our purpose is to decide at the beginning of each turn which stations to be served by each bus that lead to the minimization of the delay. In this context, we propose a distributed game model to solve the delay minimization problem using the stop skipping control strategy in the static system. The Linear Reward Inaction (LRI), a reinforcement learning algorithm, is adopted to solve the distributed game model. Then we analyse and compare the system performance and results in an offline context with complete information optimisation meta-heuristics: Simulated annealing and Descent Algorithm.

The remainder of this paper is organized as follows. Section 2 contains reviews of available control strategies. In Section 3, defines the bus transport system model with the stop skipping strategy. Section 4 describes the delay minimization problem, shows its NP-Hardness and defines the *load-delay* notion. In Section 5 we model our problem as a distributed game and propose a reinforcement learning approach, specifically Linear reward Inaction, to solve the minimization problem in a static environment. Section 6 verifies the correlation between the real delay and the new defined *load-delay* parameter and provides the performance evaluation of the descent algorithm, Simulated Annealing, and reinforcement learning algorithms.

II. RELATED WORK

There is a handful of works that have studied and proposed optimization techniques for the stop skipping problem. The stop skipping decreases the service time of buses by allowing them to skip one or more stations either entirely or after allowing alighting only.

Stop skipping is formulated as an optimization problem and is often proposed at the planning level. In the study of [4], a new stop-skipping strategy is proposed, where it is applied to alternate buses and is fixed once the bus departs from the starting terminal. The objective is to minimize the total cost of passenger waiting time, in-vehicle time and vehicle travel time. They used real data in SimTransit simulation model. The analysis showed that stop-skipping

control is effective in reducing the passenger in-vehicle time, waiting time, and operational vehicle trip time.

To address the problem of determining the skipped stops of multiple trips in a rolling horizon, the author in [5] suggested dividing the day into discrete time windows. The decision is taken at the start of the rolling horizon and cannot be modified. The main objective is to minimize the weighted cost of passenger waiting time, passenger in-vehicle time, and vehicle travel time. The author tested three solution methods: Brute force, sequential hill climbing, and genetic algorithm. Real data were used with 5 stops as stop-skipping candidates to be able to determine a stop-skipping solution in near real-time. Results have showed that the stop-skipping control in rolling horizon is beneficial in scenarios with mild travel time variations.

The authors in [2] identified the deadheading strategy as a part of the stop skipping strategy. They aim to minimize the sum of the total travel time, the waiting time of all the bus passengers, and the total bus travel time. They assumed that prescribed stops cannot be changed after dispatching and that if a bus is allowed to skip stations, the following bus should serve all the stations. The authors adopted the Genetic Algorithm as a solution method and the Monte Carlo Simulation to estimate the objective function. They verified the proposed mathematical model using abstracted data from real bus line in Suzhou city of China.

A new modification for the stop skipping control strategy was introduced by [11] as they considered that the bus can still drop off passengers at stops in the skipping segment since the decision is in real time. This approach is proposed to respond to disruptions in real time. They compared this approach with the original stop skipping control by applying an exhaustive search method on a small scale network.

The advantage of fixed stop skipping control is that it can be communicated to bus drivers and passengers in advance. The authors in [4], [2], and [6] assumed of that if one bus skips any stop(s), then the next bus of the same line should serve all the intermediate stops along the bus line. However, next bus should also be allowed to skip stops as long as the service is maintained and the demand is served. In addition to that, the determination of the skipped stop for each trip is done in isolation to reduce the computation complexity [5].

Most of these previous studies do not take into consideration that the stop skipping affects the total trip time of other buses and focus only on minimizing the waiting time considering they have previous knowledge of this information. These models will be hard to implement in real life scenarios to adapt to dynamic changes.

In this paper, we adopt the stop skipping control to decide at the beginning of each turn which stations to be served by each bus that lead to the minimization of the delay. We consider that our system is static and thus we have previous knowledge of all the needed parameters for this minimization. A reinforcement learning approach is tested to solve the proposed distributed game model of the delay minimization problem.

III. MODELING THE TRANSPORTATION SYSTEM

A first theoretical simplified model of the public transport corridor is represented as a ring R made of N consecutive cyclic slots numbered from 0 to $N-1$ that represent discrete events as shown in Fig 1. Each slot is the possible position of a bus moving and can be occupied by only one bus. The public transport system is defined as follows:

- N consecutive cyclic slots on ring R.
- A set of K stations s_1,...s_K, with $2 \leq K < N$. Each station s_i is associated with a slot (with at most one station per slot).
- A set of B buses having the same capacity Cap such that $B < N$. Each bus $b_j \in B$ has an initial starting position $init_j$. At time step t, the position of bus b_j is denoted by Pos_j^t. Each bus b_j has a serving vector D_j that indicates the stations served: $D_j[i] = 1$ means that bus b_j will stop at station s_i, 0 otherwise.

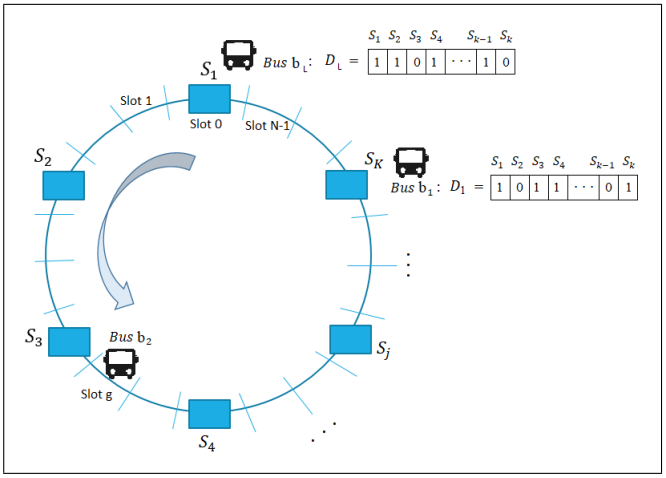

Fig. 1. Model architecture of the transport system

A. Bus Movement Definition

At every time step t, each bus $b_j \in B$ is in state $State_j^t \in \{MOV, STP\}$.
The following state transitions are allowed:

- If at time step $t-1$, the state of bus b_j is $State_j^{t-1} = STP$ then, at time step t, $State_j^t = MOV$. This indicates that the bus can only stop for 1 time step at a station for passengers to alight and board.
- If at time step $t-1$, the state of bus b_j is $State_j^{t-1} = MOV$ then $State_j^t = STP$ if and only if b_j is at a station s_i and $D_j[_i] = 1$. This means that the bus's state can change from state MOV to STP only if the slot at the next time step is a station where the bus is set to stop.

If bus b_j at time $t-1$ is in position Pos_j^{t-1}, then:

- $Pos_j^t = Pos_j^{t-1}$ if and only if $State_j^{t-1} = STP$.
- $Pos_j^t = Pos_j^{t-1} + 1$ if and only if $State_j^{t-1} = MOV$ and :

– $\nexists$ bus $b_{j'}$ such that $Pos^t_{j'} = Pos^{t-1}_{j'} + 1$ and $State^t_{j'} = MOV$.
– $\exists$ bus $b_{j'}$ such that $Pos^t_{j'}$ is at station s_i and $D_{j'}[i] = 1$ and $D_j[i] = 0$.

In all other cases $Pos^t_j = Pos^{t-1}_j$.

Note that with $B < N$, the system is deterministic and there is always at least one bus for which the decision of the next position does not depend on the other buses.

B. Assumptions

The proposed model is based on the following assumptions: (1) the origin-destination demand matrix $M[O, D]$ is given at the beginning, specifying the number of passengers waiting at each station at time step $t = 0$ and having a specific destination. Note that this demand matrix is given at the beginning of the simulation (time step $t = 0$ and does not change. (2) Passengers board the first arriving bus serving their destination without interconnections while respecting the bus capacity constraint. (3) Passengers waiting at a station with different destinations board the stopping bus based on a uniform distribution. (4) The buses stop for one time step only for passengers to board and alight. If a bus is stopped at a station and a following bus wants to stop at the same station, the following bus will be blocked on the ring. (5) For each bus, the starting and ending stations of the route can be any station on the ring. (6) The route is a one direction route with no passengers alighting at first station (s_1) and no passengers boarding at last station (s_K).

IV. PROBLEM DEFINITION

The solution we want to obtain is a schedule that defines the stopping patterns of buses in stations with a minimum delay. A $schedule$ is defined as a set of serving stops vectors $Sch = \{D_1, \ldots, D_B\}$. Each D_j corresponds to one bus b_j such that for any $1 \leq i < K$, $D_j[i] = 1$ iff b_j stops at station s_i, else $D_j[i] = 0$. The delay of a schedule is the maximum time needed until the last passenger reaches his/her destination.

A schedule is $feasible$, **if and only if**, for all couples of stations s_x, s_y such that $x, y \in \{1...K\}$, $M[x, y] > 0$ and $s_x \neq s_y$, there exists at least one bus b_j such that $D_j[x] = D_j[y] = 1$. We denote by Stp_j the sum of elements of vector D_j, i.e., the number of stations that b_j serves.

Given the number of buses, the number of stations, and the static demand origin-destination matrix, the goal is to find a schedule that serves all passengers waiting at stops with minimum expected $delay$. Calculating the real delay value is time consuming considering the large number of constraints in our model to be evaluated at every time step. This minimization problem is NP-Hard even if $|B|$=2, and we consider a polynomial transformation from the Set Partition Problem (SPP). Thus, we define a new notion which is the $load - delay$ and prove that the problem of minimizing the $load - delay$ gives a rise to a problem in NP for any schedule of any instance of this problem.

"The more balanced and full the buses are, the smaller the delay". Based on this proposition, we define the bus $load - delay$ as follows. For any instance (R, K, B, M) with a feasible schedule, we calculate first the bus load at every station as defined in Definition 1. For each bus b_j at each station s_i, we define the load of b_j at s_i as $load_j(i)$. If $D_j[i] = 1$, then $load_j(i)$ is the sum of the total number of waiting passengers that can get into bus b_j based on the serving vector $D_j[i]$ and the passengers already on-board, divided by the number of buses serving every origin destination station denoted as $deg(x, y)$. If $D_j[i] = 0$, then $load_j(i) = load_j(i - 1)$.

Definition 1: For any station $s_i \in \{1...K\}$ and any bus $b_j \in \{1...B\}$, we define the **load** of b_j on s_i by :

$$load_j(i) = \sum_{\substack{x \neq y \ crossing \ s_i \\ s.t. \ D_j[x] = D_j[y] = 1}} \frac{M[x, y]}{deg(x, y)} \tag{1}$$

Then, we compute the maximum load of bus b_j along all the stations s_i for $1 \leq i \leq K$, denoted by $MAXload_j$, as defined in Definition 2.

Definition 2: The maximum load of bus b_j is defined as:

$$MAXload_j = \max_{1 \leq i \leq K} b - load_j(i) \tag{2}$$

Finally, the $load - delay$ of a schedule $LD(Sch)$ is used to indicate how many time steps are needed for all the passengers to be served and reach their destination. Hence, the bus that has the highest load requires the largest time to serve all the demand. The $load - delay$ of a schedule, $LD(Sch)$, is calculated as in Definition 3. The expression $\lceil \frac{MAXload_j}{Cap} \rceil$ represents the number of turns needed by bus b_j to serve the passenger load that it will carry. Since any bus b_j needs one time step to move between slots and stops also for one time step at each station for passengers to board and alight, then the expression $(N + Stp_j)$ is used to express the delay of the schedule in time steps.

Definition 3: The ***load-delay*** of Sch is

$$LD(Sch) = \max_{b_j \in B} \left(\left\lceil \frac{MAXload_j}{Cap} \right\rceil \times (N + Stp_j) \right) \tag{3}$$

V. MODELING THE PROBLEM AS A DISTRIBUTED GAME

A. Game model definition

This game model is the one on which the reinforcement learning approach that we propose is based. We suppose that the stations are the players and the choices of bus stops at each station are the strategies (i.e., 2^B strategies for each of the K player).

First, given an instance (R, K, B, M) with a feasible schedule Sch, for any station s_i, we define a local load parameter related to the maximum load of a bus stopping in s_i. The station local load denoted by SJN in (4) for each station s_i corresponds to the maximum load of bus b_j, such that $D_j[i] = 1$ multiplied by the number of slots and stations

to stop at which represents the time units needed to serve the demand:

$$SJN = \max_{j \ s.t. \ \ D_j[i]=1} \left(\left\lceil \frac{\sum_{\substack{x \neq y \ \ crossing \ \ s_i \\ s.t. \ \ D_j[x]=D_j[y]=1}} \frac{M[x,y]}{deg(x,y)}}{Cap} \right\rceil \times (N + Stp_j) \right) \quad (4)$$

We consider also an upper bound, denoted as WST, of the load-delay of any schedule as defined in (5). This upper bound is considered as if we have only one bus serving the stops.

$$WST = \left\lceil \frac{\max_{s_i \in \mathcal{L}} \left(\sum_{x \neq y \ \ crossing \ \ s_i} M[x,y] \right)}{cap} \right\rceil \times (N+K+1) \quad (5)$$

We define a game called St-Load-Game as follows.

- The set of players is K.
- The action set A_i of each player $s_i \in \{1...K\}$ is all possible combinations of stopping patterns of the buses at this station, that is $|A_i| = 2^B$.
- A strategy profile $\pi = a_1, \ldots, a_B$ implies a unique schedule Sch_π, that is $D_j[i] = 1$ if $b_j \in a_i$ and reciprocally.

The purpose of each player is to minimize its cost. The cost C_i of a player s_i for the strategy profile π is defined as: $C(i) = \gamma \times TOT - \alpha \times Sol_i$, where:

- $TOT = WST$ if Sch_π is not feasible, else $TOT = LD(Sch_\pi)$,
- $Sol_i = 0$ if there exists i' such that $M[i,i'] > 0$ and $deg(i,i') = 0$, else $Sol_i = SJN_i$,

and $\gamma \geq 1$ and $\alpha \geq 0$ are two tuning parameters of the game indicating the weight of the global and local loads.

B. Distributed reinforcement learning approach

To solve the game defined above, we consider a distributed reinforcement learning approach based on the Linear Reward-Inaction (LRI) algorithm [8]. LRI is a reinforcement learning method based on a reward system where players aim to minimize or maximize a common cost. Each player has a stochastic vector of actions called strategy vector. This vector represents the possibilities of actions that a player s_i has. Every action $a_i \in A_i$ has an initial probability value to be selected $q_{i,a}$. The players learn at the same time and try actions in order to achieve their objective. A solution is said to be good given a strategy profile π, if it improves the utility. At each iteration, each player randomly chooses an action among its own strategy vector. For each player, we calculate its utility which is based on the action chosen. At the end of each iteration, each player updates its strategy vector following the update rule of LRI presented in (6).

$$\begin{cases} q_{i,a}^{t+1} = q_{i,a}^t + b * U_i^t * (1 - q_{i,a}^t) & \text{If } a = a_i^t \\ q_{i,a'}^{t+1} = q_{i,a'}^t - \left(\frac{q_{i,a'}^t}{1 - q_{i,a}^{t+1}} \times b * U_i^t * (1 - q_{i,a}^t) \right) & \forall \ a' \neq a_i^t \end{cases} \quad (6)$$

b: learning parameter with $0 < b < 1$.
$q_{i,a}^t$: the probability that player i plays action a at iteration t.
$q_{i,a'}^t$: the probability that player i plays action a' for $a' \neq a$ at iteration t.
U^t: utility function.

Consider the time step t of the learning process (i.e., round t of the simultaneous game in the learning), and let $a_i^t \in A_i$ be the action played by s_i and $C^t(i)$ be the cost of player s_i at this step. Then the utility function to be considered for s_i at this step is

$$U_i^t = \frac{C_{a_i}^{max}(i) - C^t(i)}{C_{a_i}^{max}(i) - C_{a_i}^{min}(i)} \quad (7)$$

with $C_{a_i}^{max}(i)$ (resp. $C_{a_i}^{min}(i)$) being the maximum (resp. minimum) cost impacted to s_i when choosing action a_i in a step between 1 and t. The design of the utility function is critical for the player's learning of optimal stop skipping pattern. It should be both broad enough to capture the impact of the chosen action, but specific enough to not cause noise during learning.

VI. PERFORMANCE EVALUATION

In this section, we experimentally evaluate the correlation between the delay and the load-delay for all schedules of a same problem instance. We will also compare the results of the LRI algorithm to the results of two meta-heuristic algorithms: Simulated Annealing (SA) and Descent Algorithm.

A. Correlation between real delay and load-delay:

We consider here an instance (R, K, B, M) obtained from real data measured on a bus line of the urban community of Saint-Quentin-en-Yvelines, France (a Paris suburban area). The data were collected in 2011. The bus line 414 was divided into 8 sectors and for each sector the in going and out going demand of the buses were analyzed. Based on that, we consider each sector as a station. Hence, our study is based on 8 stations served by 3 buses of identical capacity of 22 passengers each. The demand matrix $M[O, D]$ is provided in Table I representing the demand for every origin destination station at time step $t = 0$.

For each possible feasible schedule Sch for this instance, we measure by simulation using SUMO simulator [12] the real delay $Delay(Sch)$ and we calculate its corresponding $load - delay$ $LD(Sch)$ as defined in Definition 3. Figure 2 plots the values of the real deal and the $load - delay$ for all of the obtained feasible schedules Sch, sorted in increasing order of $LD(Sch)$. The $x - axis$ represents the schedule number and the $y - axis$ represents its corresponding delay in time steps. The blue curve (upper dense curve) represents the

TABLE I

ORIGIN-DESTINATION(O/D) DEMAND MATRIX OF LINE 414 IN SAINT-QUENTIN-EN-YVELINES IN ONE DIRECTION

O/D	1	2	3	4	5	6	7	8
1	0	77	34	6	14	3	3	2
2	0	0	55	43	89	17	31	6
3	0	0	0	22	53	20	57	7
4	0	0	0	0	5	4	8	2
5	0	0	0	0	0	6	20	8
6	0	0	0	0	0	0	43	9
7	0	0	0	0	0	0	0	40
8	0	0	0	0	0	0	0	0

real delay values obtained using SUMO and the orange curve (linear line) represents their corresponding $load - delay$ values. These curves show a real correlation between these two measurements. Although the real delay calculated by SUMO increases slightly in a quicker manner than the load-delay, we still consider that there exists a correlation between these two values. Thus we are going to focus the algorithmic approaches on the optimization of $LD(Sch)$, a problem which is in NP.

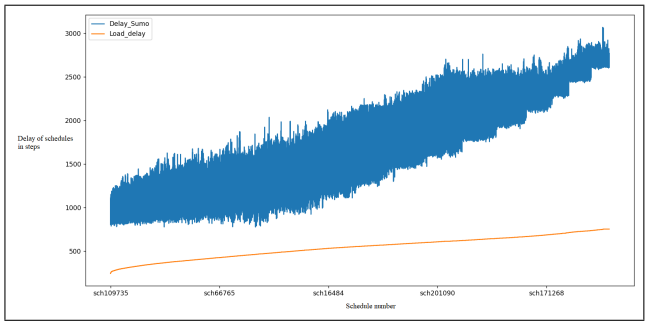

Fig. 2. Correlation between the real delay obtained by SUMO and the defined load-delay $LD(Sch)$ sorted by $LD(Sch)$

B. Solution Methods:

In this section, we compare the performance of the LRI approach with two neighborhood meta-heuristics: (1) Descent search algorithm and (2) Simulated Annealing in a static environment. For the meta-heuristics, we use the same neighborhood definition between two schedules defined as follows. A schedule Sch is considered as the concatenation of the schedule binary vectors V_{Sch} of all buses $b_1, b_2, \ldots, b_B$, i.e., a binary vector of size $B \times L$. A neighbor of a schedule Sch is a feasible schedule Sch' such that $V_{Sch'} = V_{Sch} \oplus \mu$, where μ is a binary vector of size $B \times L$ and with Hamming weight w. It determines the number of stops we are allowed to alter their values.

1) Simulated Annealing: Simulated Annealing (SA) Algorithm is a probabilistic method proposed by [9]. SA can be used to estimate the global minimum for a function with many variables. SA can produce a good local though not necessarily global optimal solution within a reasonable computing time. Essentially speaking, simulated annealing can be seen as a "randomized variation" of the local search method [10].

2) Descent Algorithm: An approach inspired by the Gradient descent algorithm is adopted. For this algorithm, the Hamming weight w is set to one. Starting from an initial schedule Sch_{init} that serves all stations by all buses, we generate all possible neighbor schedules using w based on the definition above. Among the neighbor schedules, we select the schedule that has the minimum delay $LD(Sch_{neighs})$. We continue this process until there is no more feasible neighbors. This method has showed that if we start from a schedule that serves all stops, it does not find a better schedule in terms of $load - delay$. For that, we investigate the nature of the data. Almost 15% of the feasible schedules are local minimums. Hence the descent algorithm will not converge as expected starting from an initial schedules Sch_{init} that serves all stations, knowing that this schedule is a local minimum.

Since the SA algorithm is guaranteed to converge to at least a good local minimum, it is the one we consider to compare the LRI algorithm with in the following.

C. Comparison between SA and LRI:

After setting the initial temperature $T_{initial}$ to 150000 iterations and the cooling rate to 0.85 for the SA algorithm, we test the algorithm with Hamming weight $w = 1$ and $w = 2$. For each value, we run the algorithm 30 times. With an average number of iterations 35000, the SA meta-heuristic shows that it can converge to a schedule near the optimal one in both cases. It also shows that the Hamming weight does not affect the speed of convergence, since in the two cases, the algorithm starts to converge between the 30000 and the 35000 iteration. For Hamming weight $w = 1$, the mode schedule that the algorithm stabilizes on is a schedule that serves the demand without each bus stopping at every station as shown in Table II with an average delay of 247 time steps, while with hamming weight $w = 2$ (see Table III), the mode schedule has an average delay of 260 time steps and not every bus stops at all stations.

TABLE II

THE RESULTING SCHEDULE OF THE SA ALGORITHM WITH w=1

Bus/Stop	1	2	3	4	5	6	7	8
B1	0	1	1	1	0	1	1	1
B2	1	1	0	0	1	1	1	0
B3	1	0	1	1	1	1	1	1

TABLE III

THE RESULTING SCHEDULE OF THE SA ALGORITHM WITH w=2

Bus/Stop	1	2	3	4	5	6	7	8
B1	1	1	1	0	1	1	1	1
B2	1	1	1	1	1	0	1	0
B3	1	1	1	1	1	1	0	1

For the LRI algorithm, we are interested in testing the impact of γ and α in the player cost C_i on the schedule

convergence of the LRI algorithm. For this reason, we test: (1) $\gamma = 1$ and $\alpha = 0$ and (2) $\gamma = 2$ and $\alpha = 1$. Clearly, from the expression of C_i that with $\gamma = 1$ and $\alpha = 0$, the local load on the station is neglected and only the global $load - delay$ of the schedule is taken into consideration. The algorithm in this case converges to a schedule that serves all the stations with a delay of 260 time steps.

For $\gamma = 2$ and $\alpha = 1$, we run the algorithm several times and each run for three hours producing around 6 million iterations. The LRI algorithm converges and stabilize on a schedule that serves the demand without stopping at all stations and with 268 time steps as a delay. We note that the players (stations) have learned their best moves and stabilize to an action that gives them the best benefit as shown in Table IV.

TABLE IV

THE RESULTING SCHEDULE OF THE LRI ALGORITHM WITH γ=2, α=1

Bus/Stop	1	2	3	4	5	6	7	8
B1	1	1	1	1	1	1	0	1
B2	0	1	1	1	1	1	1	0
B3	1	1	1	1	1	0	1	1

Figure 3 represents the probability evolution of the best chosen action per player and how it affects the converge of the $load - delay$ value. The $x - axis$ is the number of iterations (in millions) and the $y - axis$ is the probability of the actions in the upper part of the figure and the load delay of the schedule in the second part of the figure. We notice that most of the stations learn their best action when the algorithm stabilizes on the schedule presented in Table IV after 400 thousand iterations. Some players try to choose different actions at some point, but as seen in the lower graph of Fig 3, this does not affect the convergence and stabilization of the algorithm.

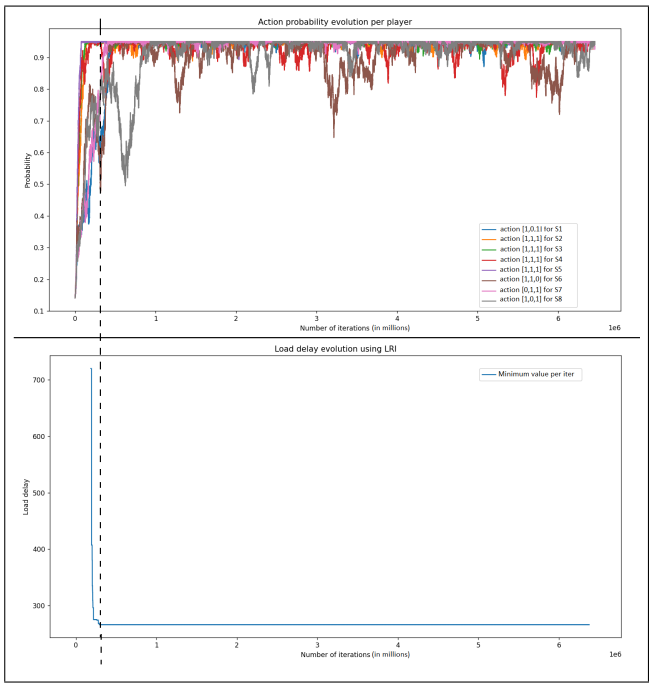

Fig. 3. Probability evolution of the best action per player

VII. CONCLUSIONS

In this paper, we define a $load - delay$ notion based on the concept of balancing the load inside the buses. We propose modeling our problem as a distributed game in the context of a static environment and we consider a distributed reinforcement learning approach based on Linear Reward Inaction algorithm to implement it. We validate the correlation between the proposed $load - delay$ and real delay of every instance using real data. Results have showed that the LRI performs well and converges to a near optimal schedule in a well known static system. The SA algorithm has showed that it converges to a near optimal schedule with minimizing the load-delay value better than LRI. In the proposed static test case scenario, both the Simulated Annealing and the Linear Reward Inaction algorithms show that they are able to converge to a schedule that minimizes the $load - delay$. Since the presented approaches results in a near optimal schedule in a static environment, we plan in the future to dynamically adjust the schedules of the buses to adapt to real road conditions and passengers demands.

REFERENCES

[1] H. Kei and A. Takehiro and K. Nobuo, Simulation for Passengers Convenience Using Actual Bus Traffic Data (Book) , July, 2019 , pp. 175-194.

[2] Z. Liu , Y. Yan , X. Qu and Y. Zhang , Bus stop-skipping scheme with random travel time, Journal of Transportation Research Part C: Emerging Technologies, vol. 35, pp. 46-56, 2013.

[3] H. Larrain and J. C. Muñoz, When and where are limited-stop bus services justified?, Journal of Transportmetrica A Transport Science, vol. 12, no. 9, pp. 811-831, 2016.

[4] L.Fu, Q. Liu and P. H. Calamai, Real-Time Optimization Model for Dynamic Scheduling of Transit Operations, Journal of Transportation Research Record, vol. 1857, pp. 48 - 55, 2003.

[5] K. Gkiotsalitis, Stop-skipping in Rolling Horizons, Journal of Transportmetrica: A Transport Service, 2020.

[6] N. Khoat and D. Bernard, THE REAL-TIME STOP-SKIPPING IN THE URBAN TRANSPORTATION NETWORKS, Journal of IFAC Proceedings Volumes, vol 40, no. 18, pp. 637-642, 2007.

[7] O.J. Ibarra-Rojas and F. Delgado and R. Giesen and J.C. Muñoz, Planning, operation, and control of bus transport systems: A literature review, Journal of Transportation Research Part B: Methodological, vol. 77, pp. 38-75, 2015.

[8] P.S. Sastry, V.V. Phansalkar, M.A.L. Thathachar, Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information, Journal of IEEE Transactions on Systems, Man, and Cybernetics, vol. 24, pp. 769-777, 1994.

[9] S. Kirkpatrick , C. D. Gelatt , and M. P. Vecchi , Optimization by Simulated Annealing, Journal of Science, vol. 220 , pp. 671- 680, 1983.

[10] W. FanRandy and B. Machemehl , Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem, Journal of Transportation Engineering , vol. 132, no. 2 ,February 1, 2006.

[11] A. Sun and M. Hickman, The Real–Time Stop–Skipping Problem, Journal of Intelligent Transportation Systems: Technology, Planning, and Operations, vol. 9, 26 Jan 2007.

[12] Behrisch, Michael and Bieker-Walz, Laura and Erdmann, Jakob and Krajzewicz, Daniel. (2011). SUMO – Simulation of Urban MObility: An Overview. Proceedings of SIMUL. 2011.

[13] K. Gkiotsalitis, O. Cats, At-stop control measures in public transport: Literature review and research agenda, Transportation Research Part E: Logistics and Transportation Review, Volume 145, 2021