Learning Automata-Based Solutions to the Single Elevator Problem

O. Ghaleb, B. John Oommen

Learning Automata-based Solutions to the Single Elevator Problem

O. Ghaleb[1] and B. John Oommen[2] *

School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6.
[1]omar.ghaleb@carleton.ca,[2]oommen@scs.carleton.ca

Abstract. The field of AI has been a topic of interest for the better part of a century, where the goal is to have computers mimic human behaviour. Researchers have incorporated AI in different problem domains, such as autonomous driving, game playing, diagnosis and security. This paper concentrates on a subfield of AI, i.e., the field of Learning Automata (LA), and to use its tools to tackle a problem that has not been tackled before using AI, namely the problem of the optimally scheduling and parking of elevators. In particular, we are concerned with determining the Elevators' optimal "parking" location. In this paper, we specifically work with the Single[1] Elevator Problem (SEP), and show how it can be extended to the solution to Elevator-like Problems (ELPs), which are a family of problems with similar characteristics. Here, the objective is to find the optimal parking floors for the single elevator scenario so as to minimize the passengers' Average Waiting Time (AWT). Apart from proposing benchmark solutions, we have provided two different novel LA-based solutions for the single-elevator scenario. The first solution is based on the well-known L_{RI} scheme, and the second solution incorporates the *Pursuit* concept to improve the performance and the convergence speed of the former, leading to the PL_{RI} scheme. The simulation results presented demonstrate that our solutions performed much better than those used in modern-day elevators, and provided results that are near-optimal, yielding a performance increase of up to 80%.

Keywords : *Learning Automata (LA), Learning Systems, Single Elevator Problem (SEP), Elevator-like Problem (ELP), Parking Problem*

1 Introduction

Right from its infancy, the goal of the field of Artificial Intelligence (AI) has been to make computers respond intelligently in everyday and challenging situations. Turing, in his Turing test, suggested that a machine could be considered to possess AI if a human observer would not have been able to distinguish its behavior from the behavior of a real human being. This lofty goal has been achieved to a phenomenal degree. AI-based computer programs have challenged and beaten the best players in many games, including Chess and Go.

* *Chancellor's Professor* ; *Life Fellow: IEEE* and *Fellow: IAPR.* This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway.

[1] We consider the more complicated multi-elevator problem in a forthcoming paper.

This paper considers a field in which AI has not been applied much. Consider a tennis player who is moving within his side of the court. After he hits the ball, the question that he encounters is to know where he should place himself so as to best counter his opponent's response. This can be modelled as a parking problem, i.e., where should the player "park" himself so that when his opponent hits the ball, it is in the vicinity of where he has parked. We could refer to problems of this type as "Elevator-like" Problems (ELPs). They are, in fact, in a unidimensional domain, related to the problem of where an elevator within a building should be parked. Thus, if there is a building with n floors and if the elevator car is requested at floor i, where should the car move to after the passenger is dropped off? While this depends on the criterion function, it is expedient to work with the understanding that the car should be parked near to the next passenger call. We can extend the problem to consider where emergency vehicles should be parked so as to be readily available for the next call.

Although the above problems are, in general, transportation problems, their common facet can also be extended to other domains. For example, one could consider the problem of where the read/write disk head in a memory bank should be placed so that it can access data more expeditiously. Indeed, our problem model can be extended to consider improving underwater communication systems by determining where the underwater sensors should be located.

In this paper, we refer to all of these problems as ELPs, and this is the primary focus of the research. However, to render the problem to be non-trivial, we assume that "the world" in which we are operating is stochastic and that the underlying distributions are unknown. For example, in the case of the tennis player, we assume that there is a distribution for the place where the opponent will place his counter-shot, but that this distribution is unknown.

We will discuss the Single Elevator Problem (SEP) (also please see [1] and [7]) and explain the solutions by which researchers have tackled it in the past. The SEP is a subclass of the Multi-Elevator Problem (MEP) in which we have a building with n floors and a single elevator, and where we are to design a policy for the elevator in order to operate in such a way so as to save energy, reduce the travel time or the waiting time for passengers. Moreover, the elevator policy should decide how the elevator should operate in order to achieve its goal. For example, one possible policy could require that the elevator picks the best route in order to pick passengers, while another could be to serve the longest queue first or to decide where the elevator should wait for the next call. As we shall see, the literature does not record several solutions for the SEP. The reason for this is that most papers deal with the more complex scenario, the MEP, which is a generalization of the SEP.

1.1 Prior Art for the SEP

Two notable contributions[2] for the SEP were by Tanaka *et al.* in [14, 15]. In their papers, they studied the operation problem of a single-car elevator with so-called

[2] The review presented here is necessarily brief. A more detailed survey of the field is included in [2] and [3].

"destination hall call registration", where the system registers the passengers' destination floors *before* boarding the elevator. This is opposed to regular or "traditional" elevators, where passengers can only pick the direction of their trip before boarding, and thereafter, specify the destination floor *after* boarding.

In the first part of their study [14], the authors tried to answer two questions, namely: (a)"Can (single) car operations be improved with destination hall call registration?", and (b)"How can it be realized?". They were able to formulate the operation problem as a Dynamic Optimization Problem (DOP) such that an objective function, which is the weighted average waiting time for passengers, is minimized. They showed that the DOP substantially improved the system's capability when compared to the conventional "selective collective" operation.

The authors of the second part [15] introduced a branch-and-bound algorithm to solve the formulation of the DOP problem [14]. Here, the lower bound calculations for the subproblems generated in the course of the branch-and-bound algorithm, came from decomposing the problem into three subproblems, i.e., the passenger loading/unloading, car stops, and lastly the floor-to-floor travel. They then applied the Lagrangian relaxation method to solve the overall problem. Unfortunately, their algorithm was not fast enough for real-life situations.

After the studies in [14], [15], the authors [16] examined how one could improve the efficiency of a single elevator system with "destination hall call" by considering dynamically optimized objective functions. Here, they applied simulated annealing, and from their results they showed that the weighted average of two different objective functions, i.e., the weighted average service time and the maximum, yielded a better performance than by using only a single objective function. They also observed was that one had to choose the weights of the objective functions carefully because this significantly affected the results of the experiments. Moreover, these weights depended on the elevator's characteristics.

The authors of [13] addressed the goal of estimating the optimal values for the upper and lower bounds for the elevator scheduling problem, in which they assumed the availability of all the information about the passengers. To achieve this, they formulated the problem in two parts, a high level and a low level component. The high level component was a passenger-to-car assignment, and the low level component was the passenger-to-trip assignment. This then was used for the formulation of the low-level component. The authors obtained the upper bound, by finding a reasonable solution to the problem. They thus obtained the lower bound, by defining a lower bound for a newly-constructed problem using the Lagrangian Relaxation method. Their results were efficient and scalable.

Another work, presented by Xu and Feng [22], modelled the single elevator scheduling problem as a Mixed Integer Linear Program (MILP). They focused on using this model to improve the service time for passengers. Based on a prototype model obtained from the industry for the dynamics of a moving elevator, they linearized the nonlinear travel activities for the problem. They tested their "real-time" model under different circumstances and scenarios, and conjectured that their model could be extended and used as a benchmark.

1.2 Learning Automata (LA)

We now briefly review[3] the field that we shall work in, namely, that of LA. The concept of LA was first introduced in the early 1960's in the pioneering research done by Tsetlin [20]. He proposed a computational learning scheme that can be used to learn from a random (or stochastic) *Environment* which offers a set of actions for the automaton to choose from. The automaton's goal is to pick the best action that maximizes the average *reward* received from the *Environment* and minimizes the average *penalty*. The evaluation is based on a function that permits the Environment to stochastically measure how good an action is, and to, thereafter, send an appropriate feedback signal to the automata.

After the introduction of LA, different structures of LA, such as the deterministic and the stochastic schemes, were introduced and studied by the famous researchers Tsetlin, Krinsky and Krylov in [20] and Varshavskii in [21].

The field of LA, like many of the Reinforcement Learning techniques, has been used in a variety of problems, mainly optimization problems, and in many fields of AI. Among other applications, it has been used in neural network adaptation [6], solving communication and networking problems [9], [11], in distributed scheduling problems and in the training of Hidden Markov Models [5].

In Figure 1, we have the general stochastic learning model associated with LA. The components of the model are the *Random Environment*, the *Automaton*, the *Feedback* received from the environment and the *Actions* chosen by the automaton.

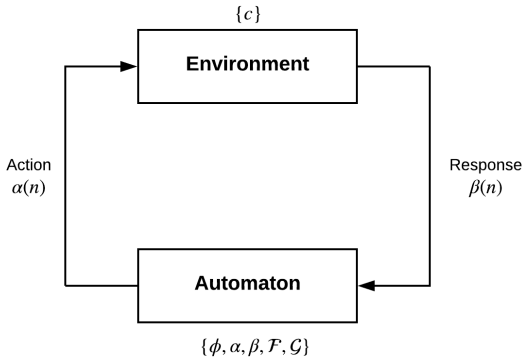

Fig. 1: The Automaton-Environment Feedback Loop.

The Environment: We first define the stochastic Random Environment that the automaton interacts with. Initially, the automaton picks an action from the set of actions available to it and communicates it to the environment. The Environment evaluates that action according to a random function, and sends back a *reward* or a *penalty* feedback signal to the automaton.

[3] We will not go through irrelevant details and/or the proofs of the LA-related claims. They can be found in [2] and [3], and the reference book by the pioneers [8].

The Automaton: The LA achieves its learning process as an iterative operation that is based on the Environment and the interaction between them. This process consists of two main steps, the first of which is *policy evaluation*, which is how the Environment evaluates the selected action. The second step involves *policy improvement*, where the LA improves the probability of selecting an action that would maximize the *reward* received from the Environment.
Estimator and Pursuit Algorithms: The concept of Estimator Algorithms was introduced by Thathachar and Sastry in [17, 18] when they realized that the family of Absolutely Expedient algorithms would be absorbing, and that they possessed a small probability of not converging to the best action. Estimator algorithms were initially based on Maximum Likelihood (ML) estimates (and later on Bayesian Estimates), where they also used the estimates of the reward probabilities to update the actions' probabilities. By doing this, the LA converged faster to the actions that possessed the higher reward estimates. The original Pursuit Algorithms are the simplest versions of those using the Estimator paradigm introduced by Thathachar and Sastry in [19] and [17]. These algorithms are based on the pursuit strategy, where the idea is to have the algorithm *pursuing* the best-known action based on the corresponding reward estimates. The pursuit algorithms were proven to be ϵ-optimal. The concept of designing ϵ-optimal discretized Pursuit LA was introduced by Oommen and Lanctot in [10]. The discretized Pursuit LA converged faster than their continuous counterparts.

1.3 Contributions of this Paper

The novel contributions of this paper are:

- We have surveyed a subfield of AI, i.e., the field of Learning Automata (LA), and have concluded that it has not been used previously to solve the SEP.
- We were able to identify two different models of computations for the elevator problem, where the first requires the calling distribution to be known *a priori*, and the second does not need such information.
- We were able to model the elevator problem in such a way that it can be solved using LA approaches.
- We introduced a Linear Reward-Inaction (L_{RI})-based solution to tackle the single elevator problem. It has been referred to as SEP3.
- We also presented an improvement on SEP3 by including the so-called Pursuit phenomenon into the LA solution. This led to the PL_{RI}-based solution, referred to as SEP4, and this yielded better results and faster convergence than SEP3.
- We have shown that LA-based solutions can solve elevator-like problems without requiring any knowledge of the underlying distributions. Amazingly enough, the results and solutions that they yielded are near-optimal.

2 The Single Elevator Problem (SEP)

The problem we are trying to tackle can be stated as follows: We have a specific building with n floors and a single elevator. The floors and passengers are characterized by distributions $\mathcal{C} = \{c_1, ..., c_n\}$ and $\mathcal{D} = \{d_1, ..., d_n\}$, where c_i is the probability of receiving a call from floor i, and d_j is the probability that the elevator drops the passenger off at floor j. These distributions are unknown to the decision-making algorithm, and our goal is to design LA-based solutions such that they adaptively determine a set of floors for the e elevators to park at during the idle period so as to minimize the passengers' waiting time.

The metric used as a performance measure in our study is the Average Waiting Time (AWT) of the passengers. This is, clearly, a function of the number of floors, and of also how close the converged solution is to the optimal floor.

2.1 Competitive Solutions

The "Do Nothing" Policy: SEP1 The "Do Nothing" policy, referred to as SEP1, is the one where the systems does nothing after dropping off a passenger. It is formally presented in [2] and [3]. As we can observe from the algorithm, the simulation starts by selecting a random initial parking floor for the elevator. This is done in an equiprobable manner. We then simulate a number of passenger calls that require the elevator to go to the calling floor, dropping-off the passenger at the destination floor, and parking the elevator car at the same destination floor while it waits for the next passenger.

To evaluate the AWT, we calculated the waiting time for each call which, as mentioned, is used to evaluate the performance of the policy. This is done by using the following equation:

$$WT = \theta * |calling\_floor - parking\_floor|, \tag{1}$$

where θ is a parameter characterizing the pace variable, and which differs from one system to another because of different shaft speeds and the associated accelerations/decelerations of the elevator shafts. Since this is constant for all the simulations, we ignored this pace parameter and focused on the travel distance from the parked location to the floor where the call is made, $|calling\_floor - parking\_floor|$, which does not change for different elevator systems even if their paces are different. By doing this, the model would be generalized so that it can be applied to different elevator systems since it will be system independent. This yields the final waiting time equation to be:

$$WT = |calling\_floor - parking\_floor|, \tag{2}$$

which is used to calculate the waiting time for all the policies studied here.

Myopic Policy: SEP2 The second policy, SEP2, was based on the model that was proposed in [12], referred to as a "Myopic Policy". The principle motivating

it is that the model selects a predetermined floor that the elevator waits at for the next call. Using a complete knowledge of the relevant distributions, SEP2 pre-computes the best possible floor that the elevator should park at, so as to minimize the AWT. It uses the Call Distribution, $\mathcal{C}$, to determine that floor.

The main simulation follows the same process as the SEP1 policy in Section 2.1, where the elevator receives a call and then picks the passenger up at that floor and drops the passenger off at the destination. It then moves to the pre-determined floor to wait for the next call. The difference between SEP1 and SEP2 lies in the selection of the parking floor policy, where, instead of waiting at the drop-off floor, it moves to the pre-determined parking floor where it will wait for the next call. The corresponding formal algorithm is in [2] and [3], omitted here in the interest of space.

This algorithm also shows the "optimal floor" selection scheme used to compute the optimal parking floor. This is done by exhaustively searching across all the floors so as to compute which floor produces the minimum expected waiting time as per:

$$T(f) = \sum_{y=1}^{n} |y - f| * g(y), \tag{3}$$

where f is the floor selected as a parking floor, n is the number of floors in the building, $g(y)$ is the probability of receiving a call from the floor y.

The main disadvantage of this policy is that it requires the *a priori* knowledge of $\mathcal{C}$ so as to calculate the expected waiting time for each floor.

3 LA-Based Solutions

In this section, we present our proposed LA-based solutions for the SEP. First, we will show how we have modelled the problem, and thereafter we present an L_{RI}-based solution to the problem. Subsequently, we submit an enhancement to the L_{RI} solution, in which we use the Pursuit concept for the L_{RI}, and this yields the second and even better solution, which is the PL_{RI}-based solution.

3.1 Problem Modelling

Before we present our proposed solutions, we need to explain how the problem was modelled so that it could be solved using an LA approach. As mentioned in Section 1.2, any LA structure consists of an Environment and the LA itself. The LA chooses one of the actions it is offered, i.e., one that is relevant to the problem domain, and then the Environment evaluates it and reacts based on the criterion it uses by responding with a reward or penalty feedback to the LA.

In the SEP, the modelling is much simpler than in the MEP investigated in [4]. In the SEP, we modelled the floors as the actions of the LA, which, hopefully, will eventually converge to one that can be reckoned as the best parking floor. We then modelled the Environment so that it could provide us with the feedback

about whether the selected floor was good or bad. In the former case the decision was rewarded, and in the latter, it was penalized.

To achieve this, we divided the SEP into two different parts, namely the controller and the evaluator. When it concerns the LA-based solution, we can say that the controller is the LA and the evaluator is the Environment, which evaluates the action selected by the controller or the LA.

The LA acts as the elevators' controller, which chooses one of the available floors or (actions) where the elevator will park at to wait for the next passenger call. On the other hand, the Environment evaluates the selected floor based on the objective or fitness function that is specified. One thing to note here is that the LA, or the controller, does not know anything about the distribution of passengers' calls, $\mathcal{C}$, unlike the solutions presented earlier.

3.2 L_{RI}-based Solution: SEP3

Our first proposed solution, referred to as SEP3, is based on the L_{RI} scheme [8], in which the LA updates the actions probabilities when it receives a reward from the Environment, and it does nothing when it receives a penalty. Based on the theory of LA, the L_{RI} scheme helps us to achieve a near-optimal solution by updating the action probabilities to converge towards the best possible solution, which, in our case, is the best parking floor.

The corresponding algorithm is presented in [2] and [3], omitted here in the interest of brevity. Initially, when the simulation of the experiments begins, the LA begins by selecting one of the available floors (actions) as the initial parking floor with equal probability and sends the selected floor to the Environment. Once the Environment receives the selected floor, it evaluates it based on the passengers' AWT from the start of the simulation until that time instance. If the Environment determines that the waiting time is less than or equal to the current AWT, it sends a reward feedback to the LA, informing it that it was a good choice. Otherwise, it sends a penalty feedback.

Once the LA receives the feedback, it checks whether it was a reward or a penalty. If it was a reward, the LA updates the probabilities of the floors according to the previously selected action by increasing the probability of the selected floor and decreasing the probabilities of the rest. On the other hand, if it was penalty feedback, the probabilities remain as they are - with no change.

The LA then selects a new parking floor based on the updated distribution, and repeats the process until it, hopefully, converges to the best parking floor.

3.3 PL_{RI}-based Solution: SEP4

The second LA-based solution, referred to as SEP4, is an improvement on the L_{RI}-based solution, SEP3, and it aimed to achieve an even better performance and faster convergence. To design it, we included the Pursuit phenomenon described in [10], [17] and [19]. This allowed the LA to pursue the action with the superior reward ratio rather than just updating the selected action. The formal algorithm is shown in [2] and [3], and omitted here due to space limitations.

One can observe that it executes in the same manner as SEP3, but it differed when achieving the task of updating the action probabilities. For every action, it accomplished this by keeping track of the *ratio* of the rewards obtained to the number of times the action was selected. Consequently, we introduced the vector of reward estimates to keep track of the reward ratio.

The simulation started by selecting each floor a small number of times (i.e., ten times each in our case), and recording the ratio of how many times each floor received a reward when compared to the number of times it was selected. Thereafter, the simulation proceeded also the same lines as the previous SEP3. The system first receives a call from a passenger, and it calculates the waiting time. It then requests the Environment for the feedback. The Environment evaluates the selected floor, and it uses the average waiting time to decide on the feedback sent to the LA.

If the LA gets a reward, it updates the probabilities of the parking floors. Instead of increasing the probability of selected floor, it pursues the one with the maximum reward estimate and increases its probability. This mechanism ensures that the algorithm converges towards the best solution faster than SEP3.

After that, the LA updates the rewards' estimates and again chooses a parking floor to be evaluated, and repeats the same cycle until it converges.

4 Results and Discussions

We now comparatively discuss the simulations results obtained from the previous solutions and our new solutions. The entires suite of results involve all the methods SEP1, SEP2, SEP3 and SEP4, and for many distributions, namely the Exponential, Inverse Exponential, Gaussian and Bimodal, represented by Exp, $InvExp$, $Gaussian$ and $Bimodal$ respectively. The results that we have obtained are extensive and involve all these distributions and for numerous settings. These are given briefly in [3], but the more detailed set of results is included in the respective chapter and the Appendix of the thesis of the First Author [2].

A very brief summary of the results are given in Table 1, where we submit all the results of the four solutions, for some typical settings and parameters. From the table, we see that SEP1, which we believe is the most popular policy currently used in buildings, performed very poorly in comparison to our proposed solutions. The improvement in the AWT was more than 50% and up to 80%. SEP1 is a lower bound for all schemes, as no solution should be worse than this.

SEP2 was able to give us the optimal parking floors from the beginning, but it required the *a priori* knowledge of $\mathcal{C}$ for each floor. However, our LA-based solutions were able to achieve a close-to-optimal AWT without this knowledge.

SEP3 yielded a performance that was better than SEP1 and recorded an AWT improvement in 78.56% for the Exp, 77.85% for the $InveExp$, 32.49% for the $Gaussian$ and 30.9% for the $Bimodal$. Moreover, the results were close to the values achieved by SEP2.

We attempted to improve SEP3 by proposing SEP4 that incorporated the Pursuit concept. Here, we achieved even better results than what we obtained for

the SEP3. The algorithm helped the system to converge faster to the optimal locations. It also resulted in a better overall AWT. The improvements were 26.3% for the Exp, 29.5% for the $InvExp$, 3% for the $Gaussian$ and 1% for the $Bimodal$ distributions.

In Figure 2, we present the performance of each algorithm and show how our proposed LA-based algorithms were able to achieve an AWT that is very close to the optimal solution, SEP2, and how it significantly outperforms SEP1.

One result that we discovered is that the more equally-distributed the calls are, for example, for the $Gaussian$ distribution, the higher the AWT will be. On the other hand, the importance of having a good policy shines when the calling distribution is skewed toward a specific region or a specific floor. Also, one can observe how the policies affected the AWT in the first and the second scenarios, i.e., for the Exp and $InvExp$ distributions.

Dist	**SEP1**	**SEP2**	**SEP3**	**SEP4**
Exp	$(-, 5.364)$	$(1, 0.712)$	$(2, 1.150)$	$(1, 0.847)$
$InvExp$	$(-, 5.377)$	$(12, 0.751)$	$(11, 1.191)$	$(12, 0.839)$
$Gaussian$	$(-, 3.607)$	$(6, 2.131)$	$(7, 2.435)$	$(6, 2.364)$
$Bimodal$	$(-, 3.707)$	$(8, 2.251)$	$(8, 2.560)$	$(8, 2.533)$

Table 1: Simulation results for the previous and newly proposed solutions for the SEP for an ensemble of 200 experiments for the 12-floor settings. The results are given here as a tuple (α, β) where the first field, α, is the best optimal parking floor and the second field, β is the AWT in terms of number of floors travelled for the elevator to reach the passenger from the parked location.

5 Conclusions

In this paper, we considered the problem of determining the best parking location for the Single Elevator Problem (SEP). We surveyed various parking policies that are currently being used in various building settings, and for previously-reported solutions. We discussed two different solutions for the SEP and tabulated the experimental results that corresponded to various simulations settings. We then introduced our Learning Automata (LA)-based solutions, namely the L_{RI}-based solution, SEP3, which achieved the optimal parking floor without knowing $\mathcal{C}$ *a priori*. We proceeded to introduce an improvement on SEP3 to incorporate the Pursuit concept in SEP4, a PL_{RI}-based solution. From the simulation results for SEP4, we demonstrated that it performed even better than SEP3.

This paper included our findings and compared the results of our proposed solutions to the previous solutions, while extensive additional results are found in [2] and [3]. We showed that our LA-based solutions performed much better

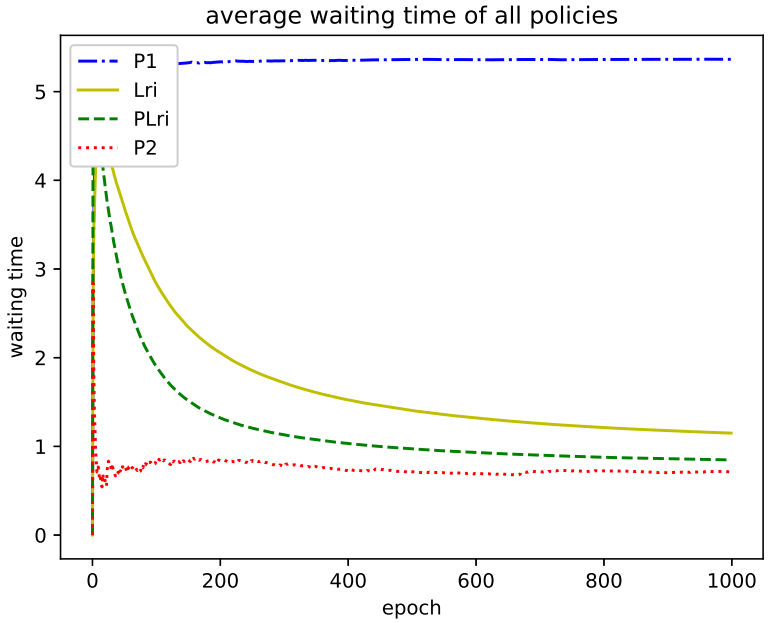

Fig. 2: The Average Waiting Time for SEP1, SEP2, SEP3 and SEP4 for an ensemble of 200 experiments for the case of the *Exp* distribution, given in the graph as $P1$, $P2$, L_{RI}, PL_{RI} respectively.

than SEP1 and converged to the optimal floor. They also reduced the AWT to be close to the optimal value with the advantage that they did not require us to know the distributions to determine the best floor.

The case in which the building use multiple elevators operating in the same environment will be presented in a future paper [4].

References

1. Das, D. and Lee, C. S.: Cross-scene trajectory level intention inference using Gaussian process regression and naive registration. Technical Report, Purdue University, West Lafayette, USA. (2018)
2. Ghaleb, O.: Novel Solutions and Applications to Elevator-like Problems. M.C.S. Thesis, Carleton University, Ottawa, Canada. (2018)
3. Ghaleb, O. and Oommen, B.J.: On Solving Single Evelatior-like Problems using Learning Automata. Unabridged version of this paper. (*In Preperation*). (2019)
4. Ghaleb, O. and Oommen, B.J.: Learning Automata-based Solutions to the Multi-Elevator Problem. (*To be Submitted*). (2019)
5. Kabudian, J., Meybodi, M.R. and Homayounpour, M.M.: Applying continuous action reinforcement learning automata (carla) to global training of hidden markov models. In: Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on. Volume 2., IEEE (2004) 638–642

6. Meybodi, M.R. and Beigy, H.: New learning automata based algorithms for adaptation of backpropagation algorithm parameters. International Journal of Neural Systems **12**(01) (2002) 45–67
7. Molina, S. and Leguizamon, G: An ACO model for a non-stationary formulation of the single elevator problem. Journal of Computer Science & Technology **7**(1) (2007)
8. Narendra, K.S. and Thathachar, M.A.L.T.: Learning Automata: An Introduction Prentice-Hall, New Jersey (1989)
9. Obaidat, M.S., Papadimitriou, G.I., Pomportsis, A.S. and Laskaridis, H.S.: Learning automata-based bus arbitration for shared-medium ATM switches. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **32**(6) (2002) 815–820
10. Oommen, B.J. and Lanctot, J.K.: Discretized pursuit learning automata. Systems, Man and Cybernetics, IEEE Transactions on **20**(4) (1990) 931–938
11. Oommen, B.J. and Roberts, T.D.: Continuous learning automata solutions to the capacity assignment problem. IEEE Transactions on Computers **49**(6) (2000) 608–620
12. Parlar, M., Sharafali, M. and Ou, J.: Optimal parking of idle elevators under myopic and state-dependent policies. European Journal of Operational Research **170**(3) (2006) 863–886
13. Sun, J., Zhao, Q., Luh, P.B. and Atalla, M.J.: Estimation of optimal elevator scheduling performance. Proceedings - IEEE International Conference on Robotics and Automation **2006**(May) (2006) 1078–1083
14. Tanaka, S., Uraguchi, Y. and Araki, M.: Dynamic optimization of the operation of single-car elevator systems with destination hall call registration: Part I. Formulation and simulations. European Journal of Operational Research **167**(2) (2005) 550–573
15. Tanaka, S., Uraguchi, Y. and Araki, M.: Dynamic optimization of the operation of single-car elevator systems with destination hall call registration: Part II. The solution algorithm. European Journal of Operational Research **167**(2) (2005) 550–573
16. Tanaka, S., Innami, Y. and Araki, M.: A study on objective functions for dynamic operation optimization of a single-car elevator system with destination hall call registration. Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics **7** (2004) 6274–6279
17. Thathachar, M.A.L. and Sastry, P.S.: A new approach to the design of reinforcement schemes for learning automata. IEEE Transactions on Systems, Man, and Cybernetics **SCM-15**(1) (1985) 168–175
18. Thathachar, M.A.L. and Sastry, P.S.: A class of rapidly converging algorithms for learning automata. In: IEEE Int. Conf. on Systems, Man and Cybernatics, IEEE (1984)
19. Thathachar, M.A.L. and Sastry, P.S.: Estimator algorithms for learning automata. In: Platinum Jubilee Conference on Systems and Signal Processing. (1986)
20. Tsetlin, M.: On behaviour of finite automata in random medium. Avtomat. i Telemekh **22**(10) (1961) 1345–1354
21. Varshavskii, V. and Vorontsova, I.P.: On the behavior of stochastic automata with a variable structure. Avtomatika i Telemekhanika **24**(3) (1963) 353–360
22. Xu, J. and Feng, T.: Single Elevator Scheduling Problem with Complete Information : An Exact Model using Mixed Integer Linear Programming. (2016) 2894–2899