On Solving Single Elevator-like Problems Using a Learning Automata-based Paradigm

Omar Ghaleb*, and B. John Oommen†

Abstract

This paper[1] concentrates on a host of problems with characteristics similar to those that are related to moving elevators within a building. These are referred to as Elevator-like Problems (ELPs), and their common phenomena will be expanded on in the body of the paper. We shall resolve ELPs using a subfield of AI, the field of Learning Automata (LA). Rather than working with the well-established mathematical formulations of the field, our intention is to use these tools to tackle ELPs, and in particular, those that deal with single "elevators" moving between "floors"[2]. ELPs have not been tackled before using AI. In a simplified domain, the ELP involves the problem of optimizing the scheduling of elevators. In particular, we are concerned with determining the Elevators' optimal "parking" location. In our case, 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 as the multi-elevator setting is more complicated. 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 first solution, leading to the PL_{RI} scheme. The simulation results presented demonstrate that our solutions performed better than those used in modern-day elevators, and provided results that are near-optimal, yielding a performance increase of up to 90%. The solutions presented for real elevators are directly applicable for the entire family of ELPs.

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

*This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. E-mail: omar.ghaleb@carleton.ca.

†Author's status: *Chancellor's Professor, Life Fellow: IEEE* and *Fellow: IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. The author is also an Adjunct Professor with University of Agder, Grimstad, Norway. E-mail: oommen@scs.carleton.ca.

[1]This paper is a significant extension to a very abridged conference paper [2], which was a brief introduction to the problem and contained some *initial* solutions [1].

[2]The concepts of floors and parking for real elevators are but simplified versions of their abstract representations in ELPs.

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 behaviour from the behaviour of a real human being. The goal, although lofty, 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.

It is staggering to record the areas in which AI has been used. Machine learning, pattern recognition, medical diagnosis and voice-operated systems are commonplace in today's world. AI has been used in practically every single application domain. 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 attempted to incorporate AI in different problem domains, such as autonomous driving, playing games, diagnosis and security. They have also worked extensively on different subfields of AI such as machine learning, pattern recognition and voice operated-systems.

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. In other words, 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 has been dropped off? Of course, the answer to this question depends on the criterion function, but it is expedient to work with the understanding that the car should be parked near to the next passenger call. The same analogy can be seen if we extend the problem domain to know where police or emergency vehicles should be parked so as to be available, in the shortest possible time, 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. Similarly, one could consider the problem of where security guards, who move on a rotational basis, should be placed so that the facility is maximally secured. 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) and explain the solutions by which previous researchers have tackled it. The SEP is a subclass of the Multi Elevator Problem (MEP), where we have a building with n floors and a single elevator, and where we are to design a policy for the elevator so as to save energy, reduce the travel time or the waiting time for passengers. The elevator policy should decide how the elevator should operate so as to achieve its goal. For example, one possible policy could require that the elevator picks the best route, 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 some simple straightforward criteria for both the SEP and the MEP.

Two notable contributions for the SEP were by Tanaka *et al.* in [16, 17]. In their papers, they studied the operation problem of a single-car elevator with so-called "destination hall call registration". "Destination hall call registration" is a system that is responsible for passengers registering their destination floors at the hall *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 [16], the authors were trying 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. This set the stage for the second question. For the first question, they were able to show, by simulation, that the DOP substantially improved the capability of the transportation when compared to the conventional "selective collective" operation.

In the second part [17], the authors introduced a branch-and-bound algorithm to solve the formulation of the DOP problem, which was the exact formulation proposed for the work in [16]. In this case, the lower bound calculations for the subproblems, that were generated in the course of the branch-and-bound algorithm, came from decomposing the problem into three subproblems, i.e., the passenger loading and unloading, car stops and lastly the car floor-to-floor travel. They then applied the Lagrangian relaxation method to solve the overall problem. Unfortunately, the authors observed that their algorithm was not fast enough for real-life situations. To improve it, they had to take into consideration the constraints of the elevator's capacity.

After their previous studies in [16] and [17], the authors conducted a study in [15] where they tried to examine how one could improve the efficiency of a single elevator system with "destination hall call" considering the objective functions that had to be dynamically optimized. In this paper, the authors applied a simulated annealing-based method, and from their results they showed that the weighted average of two different objective functions, such as the weighted average service time and the maximum, yielded a better performance than using only a single objective function. Another point that they observed was that one had to choose the weights of the objective functions carefully because this choice significantly affected the results of the experiments. Moreover, these weights changed depending on the elevator's specific characteristics.

The authors of [14] 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. Because of the large numbers of decision variables that had to be taken into consideration, one could not compute the exact optimal performance, and they, thus, provided its estimate. 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. In both, they defined a "trip" as the start of the elevator's movement in a single direction until the elevator reversed to go to the other direction. This 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 then obtained the lower bound by defining a lower bound for a newly constructed problem using Lagrangian Relaxation. This method permitted the approximation of a constrained optimization problem by a more straightforward problem. Moreover, the solution to the simpler problem provided an approximation to the original problem. The results that the authors obtained were both efficient and scalable.

A subsequent work was done by Molina *et al.* [6], where they designed an algorithm that is based on Ant Colony Optimization (ACO) for solving the SEP. The ACO model involved a partially-connected construction graph used by the artificial ants to construct the solutions to the SEP. It provided a sequence of visits to the requested floors to minimize the Average Waiting Time (AWT) for the passengers. They introduced an objective function that specified how good the proposed solution was, when the problem was formulated as a combinatorial optimization problem. Using the latter, the authors were able to minimize the AWT for passengers by invoking straightforward well-known ACO methods. In their experiments, they focused on finding the best parameters for their algorithm, which they referred to as the *ACS-elevator* to obtain the best sequence of visits for the elevator. These parameters were the initial positions of the elevator, the maximum capacity of the elevator, the number of floors in the building, the length of the sequence of visits or the solution, the maximum number of algorithm iterations, etc.

Another work was presented by Xu and Feng [23], who 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 also introduced many constraints to accelerate the computation for solving the MILP. They tested their model under different circumstances and scenarios. After introducing their constraints, they observed that the problem converged in a radically short time. They conjectured that their model could be extended and used as a benchmark because of its simple implementation and its speed.

1.1 Learning Automata (LA)

We now concentrate on 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 [21]. 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 *reward* received from the *Environment* and minimizes the *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 LA.

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 [21] and Varshavskii in [22].

The field of LA, like many of the Reinforcement Learning techniques, has been used in a variety of (mainly optimization) problems, and in many AI applications. It has been used in neural network adaptation [4], solving communication and networking problems [5], [8], [10] and [11], in distributed scheduling problems [13], and in the training of Hidden Markov Models [3].

In this section, we will cover the relevant background required to help the reader understand the fundamental concepts for our proposed work[3].

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 LA.

[3]We will not go through irrelevant details and/or the proofs of the LA-related claims. This overview section can be abridged if the Referees request it.

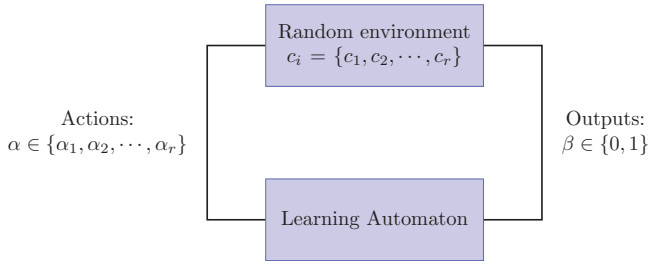


Figure 1: The Automaton-Environment Feedback Loop.

1.2 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, to the automat, a feedback signal, depending on whether that action resulted in a *Reward* or a *Penalty*.

The Environment can be mathematically described as a triple: $\mathbb{E} = \{\underline{\alpha}, \underline{c}, \underline{\beta}\}$, where:

- $\underline{\alpha}$: is the set of actions $\{\alpha_1, \alpha_2, ..., \alpha_r\}$
- $\underline{\beta}$: is the set feedbacks, where, typically, $\underline{\beta} = \{0, 1\}$, and is transmitted from $\mathbb{E}$ to the LA
- $\underline{c}$: is the set of penalty probabilities associated with the Environment, and it corresponds to the set of actions $\underline{\alpha}$ where:

$$c_i = \Pr\left[\beta\left(n\right) = 1 | \alpha\left(n\right) = \alpha_i\right] \qquad (i = 1, 2, ..., r).$$

The Environment, $\mathbb{E}$, can be classified as being one of two types. $\mathbb{E}$ is stationary when the penalty probabilities are constant. On the other hand, it is non-stationary if it has penalty probabilities that change with time.

1.3 Automaton

Narendra and Thathachar [7], the pioneers of the field, define the LA as a quintuple: $\{\Phi, \underline{\alpha}, \underline{\beta}, F\left(\cdot, \cdot\right), G\left(\cdot\right)\}$ where:

- $\Phi = \{\phi_1, \phi_2, ..., \phi_s\}$ represents the set of states. $\phi\left(n\right)$ is the current state at time n.
- $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents the set of actions that the *automaton* can pick from. $\alpha\left(n\right)$ is the action selected by the automaton at time n.
- $\underline{\beta} = \{0, 1\}$ represents the set of possible feedback signals transmitted by $\mathbb{E}$. $\beta = 0$ is the case when $\mathbb{E}$ *rewards* the taken action, and $\beta = 1$ is the case when $\mathbb{E}$ *penalizes* it.
- F represents the transition function for the LA from the current state $\phi(n)$ to the next state $\phi(n+1)$. Formally, $\phi(n+1) = F(\phi(n), \beta(n))$
- G represents the output function of the automaton, where G can be *stochastic* or *deterministic*. This output specifies the selection of the action by the LA. Formally, $\alpha(n) = G(\phi(n))$.

The LA achieves its learning process as an iterative operation that is based on $\mathbb{E}$ and the interaction between them. This process consists of two main steps which are, *policy evaluation*, which is how $\mathbb{E}$ evaluates the selected action. Moreover, the second step involves *policy improvement*, where the LA improves the probability of selecting an action that would maximize the *reward* received from $\mathbb{E}$.

We can formalize the LA in terms of the following equation $\forall\ n$:

$$\begin{aligned} &p_i(n) = \Pr[\,\alpha(n) = \alpha_i\,] :\ (i = 1, ..., r),\ \text{and} \\ &\sum_{i=1}^{r} p_i(n) = 1, \end{aligned} \tag{1}$$

where $p_i\,(n)$ is the probability of the LA choosing an action α_i at time n, and where the collective probability vector, $P(n)$, has all the action probabilities in the corresponding indices of that vector.

The performance of the LA is measured by the *average penalty*, $M(n)$, for the vector $P(n)$, specified by:

$$\begin{aligned} M(n) &= E\,[\,\beta(n)|P(n)\,] = \Pr\,[\,\beta(n) = 1|P(n)\,] \\ &= \sum_{i=1}^{r} c_i p_i(n). \end{aligned} \tag{2}$$

Consider the *pure-chance* automaton, which does not have any preference or biases for any actions. In this automaton, the average penalty M_0 is given:

$$M_0 = \frac{1}{r} \sum_{i=1}^{r} c_i. \tag{3}$$

This *pure-chance* automaton is used to compare the performances of different LA schemes. For the LA to be a well-performing LA, it has to be at least better than the *pure-chance* LA. Hence, we compare $E\,[M(n)]$ with M_0 and determine whether the expected average penalty is better than the *pure-chance* machine or not.

We can characterize the LA by four possibilities having four different cases for $E\,[M(n)]$. If the LA performs better than the *pure-chance* machine, we say that we have an *Expedient* LA, where:

$$\lim_{n\to\infty} E\,[M(n)] < M_0. \tag{4}$$

Secondly, we may have an *optimal* LA when:

$$\lim_{n\to\infty} E[\,M(n)\,] = c_l, \quad \text{where} \quad c_l = \min_i \{\,c_i\,\}. \tag{5}$$

Thirdly, an LA is said to be ϵ-optimal whenever:

$$\lim_{n\to\infty} E\,[\,M(n)\,] \leq c_l + \epsilon, \tag{6}$$

where $\epsilon > 0$ is an arbitrarily small user-defined value.

Finally, we say that the LA is *Absolutely Expedient* when:

$$E\,[\,M(n+1)|P(n)\,] < M(n). \tag{7}$$

1.4 Estimator and Pursuit Algorithms

The concept of Estimator Algorithms was introduced by Thathachar and Sastry in [18,19] 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. This concept was achieved by keeping track of the number of rewards received by the selected actions and by pursuing the ones with the superior estimates. 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 [20]. 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. After the initial family of Pursuit algorithms, the concept of designing discretized Pursuit LA was introduced by Oommen and Lanctot in [9]. These LA were also proven to be ϵ-optimal. The discretized versions of LA were shown to converge faster than their continuous counterparts.

1.5 Contributions of this Paper

The novel contributions of this paper are:

- We have surveyed a subfield of AI, namely the field of Learning Automata (LA), and have concluded that it has not been used previously to solve the Single Elevator Problem.
- 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.
- To summarize, 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 $\mathcal{C} = \{ca_1, ..., ca_n\}$ and $\mathcal{D} = \{de_1, ..., de_n\}$, where ca_i is the probability of receiving a call from floor i, and de_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. In this paper we deal with the case when $e = 1$.

The metric used as a performance measure in our study is the 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.

3 Simulation Settings

Simulations require mutual specific configurations or settings so as to be able to compare the different results obtained from different solutions. In our simulations, we will incorporate different simulation settings for the buildings. The first item for these settings is the number of floors that the building has. In the interest of uniformity, we will test the models on four different types of buildings with varying numbers of floors, which are 8-floor, 12-floor, 16-floor and 20-floor buildings.

We will also perform the simulations using four different *types* of distributions[4] for $\mathcal{C}$, listed below:

1. The first is an exponential distribution designed, referred to as Exp, to simulate an up-peak traffic pattern, where most calls come from the ground and/or lower floors, and where the passengers intend to travel up;

2. The second distribution is the inverse exponential distribution, referred to as $InvExp$, that represents a down-peak traffic pattern, where most calls are from the upper floors, and where the passengers travel downward;

3. The third distribution is the Gaussian (Normal) distribution, referred to as $Gaussian$, and this is intended to represent the traffic of passengers during the middle of the day, also referred to as "regular traffic";

4. The final distribution that we will use is a bimodal distribution, referred to as $Bimodal$, which is specified as a mixture of Gaussian distributions to represent a more complex "regular traffic" pattern.

Given a particular distribution for $\mathcal{C}$, we now specify how the discretized probabilities for the respective floors are obtained. We clarify this in Figure 2 for a 12-floor building with a bimodal distribution. The heights of the bimodal curve for the 12 uniformly placed points on the x-axis represent the corresponding probabilities. The final values of the probabilities are obtained by normalizing these heights so that the sum becomes unity.

Tables 1, 2, 3 and 4 represent the calling probabilities obtained from the four different distributions mentioned above for 8, 12, 16 and 20-floor buildings respectively. The corresponding tables for the Destination distributions are obtained in an analogous manner.

4 Competitive Solutions

4.1 Do Nothing Policy: SEP1

The Do_Nothing policy, referred to as SEP1, is formally presented in Algorithm 1. 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

[4]It is a trivial task to examine other distributions, for example, ones for which c_1 is close to unity, as in the real-life setting of early morning traffic.

Table 1: Simulation settings for $\mathcal{C}$ for an 8-floor building for different distributions. The values represent the probabilities of receiving a passenger call from each floor in accordance with the associated distribution.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodel$
ca_1	0.59387372	0.00109053	0.08215232	0.01898451
ca_2	0.24145104	0.00268228	0.1146528	0.08516571
ca_3	0.09816667	0.00659734	0.14318402	0.14316436
ca_4	0.03991159	0.01622684	0.16001087	0.12033347
ca_5	0.01622684	0.03991159	0.16001087	0.17698614
ca_6	0.00659734	0.09816667	0.14318402	0.2620605
ca_7	0.00268228	0.24145104	0.1146528	0.15804965
ca_8	0.00109053	0.59387372	0.08215232	0.03525567

Table 2: Simulation settings for $\mathcal{C}$ for a 12-floor building. The meaning of the entries is as described in the caption of Table 1.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodal$
ca_1	0.59344245	0.00002978	0.02592411	0.01270883
ca_2	0.24127569	0.00007324	0.04518336	0.03860891
ca_3	0.09809538	0.00018013	0.07046903	0.07525205
ca_4	0.0398826	0.00044306	0.09834746	0.09457958
ca_5	0.01621506	0.00108974	0.12282111	0.08017525
ca_6	0.00659255	0.00268033	0.13725493	0.06220755
ca_7	0.00268033	0.00659255	0.13725493	0.08440493
ca_8	0.00108974	0.01621506	0.12282111	0.14232761
ca_9	0.00044306	0.0398826	0.09834746	0.17475873
ca_{10}	0.00018013	0.09809538	0.07046903	0.13967666
ca_{11}	0.00007324	0.24127569	0.04518336	0.07169797
ca_{12}	0.00002978	0.59344245	0.02592411	0.02360195

Table 3: Simulation settings for $\mathcal{C}$ for a 16-floor building. The meaning of the entries is as described in the caption of Table 1.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodal$
ca_1	0.59343067	0.00000081	0.00588623	0.00309355
ca_2	0.24127091	0.000002	0.01281213	0.00952918
ca_3	0.09809343	0.00000492	0.02495463	0.02286339
ca_4	0.03988181	0.00001211	0.04349365	0.04274851
ca_5	0.01621473	0.00002978	0.06783372	0.06242093
ca_6	0.00659242	0.00007324	0.09466958	0.07186065
ca_7	0.00268028	0.00018013	0.118228	0.06788
ca_8	0.00108972	0.00044305	0.13212205	0.0604008
ca_9	0.00044305	0.00108972	0.13212205	0.06530901
ca_{10}	0.00018013	0.00268028	0.118228	0.08883733
ca_{11}	0.00007324	0.00659242	0.09466958	0.1184869
ca_{12}	0.00002978	0.01621473	0.06783372	0.13154
ca_{13}	0.00001211	0.03988181	0.04349365	0.1155474
ca_{14}	0.00000492	0.09809343	0.02495463	0.07933226
ca_{15}	0.000002	0.24127091	0.01281213	0.04245367
ca_{16}	0.00000081	0.59343067	0.00588623	0.01769641

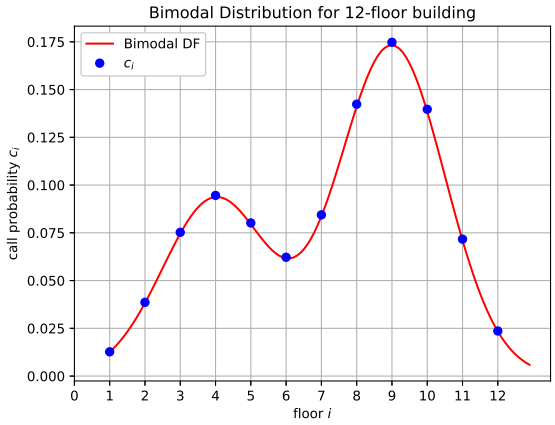


Figure 2: A typical bimodal distribution for a 12-floor building and the corresponding discretized probabilities constituting $\mathcal{C}$.

Table 4: Simulation settings for $\mathcal{C}$ for a 20-floor building. The meaning of the entries is as described in the caption of Table 1.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodal$
ca_1	0.59343035	0.00000002	0.00088438	0.00077597
ca_2	0.24127078	0.00000005	0.00240398	0.00306903
ca_3	0.09809338	0.00000013	0.00584751	0.00945329
ca_4	0.03988179	0.00000033	0.01272787	0.02267767
ca_5	0.01621473	0.00000081	0.02479051	0.04237177
ca_6	0.00659242	0.000002	0.04320761	0.06168651
ca_7	0.00268028	0.00000492	0.0673876	0.07013443
ca_8	0.00108972	0.00001211	0.09404697	0.06308409
ca_9	0.00044305	0.00002978	0.11745045	0.04806619
ca_{10}	0.00018013	0.00007324	0.13125312	0.04023324
ca_{11}	0.00007324	0.00018013	0.13125312	0.05156804
ca_{12}	0.00002978	0.00044305	0.11745045	0.0817498
ca_{13}	0.00001211	0.00108972	0.09404697	0.11525582
ca_{14}	0.00000492	0.00268028	0.0673876	0.12987546
ca_{15}	0.000002	0.00659242	0.04320761	0.11450328
ca_{16}	0.00000081	0.01621473	0.02479051	0.07868357
ca_{17}	0.00000033	0.03988179	0.01272787	0.04211504
ca_{18}	0.00000013	0.09809338	0.00584751	0.01755607
ca_{19}	0.00000005	0.24127078	0.00240398	0.00569962
ca_{20}	0.00000002	0.59343035	0.00088438	0.00144109

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 then parking the elevator car at the same destination floor while it waits for the next passenger call.

To evaluate the average waiting time, 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:

$$WT = \theta * |calling\_floor - parking\_floor|, \tag{8}$$

where θ is a parameter characterizing the pace variable, and which differs from one system to another because of

Algorithm 1 SEP1: Do Nothing Policy

Input:

- The $Call\_Distribution\ \mathcal{C} = \{ca_1, ..., ca_n\}$
- The $Destination\_Distribution\ \mathcal{D} = \{de_1, ..., de_n\}$

Output:

- Average waiting time achieved.

begin
  Initialize $parking\_floor$, chosen randomly with equal probability
  for a number of $calls$ **do**
    $calling\_floor$ = Call($Call\_Distribution$)
    $waiting\_time$ = $|calling\_floor - parking\_floor|$
    $destination\_floor$ = Call($Destination\_Distribution$)
    $parking\_floor$ = $destination\_floor$
  end for
  Print out average waiting time.
end

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:

$$WT = |calling\_floor - parking\_floor|, \tag{9}$$

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

4.1.1 Simulation Results

To test the models, we ran a number of simulations for different building settings, as mentioned in Section 3. We ran our tests on all the settings, but we will present and discuss, in the body of this section, the results for the 12-floor scenario with the four different types of call distributions specified above, and for a uniform destination distribution. A more detailed and comprehensive set of results and plots for numerous other types of buildings and distributions is included in the thesis of the First Author [1], but not reported here in the interest of space.

In Table 5, we present the different call distributions for the 12-floor building, in which we tested SEP1 on, and the results of these simulations. The table shows the ensemble AWT for passengers for an ensemble of 200 experiments, and where the number of iterations (i.e., passenger calls) was $1,000$. In the table, $Dist$ refers to the type of distribution used in that simulation, c_i is the call probability at floor i, and AWT is the corresponding average waiting time of the passengers in that experiment.

To cite one example, for the exponential distribution, Exp, we obtained an average waiting time, AWT, of 5.36. This means that if the building has call probabilities as per this distribution, the elevator with no intelligent parking policy will result in having an AWT of almost 6 floors distance.

Similarly in the inverse exponential distribution, $InvExp$, SEP1 yielded very similar results, with an AWT of 5.38, which is very reasonable since it merely "inverses" the probabilities to be in an increasing order.

In the Gaussian distribution, $Gaussian$, the AWT was reduced to 3.6 as the traffic was evenly more distributed

Table 5: Simulation results for a 12-floor building for the policy SEP1 for an ensemble of 200 experiments. The results reported are the average waiting times for passengers in terms of number of floors for the elevator car to travel so as to reach the next call from the parked location.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodal$
ca_1	0.59344245	0.00002978	0.02592411	0.01270883
ca_2	0.24127569	0.00007324	0.04518336	0.03860891
ca_3	0.09809538	0.00018013	0.07046903	0.07525205
ca_4	0.0398826	0.00044306	0.09834746	0.09457958
ca_5	0.01621506	0.00108974	0.12282111	0.08017525
ca_6	0.00659255	0.00268033	0.13725493	0.06220755
ca_7	0.00268033	0.00659255	0.13725493	0.08440493
ca_8	0.00108974	0.01621506	0.12282111	0.14232761
ca_9	0.00044306	0.0398826	0.09834746	0.17475873
ca_{10}	0.00018013	0.09809538	0.07046903	0.13967666
ca_{11}	0.00007324	0.24127569	0.04518336	0.07169797
ca_{12}	0.00002978	0.59344245	0.02592411	0.02360195
AWT	5.36434	5.376775	3.60748	3.707325

across the building. Similarly, in the more complicated distribution, $Bimodal$, the value was close to the Gaussian, with an AWT of 3.7.

Since the parking policy in SEP1 depends on the destination floor, the AWT is heavily affected by both $\mathcal{C}$ and $\mathcal{D}$. If both the distributions are skewed towards a specific area of the building, this will result in a small value for AWT. If, however, they are opposing each other, such as having $\mathcal{C}$ to be Exp and $\mathcal{D}$ to be $InvExp$, it will produce a very high AWT. Many of these simulations results are included here, but additional results are found in [1].

In Figure 3, we plot the results of the AWT for an ensemble of 200 experiments for the Exp Distribution, where the number of passenger calls is $1,000$. Observe that the SEP1 policy quickly leads to the final converged value of the AWT, in less than 50 passenger calls.

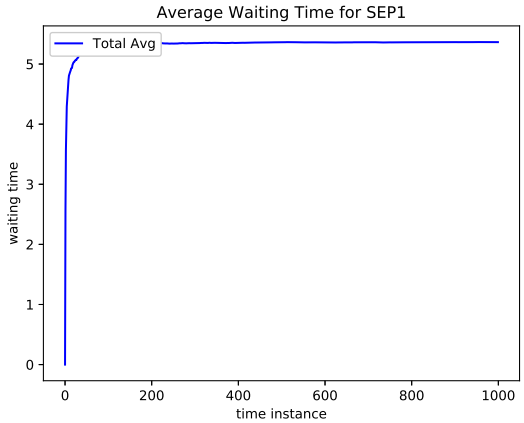


Figure 3: The Average Waiting Time for the "Do Nothing" Policy, SEP1, for an ensemble of 200 experiments for the case of the Exp distribution.

4.2 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. The model precomputes the best possible floor that the elevator should park at, so as to minimize the average waiting time. This model resorts to using the Call Distribution, $\mathcal{C}$, to determine that floor.

The main simulation follows the same process as the SEP1 policy in Section 4.1, 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 algorithm is in Algorithm 2.

Algorithm 2 SEP2: Myopic Policy

Input:
- The $Call\_Distribution\ \mathcal{C} = \{ca_1, ..., ca_n\}$
- The $Destination\_Distribution\ \mathcal{D} = \{de_1, ..., de_n\}$
- Number of floors n

Output:
- Optimal parking floor,
- Average waiting time achieved.

1: **begin**
2: $optimal\_parking\_floor$ = Call (Optimal Floor)
3: **for** a number of $calls$ **do**
4: $calling\_floor$ = Call($Call\_Distribution$)
5: $waiting\_time$ = $|calling\_floor - optimal\_parking\_floor|$
6: Update average waiting time($waiting\_time$)
7: $destination\_floor$ = Call($Destination\_Distribution$)
8: $parking\_floor$ = $optimal\_parking\_floor$
9: **end for**
10: Return $optimal\_parking\_floor$
11: Print average waiting time.
12: **end**

Algorithm 3 shows the "optimal floor" selection algorithm that is 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 described in Eq. (10).

$$T(f) = \sum_{y=1}^{n} |y - f| * g(y), \tag{10}$$

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.

4.2.1 Simulation Results

To test the performance of SEP2, the myopic policy, we used the same building settings mentioned previously. For the sake of comparison, we will include the results for the same settings that we showed the results for, in SEP1. In

Algorithm 3 SEP2: Optimal Floor

Input:

- The $Call\_Distribution\ \mathcal{C} = \{ca_1, ..., ca_n\}$,
- Number of floors n

Output:

- Optimal parking floor

begin
  $minimum\_waiting\_time = \infty$
  for each f in $floors$ **do**
    $expected\_waiting\_time = \sum_{y=1}^{n} |y - f| * g(y)$
    if $expected\_waiting\_time \leq minimum\_waiting\_time$ **then**
      $minimum\_waiting\_time = expected\_waiting\_time$
      $best\_floor = f$
    else
      do_nothing
    end if
  end for
  return $best\_floor$
end

Table 6, we use the same legend as in Table 5 with the addition of BF which is the "best parking floor" calculated.

From Table 6, where we considered the case of 12 floors, we can see that the performance of SEP2 exceeded that of SEP1, especially when the distribution is skewed towards a specific floor, as in Exp and $InvExp$. The AWT in the Exp scenario decreased significantly from 5.36 to 0.71. Unlike SEP1, the algorithm determined a floor that it will always wait at for the next call, which is the first floor in the Exp setting.

Similarly, in the $InvExp$ scenario, the results showed a huge decrease in the AWT, and the algorithm determined that the best parking floor was floor 12, and that resulted in an AWT of 0.75, which was reduced significantly from 5.38. Both the Exp and $InvExp$ scenarios yielded a huge decrease in the AWT of about 85%.

In the $Gaussian$ and $Bimodal$ cases, the improvements in the performance were not as significant as in the previous distributions, as most of the calls were more distributed across all the floors. So it is reasonable to expect a higher AWT because more calls originated from other floors. For the $Gaussian$ case, the algorithm picked floor 6 to be the optimal one, and that reduced the AWT to 2.13 from 3.61. For the $Bimodal$ case, floor 8 was selected to be optimal, and that reduced the AWT to 2.25 from 3.71. In both distributions, the decrease in waiting time was around 40%.

In Figure 4, we plot the results of the AWT for an ensemble of 200 experiments for the Exp distribution, where the number of passenger calls is $1,000$. Observe that for the SEP2, the value first increased steeply, and was then able to converge to a final small value of AWT in less than 100 calls. This is very reasonable since it calculates the best floor before the simulation deals with the calls.

In Figure 5, one can observe the difference in the performance between the two policies, SEP1 and SEP2, and how the SEP2 outperformed SEP1 to produce a very low AWT in comparison with SEP1.

The problem with SEP2 is that to achieve these results, the algorithm needed to know $\mathcal{C}$ so as to calculate the best parking floor. In the next section, we will show how one can obtain even more superior results or (close to the optimal results) with LA-based solutions that do not require this knowledge.

Table 6: Simulation results for a 12-floor building for the policy SEP2 for an ensemble of 200 experiments. The results reported are the average waiting times for passengers in terms of number of floors for the elevator car to travel so as to reach the next call from the parked location.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodal$
ca_1	0.59344245	0.00002978	0.02592411	0.01270883
ca_2	0.24127569	0.00007324	0.04518336	0.03860891
ca_3	0.09809538	0.00018013	0.07046903	0.07525205
ca_4	0.0398826	0.00044306	0.09834746	0.09457958
ca_5	0.01621506	0.00108974	0.12282111	0.08017525
ca_6	0.00659255	0.00268033	0.13725493	0.06220755
ca_7	0.00268033	0.00659255	0.13725493	0.08440493
ca_8	0.00108974	0.01621506	0.12282111	0.14232761
ca_9	0.00044306	0.0398826	0.09834746	0.17475873
ca_{10}	0.00018013	0.09809538	0.07046903	0.13967666
ca_{11}	0.00007324	0.24127569	0.04518336	0.07169797
ca_{12}	0.00002978	0.59344245	0.02592411	0.02360195
$BestFloor$	1	12	6	8
AWT	0.71168	0.750515	2.131015	2.250805

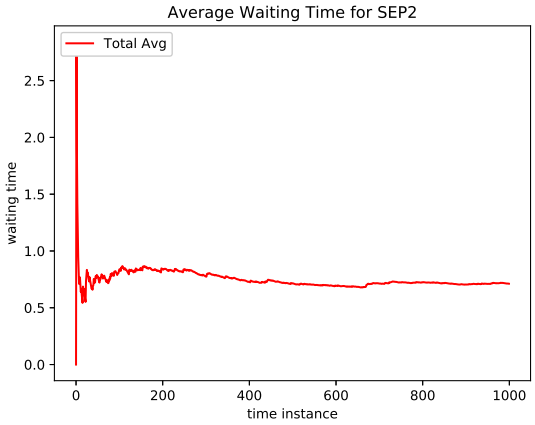


Figure 4: Average Waiting Time for the "Myopic Policy", SEP2, for an ensemble of 200 experiments for the case of Experiment $Ex(1)$.

5 LA-Based Solutions

In this section, we are going to 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 on the L_{RI} solution, in which we use the pursuit concept for the L_{RI}, and this yielded the second and even better solution, which is the PL_{RI}-based solution.

5.1 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, any LA structure consists of an Environment and the

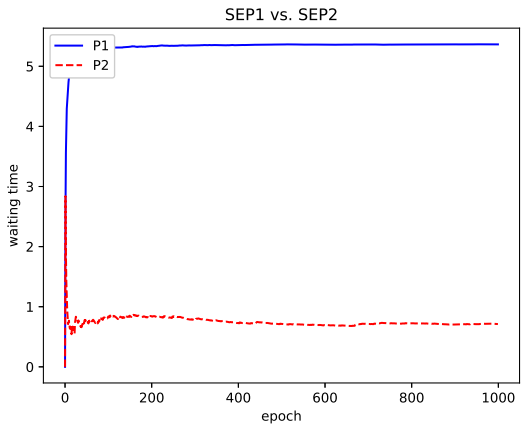


Figure 5: The Average Waiting Time for SEP1 *vs.* SEP2, for an ensemble of 200 experiments for the case of the Exp distribution.

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 later. 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.

5.2 L_{RI}-based Solution: SEP3

Our first proposed solution, referred to as SEP3, is based on the L_{RI} scheme, where 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.

We present the corresponding algorithm in Algorithm 4. 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, as shown in Algorithm 5. If the Environment evaluates that the waiting time is less than or equal to the AWT, it sends a reward feedback to the LA, informing it that it was a good choice. Otherwise, it sends a penalty feedback.

Algorithm 4 $SEP3 : SepLri()$

Input:

- The Call Distribution $\mathcal{C} = \{ca_1, ..., ca_n\}$,
- The Destination Distribution $\mathcal{D} = \{de_1, ..., de_n\}$,
- Number of floors n

Output:

- Optimal parking floor,
- Average waiting time achieved.

begin
  $parking\_distribution\ \mathcal{P} = \{p_1, ..., p_n\}$ initially Equal Probabilities
  $parking\_floor$ = chosen randomly with equal probability
  for a number of $calls$ **do**
    $calling\_floor$ = Call($Call\_Distribution$)
    $waiting\_time$ = $|calling\_floor - parking\_floor|$
    $feedback$ = call ($EnvironmentFeedback()$)
    Update average waiting time($waiting\_time$)
    $destination\_floor$ = Call($destination\_distribution$)
    if $feedback = reward$ **then**
      call ($UpdateOnReward(parking\_distribution)$)
    else
      $DoNothing$
    end if
    $parking\_floor$ = Call($parking\_distribution$)
  end for
  $optimal\_parking\_floor$ = Converged Floor
  Print out average waiting time.
end

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 in an L_{RI} manner. In Algorithm 6 we present how the LA updates the probabilities of the floors.

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

Algorithm 5 $SEP3 : EnvironmentFeedback()$

Input:

- Passenger waiting time, $waiting\_time$.
- Average waiting Time, $average\_waiting\_time$.

Output:

- $Reward$ 0 or $Penalty$ 1.

begin
  if $waiting\_time \leq average\_waiting\_time$ **then**
    $feedback = 0$
  else
    $feedback = 1$
  end if
  return $feedback$
end

Algorithm 6 $SEP3 : UpdateOnReward()$

Input:

- The parking (actions) probabilities $\mathcal{P}(t) = \{p_1, ..., p_n\}$,
- Selected parking floor n,
- Learning rate a

Output:

- Updated parking probabilities $\mathcal{P}(t+1)$.

begin
  for each p_i in $\mathcal{P}$ **do**
    if $i = n$ **then**
      $p_i(t+1) = p_i(t) + a * (1 - p_i(t))$
    else
      $p_i(t+1) = (1-a) * p_i(t)$
    end if
  end for
  return $\mathcal{P}$
end

5.2.1 Simulation Results

To test our proposed solution, we used the same building settings that were used earlier in Sections 4.1 and 4.2 for the purpose of a fair comparison. We discuss the results below in Table 7.

From Table 7, one can observe that our proposed solution yielded the optimal parking floor with a value that is very close to the optimal AWT.

Consider the first scenario with the exponential distribution, Exp. The obtained results showed that the proposed solution converged to the optimal floor, which is the first floor, and was able to reach a better AWT than SEP1 and very close to SEP2, with an AWT of around 1.15 floors to reach the calling floor.

Similarly, in the second scenario, $InvExp$, our results demonstrated that the L_{RI} algorithm was able to converge to a floor close to the best floor, which is the 11th floor. Since this distribution is merely the "inverse" of the previous distribution, this was a very reasonable choice. This yielded an AWT of 1.19. Both the first and second scenarios showed a huge increase in performance in comparison with SEP1 and also produced results comparable to the SEP2 without requiring them to know $\mathcal{C}$ *a priori*.

In the third scenario, $Gaussian$, where most calls were not concentrated and where they were more evenly distributed, the results showed that it achieved a very low AWT of 2.43 when it converged to floor 7.

Similarly, for the final scenario, $Bimodal$, it was able to produce an even lower value of AWT than the SEP1 with an AWT of 2.56, when it converged to floor 8. One can observe here that the impact of the proposed algorithm is very noticeable in the first two scenarios where the calls were skewed towards a specific zone or a floor.

The overall performance of SEP3 yielded a significant decrease in the AWT that was between 50% to almost 80% of SEP1's performance. It was also able to produce results that were very close to SEP2, where the optimal solution was known from the beginning. As opposed to this, in the LA-based case, we did not use any methods to calculate the best floor using any known distribution. Instead, SEP3 adapted to the Environment and was able to conclude the identity of the best floor to be used as a parking floor.

Figure 6 demonstrates the behaviour of the AWT for SEP3 over the number of calls as it decreases to be close to a single floor.

Table 7: Simulation results for a 12-floor building for the policy SEP3 for an ensemble of 200 experiments. The results reported are the average waiting times for passengers in terms of number of floors for the elevator car to travel so as to reach the next call from the parked location.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodal$
ca_1	0.59344245	0.00002978	0.02592411	0.01270883
ca_2	0.24127569	0.00007324	0.04518336	0.03860891
ca_3	0.09809538	0.00018013	0.07046903	0.07525205
ca_4	0.0398826	0.00044306	0.09834746	0.09457958
ca_5	0.01621506	0.00108974	0.12282111	0.08017525
ca_6	0.00659255	0.00268033	0.13725493	0.06220755
ca_7	0.00268033	0.00659255	0.13725493	0.08440493
ca_8	0.00108974	0.01621506	0.12282111	0.14232761
ca_9	0.00044306	0.0398826	0.09834746	0.17475873
ca_{10}	0.00018013	0.09809538	0.07046903	0.13967666
ca_{11}	0.00007324	0.24127569	0.04518336	0.07169797
ca_{12}	0.00002978	0.59344245	0.02592411	0.02360195
$BestFloor$	2	11	7	8
AWT	1.14951	1.191435	2.434935	2.560355

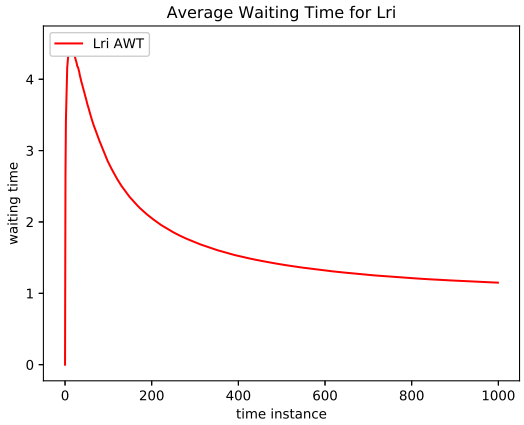


Figure 6: The Average Waiting Time for the L_{RI} solution, SEP3, for an ensemble of 200 experiments for the case of the Exp distribution.

5.3 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 Section 1.4. 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 Algorithm 7.

From Algorithm 7, 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.

Algorithm 7 SEP4: $SepPLri()$

Input:

- The Call Distribution $\mathcal{C} = \{ca_1, ..., ca_n\}$,
- The Destination Distribution $\mathcal{D} = \{de_1, ..., de_n\}$,
- Number of floors n

Output:

- Optimal parking floor,
- Average waiting time achieved.

begin
  Initialize $\mathcal{A}$ by selecting each action number of times
  Initialize $\mathcal{B}$ number of rewards received for each action
  $\mathcal{R}$ = Ratio of $\mathcal{B}$ to $\mathcal{A}$
  $parking\_distribution$ = Initial Equal Probability
  $parking\_floor$ = chosen randomly with equal probability
  for a number of $calls$ **do**
    $calling\_floor$ = passenger call($call\_distribution$)
    $waiting\_time$ = $|calling\_floor - parking\_floor|$
    $feedback$ = call ($EnvironmentFeedback()$)
    Update average waiting time($waiting\_time$)
    $destination\_floor$ = Call($destination\_distribution$)
    if $feedback = reward$ **then**
      call ($UpdateOnReward(parking\_distribution)$)
    else
      $DoNothing$
    end if
    Update($\mathcal{R}$)
    $parking\_floor$ = Pick Parking Floor($parking\_distribution$)
  end for
  $optimal\_parking\_floor$ = Converged Floor
  Print out average waiting time.
end

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 along 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 as done in Algorithm 5, 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 as in Algorithm 8. 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.

5.3.1 Simulation Results

To test SEP4, we used the same building settings as the previous solutions, SEP1, SEP2 and SEP3. Table 8 presents the results acquired from the simulations of the four distributions, and the corresponding results for SEP4.

In the first scenario, Exp, the results acquired showed an improvement in the performance with comparison to

Algorithm 8 SEP4: $UpdateOnReward()$

Input:

- The parking (actions) probabilities $\mathcal{P}(T) = \{p_1, ..., p_n\}$,
- Learning rate a,
- Reward estimates $\mathcal{R}$
- Index of maximum estimate m

Output:

- Updated parking probabilities $\mathcal{P}(t+1)$.

begin
  for each p_i in $\mathcal{P}$ **do**
    if $i = m$ **then**
      $p_i(t+1) = (1-a) * p_i(t) + a$
    else
      $p_i(t+1) = (1-a) * p_i(t)$
    end if
  end for
  return $\mathcal{P}$
end

SEP3. The AWT decreased even more as a result of a faster convergence to the optimal location and even closer results to the optimal values of SEP2 with an AWT of 0.85 down from 1.15.

Similarly, in the second scenario, $InvExp$, SEP4 displayed a superior performance with a similar decrease in AWT from 1.19 to 0.84. We can see the significant impact on the performance, and how the faster convergence led to almost a 30% decrease in the AWT when compared to SEP3.

In the third scenario, $Gaussian$, the results in SEP4 showed better results than SEP3, but the increase in the performance was not as significant as the previous scenarios with a decrease in the AWT from 2.43 to 2.37.

Similarly, for the final scenario, $Bimodal$, it was able to produce a lower AWT than SEP3 with an AWT of 2.53 down from 2.56 and converged to floor 8. One can observe how the effect of the pursuit concept in SEP4 helps to increase the performance and produce superior results in all scenarios, and how $\mathcal{C}$ affects this improvement.

Figure 7 displays the behaviour of how the AWT decreased over time for the first scenario, Exp for SEP4.

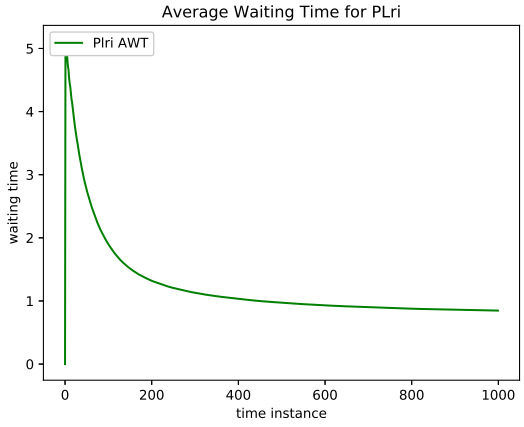


Figure 7: The Average Waiting Time for the PL_{RI} solution, SEP4, for an ensemble of 200 experiments for the case of the Exp distribution.

From Figure 8, we can see the difference in convergence speed between the SEP3 and SEP4, where SEP4 displays a better performance and faster convergence, attributed to incorporating the Pursuit concept.

Table 8: Simulation results for a 12-floor building for the policy SEP4 for an ensemble of 200 experiments. The results reported are the average waiting times for passengers in terms of number of floors for the elevator car to travel so as to reach the next call from the parked location.

$Dist$	Exp	$InvExp$	$Gaussian$	$Bimodal$
ca_1	0.59344245	0.00002978	0.02592411	0.01270883
ca_2	0.24127569	0.00007324	0.04518336	0.03860891
ca_3	0.09809538	0.00018013	0.07046903	0.07525205
ca_4	0.0398826	0.00044306	0.09834746	0.09457958
ca_5	0.01621506	0.00108974	0.12282111	0.08017525
ca_6	0.00659255	0.00268033	0.13725493	0.06220755
ca_7	0.00268033	0.00659255	0.13725493	0.08440493
ca_8	0.00108974	0.01621506	0.12282111	0.14232761
ca_9	0.00044306	0.0398826	0.09834746	0.17475873
ca_{10}	0.00018013	0.09809538	0.07046903	0.13967666
ca_{11}	0.00007324	0.24127569	0.04518336	0.07169797
ca_{12}	0.00002978	0.59344245	0.02592411	0.02360195
$BestFloor$	1	12	6	8
AWT	0.84734	0.839	2.36355	2.53346

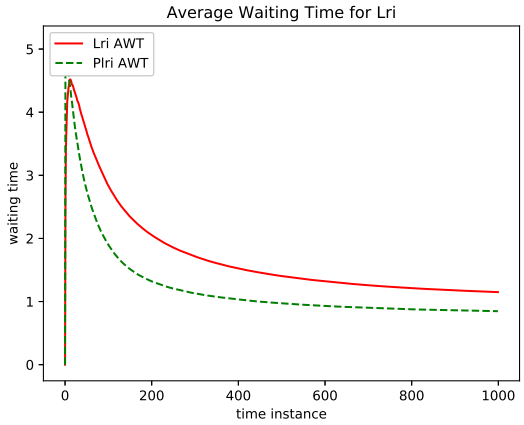


Figure 8: The Average Waiting Time for the L_{RI} solution, SEP3, vs. PL_{RI} solution, SEP4, for an ensemble of 200 experiments for the case of the Exp distribution.

6 Results & Discussion

We now comparatively discuss the simulations results obtained from the previous solutions. We will first discuss the results obtained in the previous simulations for the 12-story building setting, and then submit and discuss the simulation results for other building settings that were not discussed in our previous paper [2].

6.1 12-Story Building

In Table 9, we submit all the results of the four solutions that were discussed, namely, SEP1, SEP2, SEP3 and SEP4. 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 average waiting time was more than 50% and up to 80%. This policy serves as a lower bound for our benchmark, as no solution should be worse than this.

Table 9: 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.

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

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. On the other hand, our LA-based solutions were able to achieve a close-to-optimal AWT without the knowledge of $\mathcal{C}$.

SEP3 showed that it was better than SEP1 and recorded an AWT improvement in Exp of 78.56%, $InveExp$ of 77.85%, $Gaussian$ of 32.49% and $Bimodal$ of 30.9%. 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 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.

One result that we found 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, Exp and $InvExp$.

In Figure 9, we present the performance of each algorithm and how our proposed LA-based algorithms were able to achieve an average waiting time that is very close to the optimal solution, SEP2, and how it significantly outperforms SEP1.

Now we submit the results corresponding to the other building settings and discuss how our solutions are compared with SEP1 and SEP2.

6.2 8-Story Building

In Table 10, we submit all the results for the 8-story building setting. 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 average waiting time was more than 30% and up to 80%. As mentioned before, this policy serves as a lower bound for our benchmark, as no solution should be worse than this.

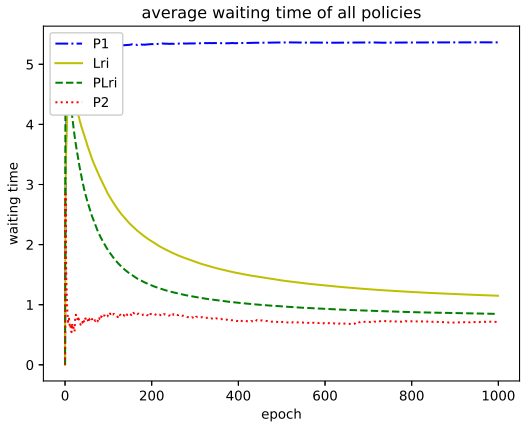


Figure 9: The Average Waiting Time for SEP1, SEP2, SEP3 and SEP4 for an ensemble of 200 experiments for the case of the Exp distribution, given in the graph as $P1$, $P2$, L_{RI}, PL_{RI} respectively.

Table 10: Simulation results for the previous and newly proposed solutions for the SEP for an ensemble of 200 experiments for the 8-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.

$Dist$	SEP1	SEP2	SEP3	SEP4
Exp	$(-, 3.4080)$	$(1, 0.673)$	$(2, 0.919)$	$(1, 0.783)$
$InvExp$	$(-, 3.4086)$	$(8, 0.760)$	$(7, 0.930)$	$(8, 0.782)$
$Gaussian$	$(-, 2.508)$	$(5, 1.747)$	$(5, 1.919)$	$(4, 1.889)$
$Bimodal$	$(-, 2.423)$	$(5, 1.473)$	$(6, 1.586)$	$(6, 1.557)$

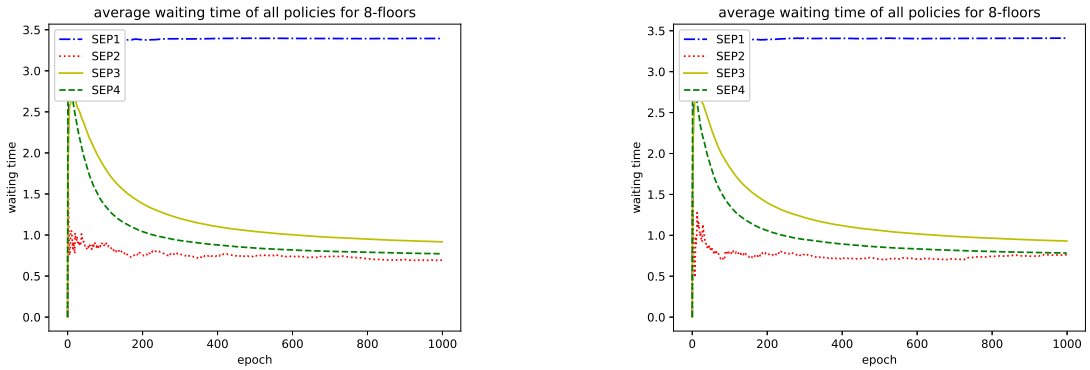
SEP2 was able to give us the optimal parking floors from the beginning, but it required the *a priori* knowledge of $\mathcal{C}$ for each floor. On the other hand, our LA-based solutions were able to achieve a close-to-optimal AWT without the knowledge of $\mathcal{C}$.

SEP3 showed that it was better than SEP1 and recorded an AWT improvement in Exp of 73.03%, $InveExp$ of 72.7%, $Gaussian$ of 23.49% and $Bimodal$ of 34.5%. 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 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 14.8% for the Exp, 15.9% for the $InvExp$, 1.56% for the $Gaussian$ and 1.8% for the $Bimodal$ distributions.

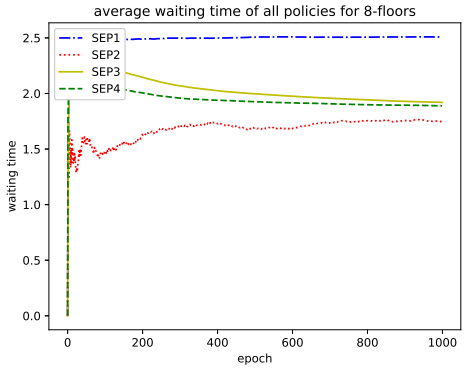
One result that we found is that the more equally distributed the calls are, for example for the $Gaussian$ distribution, the higher the AWT will be. On the other hand, the importance of having a good policy shines when the calling distribution is skewed toward a specific region or a specific floor.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

In Figure 10, we present the performance of each algorithm and how our proposed LA-based algorithms were able to achieve an average waiting time that is very close to the optimal solution, SEP2, and how it significantly outperforms SEP1 for all the different distributions.

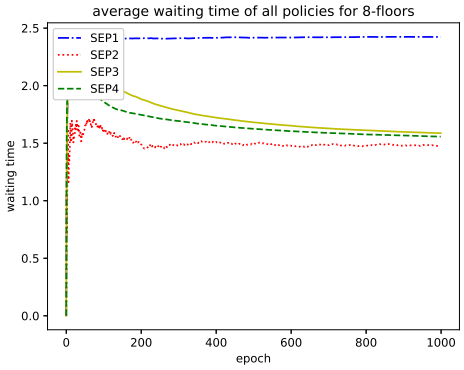


(a) Exp

(b) $InvExp$



(c) $Gaussian$



(d) $Bimodal$

Figure 10: The Average Waiting Time for SEP1, SEP2, SEP3 and SEP4 for an ensemble of 200 experiments for the case of the different distributions for 8-floor building setting.

6.3 16-Story Building

In Table 11, we submit all the results for the 16-story building setting. 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 average waiting time was more than 34% and up to 90%. As mentioned before, this policy serves as a lower bound for our benchmark, as no solution should be worse than this.

Table 11: Simulation results for the previous and newly proposed solutions for the SEP for an ensemble of 200 experiments for the 16-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.

$Dist$	SEP1	SEP2	SEP3	SEP4
Exp	$(-, 7.352)$	$(1, 0.705)$	$(2, 1.502)$	$(1, 0.897)$
$InvExp$	$(-, 7.342)$	$(16, 0.639)$	$(15, 1.557)$	$(16, 0.922)$
$Gaussian$	$(-, 4.583)$	$(9, 2.375)$	$(8, 2.669)$	$(8, 2.617)$
$Bimodal$	$(-, 4.894)$	$(11, 2.879)$	$(10, 3.213)$	$(11, 3.113)$

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. On the other hand, our LA-based solutions were able to achieve a close-to-optimal AWT without the knowledge of $\mathcal{C}$.

SEP3 showed that it was better than SEP1 and recorded an AWT improvement in Exp of 79.56%, $InveExp$ of 78.79%, $Gaussian$ of 41.76% and $Bimodal$ of 34.34%. Moreover, the results were close to the values achieved by SEP2.

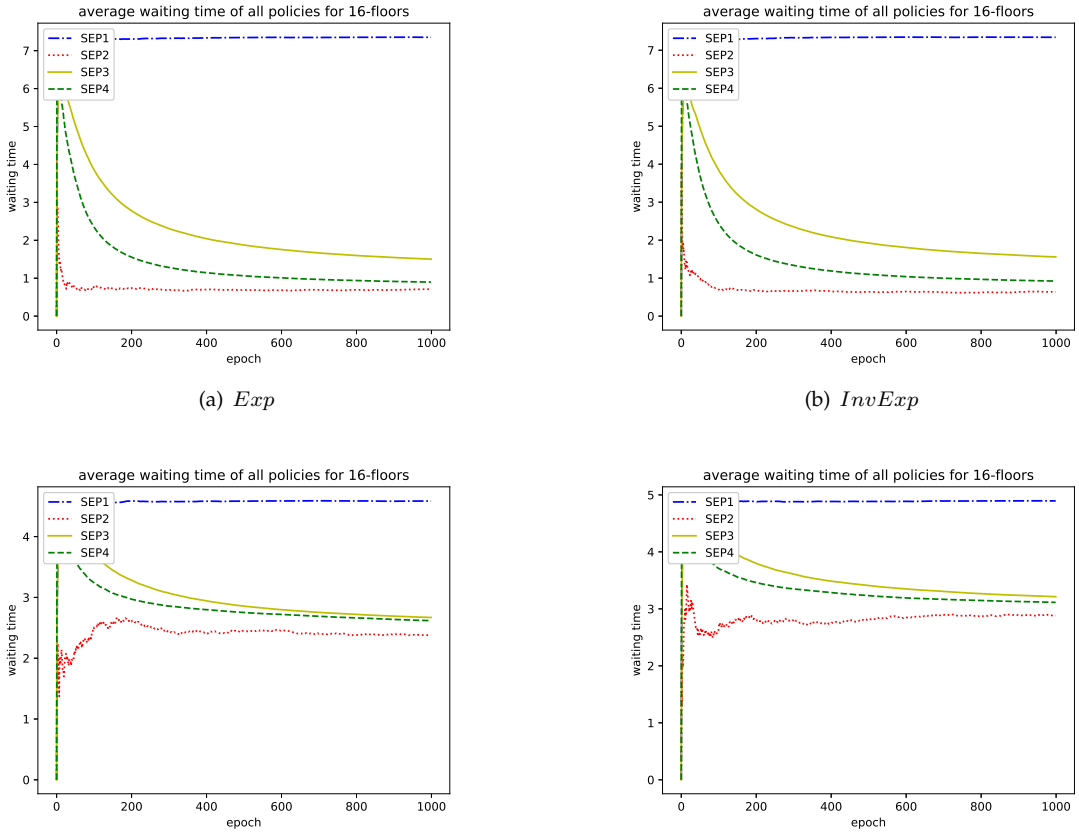
As in the previous settings, we attempted to improve SEP3 by proposing SEP4 that incorporated the Pursuit concept. Here, we achieved even better results than obtained for the SEP3. The algorithm helped the system to converge faster to the optimal locations. It also resulted in a better overall AWT. The improvements were 40.27% for the Exp, 40.7% for the $InvExp$, 1.98% for the $Gaussian$ and 3.11% for the $Bimodal$ distributions.

As in the previous results, we found 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, Exp and $InvExp$.

In Figure 11, we present the performance of each algorithm and how our proposed LA-based algorithms were able to achieve an average waiting time that is very close to the optimal solution, SEP2, and how it significantly outperforms SEP1.

6.4 20-Story Building

In Table 12, we submit all the results for the 20-story building setting. 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 average waiting time was more than 37% and up to 90%. As mentioned before, this policy serves as a lower bound for our benchmark, as no solution should be worse than this.



(a) Exp (b) $InvExp$

(c) $Gaussian$ (d) $Bimodal$

Figure 11: The Average Waiting Time for SEP1, SEP2, SEP3 and SEP4 for an ensemble of 200 experiments for the case of the different distributions for 16-floor building setting.

Table 12: Simulation results for the previous and newly proposed solutions for the SEP for an ensemble of 200 experiments for the 20-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.

$Dist$	SEP1	SEP2	SEP3	SEP4
Exp	$(-, 9.342)$	$(1, 0.676)$	$(2, 1.840)$	$(1, 0.938)$
$InvExp$	$(-, 9.341)$	$(20, 0.723)$	$(19, 1.921)$	$(20, 0.976)$
$Gaussian$	$(-, 5.538)$	$(11, 2.453)$	$(11, 2.805)$	$(10, 2.774)$
$Bimodal$	$(-, 5.867)$	$(13, 3.261)$	$(13, 3.687)$	$(13, 3.639)$

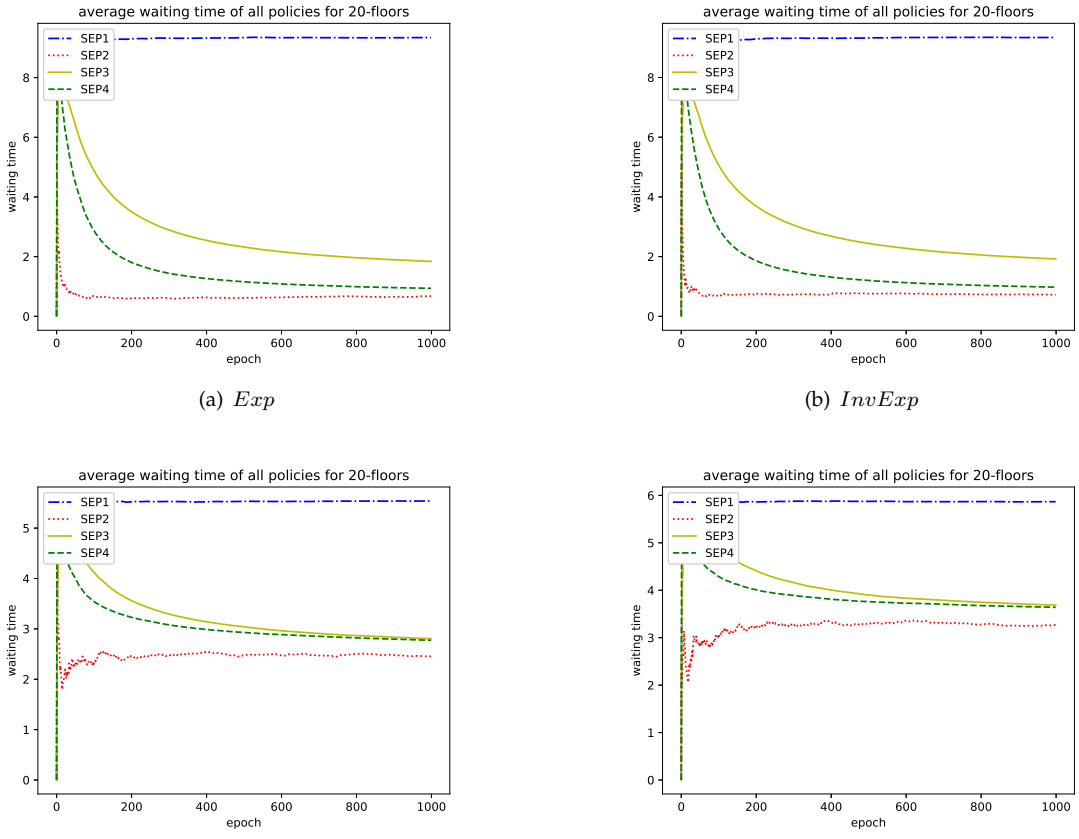
SEP2 was able to give us the optimal parking floors from the beginning, but it required the *a priori* knowledge of $\mathcal{C}$ for each floor. On the other hand, our LA-based solutions were able to achieve a close-to-optimal AWT without the knowledge of $\mathcal{C}$.

SEP3 showed that it was better than SEP1 and recorded an AWT improvement in Exp of 80.3%, $InveExp$ of 79.4%, $Gaussian$ of 49.3% and $Bimodal$ of 37.2%. Moreover, the results were close to the values achieved by SEP2.

As in the previous settings, we attempted to improve SEP3 by proposing SEP4 that incorporated the Pursuit concept. Here, we achieved even better results than 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 49.02% for the Exp, 49.2% for the $InvExp$, 1.1% for the $Gaussian$ and 1.3% for the $Bimodal$ distributions.

One more result that we found is that the more floors we have in the building, such system has more impact and better performance improvement.

In Figure 12, we present the performance of each algorithm and how our proposed LA-based algorithms were able to achieve an average waiting time that is very close to the optimal solution, SEP2, and how it significantly outperforms SEP1.



(a) Exp (b) $InvExp$

(c) $Gaussian$ (d) $Bimodal$

Figure 12: The Average Waiting Time for SEP1, SEP2, SEP3 and SEP4 for an ensemble of 200 experiments for the case of the different distributions for 20-floor building setting.

7 Conclusion

In this paper, we have concentrated on a host of problems with properties that are similar to those related to moving elevators within a building. These are referred to as Elevator-like Problems (ELPs). The paper has discussed their common properties, and we have resolved ELPs using the field of Learning Automata (LA). The simulation of these schemes has been done with modeling ELPs to be elevators working in "buildings" with "floors" etc. Our aim has been to minimize the Average Waiting Time (AWT) for the "passengers".

In this paper, we reviewed different parking policies that were used in various building settings and for previously-reported solutions. We discussed two different solutions in the single elevator scenario and tabulated the experimental results that correspond to the various simulations. We then introduced our L_{RI}-based solution, SEP3, and it achieved the optimal parking floor without knowing $\mathcal{C}$ *a priori*. We then improved on SEP3 to incorporate the Pursuit concept. Our second solution, SEP4, a PL_{RI}-based solution, out-performed SEP3.

We further reviewed the results of the different building settings, 8,16 and 20-floor buildings, and compared the results that demonstrated that our proposed solutions outperformed the SEP2 and produced AWT results that are close to the optimal results and was able to converge to the correct optimal parking locations.

Thus, we showed that our LA-based solutions performed better 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.

References

[1] O. Ghaleb and B.J. Ooommen. Novel Solutions and Applications to Elevator-like Problems. MCS Thesis, Carleton University, Ottawa, Canada. 2018.

[2] Omar Ghaleb and B. Oommen. *Learning Automata-Based Solutions to the Single Elevator Problem*, pages 439–450. 05 2019.

[3] J Kabudian, M R Meybodi, and M M Homayounpour. 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, pages 638–642. IEEE, 2004.

[4] M R Meybodi and H Beigy. New learning automata based algorithms for adaptation of backpropagation algorithm parameters. *International Journal of Neural Systems*, 12(01):45–67, 2002.

[5] S Misra and B J Oommen. GPSPA: A new adaptive algorithm for maintaining shortest path routing trees in stochastic networks. *International Journal of Communication Systems*, 17(10):963–984, 2004.

[6] S Molina and G Leguizam. An ACO Model for a Non-stationary Formulation of the Single Elevator Problem on. 7(1):45–51, 2007.

[7] K. S. Narendra and M. A. L. Thathachar. *Learning automata: an introduction*. Courier Corporation, 2012.

[8] M S Obaidat, G I Papadimitriou, A S Pomportsis, and H S Laskaridis. Learning automata-based bus arbitration for shared-medium ATM switches. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(6):815–820, 2002.

[9] B J Oommen and J K Lanctot. Discretized pursuit learning automata. *Systems, Man and Cybernetics, IEEE Transactions on*, 20(4):931–938, 1990.

[10] B J Oommen and T D Roberts. Continuous learning automata solutions to the capacity assignment problem. *IEEE Transactions on Computers*, 49(6):608–620, 2000.

[11] G I Papadimitriou and Pomportsis A S. Learning-automata-based TDMA protocols for broadcast communication systems with bursty traffic. *IEEE Communications Letters*, 4(3):107–109, 2000.

[12] Mahmut Parlar, Moosa Sharafali, and Jihong Ou. Optimal parking of idle elevators under myopic and state-dependent policies. *European Journal of Operational Research*, 170(3):863–886, 2006.

[13] F Seredyński. Distributed scheduling using simple learning machines. *European Journal of Operational Research*, 107(2):401–413, 1998.

[14] Jin Sun, Qianchuan Zhao, Peter B. Luh, and Mauro J. Atalla. Estimation of optimal elevator scheduling performance. *Proceedings - IEEE International Conference on Robotics and Automation*, 2006(May):1078–1083, 2006.

[15] Shunji Tanaka, Yasuyuki Innami, and Mituhiko Araki. 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*, 7:6274–6279, 2004.

[16] Shunji Tanaka, Yukihiro Uraguchi, and Mituhiko Araki. 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):550–573, 2005.

[17] Shunji Tanaka, Yukihiro Uraguchi, and Mituhiko Araki. 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):550–573, 2005.

[18] M A L Thathachar and P S Sastry. A class of rapidly converging algorithms for learning automata. In *IEEE Int. Conf. on Systems, Man and Cybernatics*. IEEE, 1984.

[19] M A L Thathachar and P S Sastry. A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SCM-15(1):168–175, 1985.

[20] M A L Thathachar and P S Sastry. Estimator algorithms for learning automata. In *Platinum Jubilee Conference on Systems and Signal Processing*, 1986.

[21] M Tsetlin. On behaviour of finite automata in random medium. *Avtomat. i Telemekh*, 22(10):1345–1354, 1961.

[22] V Varshavskii and I P Vorontsova. On the behavior of stochastic automata with a variable structure. *Avtomatika i Telemekhanika*, 24(3):353–360, 1963.

[23] Jingyang Xu and Tianke Feng. Single Elevator Scheduling Problem with Complete Information : An Exact Model using Mixed Integer Linear Programming. pages 2894–2899, 2016.