Comparative Study of Various Elevator Dispatching Algorithms

\*Note: Sub-titles are not captured in Xplore and should not be used

line 1: 1st Given Name Surname   
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 4th Given Name Surname  
line 2: *dept. name of organization*  
*(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCIDline 1: 2nd Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 5th Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCIDline 1: 3rd Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 6th Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

*Abstract*—This electronic document is a “live” template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. *\*CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract*. (*Abstract*)

Keywords—component, formatting, style, styling, insert (key words)

# Introduction (*Heading 1*)

This template, modified in MS Word 2007 and saved as a “Word 97-2003 Document” for the PC, provides authors with most of the formatting specifications needed for preparing electronic versions of their papers. All standard paper components have been specified for three reasons: (1) ease of use when formatting individual papers, (2) automatic compliance to electronic requirements that facilitate the concurrent or later production of electronic products, and (3) conformity of style throughout a conference proceedings. Margins, column widths, line spacing, and type styles are built-in; examples of the type styles are provided throughout this document and are identified in italic type, within parentheses, following the example. Some components, such as multi-leveled equations, graphics, and tables are not prescribed, although the various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

# Ease of Use

## Selecting a Template (Heading 2)

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the Microsoft Word, Letter file.

## Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

# BACKTRACKING

In this approach, we construct a state tree where each node is the floor representing the current location of the elevator. This is a recursive approach which uses the DFS(Depth First Search) technique to make a series of decisions in order to solve the problem while also keeping a track of what has been processed in previous steps. Once the initial solution has been found, the algorithm starts to backtrack the state tree. The backtracking proceeds until the final state is reached.

## Algorithm

Consider the example where we need to serve two requests – Floor 2 to Floor 4 & Floor 3 to Floor 0.

Each request consists of 3 parameters (Source, Destination & Ctr [Initially 0]). The Ctr has 3 values: 0 – Passenger not picked up, 1 – Passenger picked up & 2 – Passenger Dropped (i.e. Request Satisfied). We can explicitly specify the Floor where the lift starts serving the requests from (In this example we assume Ground Floor i.e. Floor 0).

The approach is a recursive approach where we construct a state tree where each node is the floor where the lift is currently at. We start the recursive call by passing the function with the starting Floor i.e. Floor 0. When we are generating new Nodes (From now on we will call them Floor) we go through all the requests and check the value of ctr.

For each request if the Ctr value is 0 then the new Floor generated has Floor number equal to the Source Floor. If the Ctr value if 1 then the new Floor generated has Floor number equal to the Destination Floor. (If Ctr value is 2 then we do nothing)

TABLE I

|  | Source | Destination | Ctr |
| --- | --- | --- | --- |
| Request 1 | Floor 2 | Floor 4 | 0 |
| Request 2 | Floor 3 | Floor 0 | 0 |

In this case both the requests have Ctr value 0 hence the new Floors generated have value of 2, 3 i.e. source floors. The approach of Backtracking follows DFS (Depth First Search), hence now we move onto the next Floor 2.

TABLE II

|  | Source | Destination | Ctr |
| --- | --- | --- | --- |
| Request 1 | Floor 2 | Floor 4 | 1 |
| Request 2 | Floor 3 | Floor 0 | 0 |

The Request 1’s Source is reached i.e. we have attended the passenger hence now we update the Ctr to 1.

Once a new Floor is generated we again loop through all the requests and check if the new Floor generated is the Destination for any of the requests. When we loop through – we also check whether their Ctr value is 1 because only if the Ctr value is 1 then we can drop them to their destination (Ctr value 1 means we have picked them up).

The new Floor generated is then recursively passed onto the same function to generate new floors i.e. keep on going through the rest of the floors (path to complete all the requests).

Now the request 1 has Ctr value 1 hence the new Floor generated from this Floor will have the Destination value i.e. 4, while the request 2’s Ctr value is 0 and hence new Floor generated because of it is the Source Floor i.e. 3

TABLE III

|  | Source | Destination | Ctr |
| --- | --- | --- | --- |
| Request 1 | Floor 2 | Floor 4 | 2 |
| Request 2 | Floor 3 | Floor 0 | 0 |

This process is followed recursively and hence the final tree is generated.

Below is the representation of the tree generated by the code for the example stated above.

After the entire tree is generated we find all the paths from source node and also their costs as shown in the above representation. The Costs are calculated by finding the difference between two adjacent nodes in the path and adding them together.

After the above calculations the lowest cost is found and that Path is the Optimal Path to serve all the Requests. In the above representation we can see there are two Optimal Paths to serve the required requests both having a Cost of 8 units.

## Results

The main objective of this network is to approximate the optimal Q-function that will satisfy the Bellman optimality equation.

## Conclusion

The complexity of the backtracking approach is high and proves to be very costly with an increase in parameters such as number of requests, number of floors and so on due to the increase in number of states. A solution that ensures faster updation of state changes is thus required to overcome this issue and to ensure a more efficient optimization process. This propels us in the direction of a Q-learning approach, an off policy reinforcement learning algorithm that seeks to find the best action to take given the current state.

# Q - Learning

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Terminologies

Below are some of the regularly used terminologies which will be used for the rest of the paper:

### Agent: The component of a Reinforcement learning problem which takes decisions and learns in order to maximise the rewards it is given by the environment.

### Environment: The part of the Reinforcement learning problem with which the agent interacts directly or indirectly. The environment changes with every action taken by the agent.

### Episode: The states falling between an initial state and a terminal state. Different episodes are completely independent of each other.

### Learning Rate: The tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.

### Policy: A mapping from a state s to the probabilities of selecting each possible action given that state. It is denoted as π (or sometimes π(a|s)).

### Reward: The numerical value received by the agent from the environment as a direct response to the actions taken by the agent.

### State: Every situation which the agent encounters in the environment is formally called a state.

## Model Based Q-Value Iteration

To understand this algorithm we first understand what a model is – So in Reinforcement Learning a Model is the perspective view or representation of the environment in which our Agent will be trained and used. The Model can be represented using the States, Actions, Probability Distributions and Rewards associated with the environment.

In this algorithm the Agent learns by interacting with the environment in discrete time steps and getting rewards which they try to maximise. The Agent is always in some state St which is part of the State Space of the Environment. After which the Agent selects a particular Action At according the Policy *π(St)* which it is following in a particular State.

After taking this action it transitions to a new State St+1 which depends on the stochastic dynamics of the Environment. The probablilty of ending up in a State St+1 from State St by taking an Action At is given by the probaility transition function f(St , At ,St+1). *This algorithm is so called because we know all about the probabilities of the Agent transitioning to a particular state when taking an action being present in the current state.* Once a transisiton is made the Agent receives some Reward rt = µ(St , At , St+1) from the Environment which measures the quality of the Action it took. This reward is an immediate reward and it says nothing about how good were the Agents actions in the past or future. This process is repeated by the Agent where it keeps on taking some Actions and transitions to new States thus accumulating Rewards.

The goal of the Agent is to maximize the expected reward given by:

In the above equation is the discount rate which determines how much do future rewards contribute to the current actions which the Agent takes. So the Agents goal is to maximize the Rewards in the future, but it only receives rewards based on the current transitions.

This can be achieved by learning a state-action value function. The value of the state-action pair (S, A) under the policy π, denoted by *Qπ (S, A)*, represents the expected return when starting in state S, taking action A and following policy π which is given as follows:

Eqn Goes Here

Where Eπ{.} denotes expectation under the stochastic dynamics f, given that the controller uses policy π. The optimal action value function Q∗ is defined as the maximum Q-function over all the policies:

Eqn Goes Here

Once Q∗ is known, an optimal policy (i. e., one that maximizes the return) can be found by a maximization over the action argument:

Eqn Goes Here

This is the greedy policy in Q∗. A central result in RL states that the optimal value function Q∗ is the unique solution of the Bellman optimality equation:

Eqn Goes Here

## Model less Q-Learning

This approach is also called off-policy Reinforcement Learning because it performs actions which do not follow a Policy as in case of *Model Based Q Value Iteration*. In this approach the Agent takes random actions so as to learn and eventually finds the best action to perform in a particular state.

The algorithmic aspect of this algorithm can be summarized by the equation given below where, xt, xt+1, and rt+1 are state and reward values observed by the controller while interacting with the environment at time t and t+1, and αt is the learning rate at time t :

Eqn Goes Here

For the above equation to function distinct Qt(xt,ut) values are stored in something called a Q-table which are used later on according to the update rule. The Agent tries to perform all possible actions in all states with non-zero probability which is part of the ɛ-Greedy approach. As, the equation reaches the optimality.

## Model less Q-Learning Implementation

For this implementation we have followed the paper *Reinforcement Learning for Elevator Control* by Xu Yuan, Lucian Busoniu and Robert Babuska (CITATION). In the Elevator Dispatching Model using the Q-learning approach we have simulated a single elevator system. The assumptions we have made include that the system experiences a down-peak traffic pattern which primarily occurs in commercial buildings and offices when people leave from work or at the start of the meal breaks when people use elevators to leave their office floor and go to the cafeteria downstairs. In order to reduce the complexity of the problem we have also assumed that at most one passenger will be waiting for the elevator car on each floor.

The downpeak elevator pattern is primarily characterized by a single destination floor (i.e, the ground or zeroth floor in our implementation) and multiple floors (non-zero floors) from where the elevator loads the passengers. The system implemented has following other variables:

### Number of floors: 5

### Elevator capacity: 4 passengers

### Elevator Velocity: 3 m/s (Used in state space only)

### Passenger Arrival Uniform Distribution: {0.6875, 0.0625, 0.09375, 0.09375, 0.0625} for events e = 0, 1, 2, 3, 4 (Here if e=0 is selected then no passenger arrives but if e>0 then passenger arrives at floor num ‘e’)

## Results

## Conclusion

Implementation of the Q-learning model gave rise to the realization that the model would be suitable only for environments with discrete and finite state and action spaces. A possible alternative to overcome this limitation and for extending Q-learning to richer environments could include implementation of the model by applying function approximators to learn the value function, taking states as inputs, instead of storing the full state-action table. However, since deep neural networks are powerful function approximators, it seems more logical to try to adapt them for this purpose.

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

**IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.**

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.