ECM1414 – Lift Controlled Assessment

PDF DOCUMENTATION

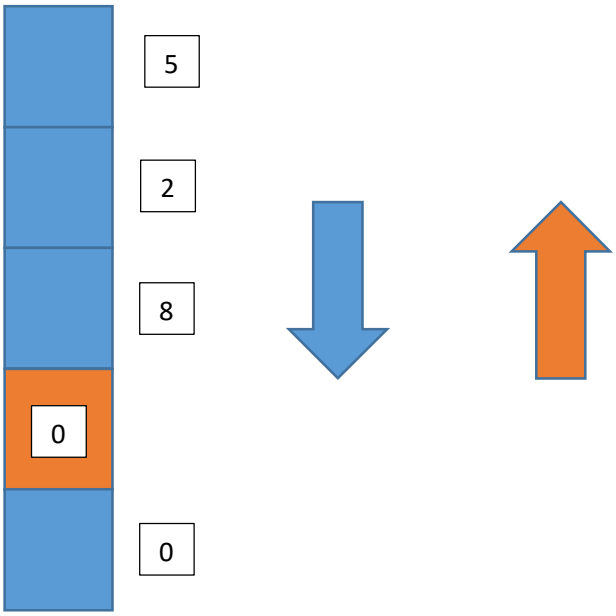
690030709

Student Number: 690030709

ECM1414 – Lift Algorithm

General description of my lift control system

For my control system, I worked on a theory of serving as many people at once as possible. For this, my lift will continue travelling in the same direction until it is empty and all the floors beyond its current floor are empty. For this to work, people will only get on the lift if it is currently travelling in the direction they want to go. If you were on floor 2 and the lift was travelling up, but you wanted to go to floor 0, you would not get on the lift. Take this diagram:



Each of the blue boxes is a floor, with the number of people on that floor indicated to the right. The orange box is the lift (the number of people in the lift indicated inside). The blue arrow indicates the direction the lift was travelling (i.e. it had just moved from floor 2 to floor 1). As there is no one in the lift, and no one to pick up on floor 0, there is no need for the lift to move in that direction. Therefore, my system will recognize this and move in the opposite direction, as indicated by the orange arrow.

Data structures and algorithms that I used

- Linked Lists – I used linked lists as they allow me to access any element within the list while also making the size of the list dynamic. They have been used to store groups of my custom object. My building object will have a linked list of floors in the building. Each floor will have a list of the people (person objects) that are currently on the floor and the lift will have a list of people currently within it.

Student Number: 690030709

Progress log

Below each date will be what I did on that day. Each of the subtitles under that will specify the class I was working on. I shall list the attributes and methods I added on each day, the specifics of what they do can be found in the Java Docstring.

08/02/20

Lift

I created my own custom class called lift responsible for holding all the methods and attributes required by the lift I am implementing. This class will be used by both control systems. I created the constructor for the lift class and gave the lift class the following attributes:

- currentFloor - int
- goingUp - boolean
- capacity - int
- currentNumPeople - int
- peopleinLift – person[]
- full - boolean
- building - Building

The following are the methods I created for lift:

- setCurrentFloor
- getCurrentFloor
- setGoingUp
- getGoingUp
- getNumPeople
- addPerson
- checkIfFull
- move
- checkIfChangeDirection

Person

I created my own custom class called person responsible for holding all the methods and attributes relating to a person. This class will be used by both control systems. I created the constructor for the person class and gave the person class the following attributes:

- currentFloor - int
- destFloor - int
- inLift - boolean

The following are the methods I created for person:

- setCurrentFloor
- setInLift
- getCurrentFloor
- getDestination
- getInLift

Floor

I created a custom class called floor to hold all the required attributes and methods of a floor object. This class is used by both control systems. I created the constructor and gave the floor class the following attributes:

Student Number: 690030709

- numPeople - int
- floorNum - int
- people – person[]
- changeDirection - boolean

The following is a method I created for floor:

- getDirectionChange

Building

I created a custom class called building to hold all the required attributes and methods of a building object. This class is used by both control systems. I created the constructor and gave the building class the following attributes:

- totalFloors - int
- floors – Floor[]

The following is a method I created for building:

- getFloor

Baseline

This is the custom class I create to run the mechanical control system of the lift. To start with I just initialized the lift and building, making sure that when I created the building it was filled with the required number of floors and people.

14/02/20

Lift

I altered the data structure used for peopleInLift. It was an array of objects (the type of which is Person), but I have now made it a linked list containing object of type Person. I also added the following methods:

- removePerson
- getPeopleInLift
- firstMove

Floor

I changed the data structure of people from an array to a linked list. The following was added as an attribute:

- empty – Boolean

The following are all methods that I added:

- getPeopleOnFloor
- getFloorNumber
- checkIfEmpty
- removePerson
- isEmpty
- getNumPeople

Building

The data structure of floors has been changed from an array to a linked list. I added the following attributes:

- numPeople – int

Student Number: 690030709

- empty – Boolean

The following are methods that I have also added:

- getNumPeople
- setNumPeople
- isEmpty

Baseline

I have set up the running conditions required for the simulation, which is that it shall carry on running until the building is empty (all people have been served and reached their destination). I have also implemented a checking system that removes people if they are in the lift and the lift has reached their destination floor.

25/02/20

Lift

I added the following attribute:

- floor – Floor

I also added the following methods:

- fillLift
- findFloor
- getActualFloor
- checkCurrentNum
- IsFull

Building

I added the following method:

- removePeople

Baseline

I initialised using the filling lift system. People can now get in the lift if they are on the same floor as the lift and the lift isn't full. If the lift is full or the floor is empty the lift carries onto the next floor. This is one of the final parts of the baseline control system, next I'm moving onto testing to make sure it works as intended.

27/02/20

Baseline

While testing and debugging the mechanical control system, the building would sometimes not empty. There would be one individual who was still in the building but not on a floor or in the lift. This is an obvious bug that I tried to fix during this session.

29/02/20

Lift

This was just a general functionality update, I altered most of the functions with tweaks that reduce the total line number and remove redundant code.

Student Number: 690030709

Floor

I've added the following attribute:

- totalFloors – int

Building

Like the lift class, this was a general functionality update that removed redundant code and increase functionality.

Baseline

The issue with the simulation not finishing (described in the previous log) has been solved. The issue had occurred in my removing people function, the number of people in the building was not being decremented correctly. Changing the function that removes people solved this issue.

05/03/20

Lift

To allow the lift class to be functional for both control systems, I have renamed my move and firstMove functions to baselineMove and baselineFirstMove respectively.

Baseline

This will be my final functionality update for baseline as it now works as intended, I may make further changes to allow it to work with graphics in the future. I have removed the main function from baseline so that it can be called and is no longer static. It has been renamed to run and takes the number of floors and people wanted in the simulation as arguments.

12/03/20

Lift

I reversed the naming of baselineFirstMove back to firstMove as, no matter what happens after the first move, the lift must always go up first. I also renamed fillLift to baselineFillLift (as the improved control system will use its own filling mechanism). I have added he functions responsible for filling the lift and moving the lift in the improved control system. They are called improvedFillLift and improvedMove respectively. People shall only get on the lift if the lift is travelling in the same direction they are.

Building

I have added the following function:

- getTotalFloors

Baseline

There are no longer any static functions within the Baseline class. The section of my code responsible for removing people from the lift has been moved into its own function called removeFromLift.

Improved

Using the mechanical control system as a basis I started creating my control system. The lift will be able to change direction at any floor and I have begun creating a function called getPriority that works out whether the lift should move up or down. The basis for all the function I am using are from the baseline class (with a few tweaks).

Student Number: 690030709

15/03/20

Improved

I have now implemented a system to getPriority that takes the number of moves people have been waiting into account. The person with the highest wait time in the lift gets their priority doubled to make it more likely that the lift will go to their floor. For this to work I created two functions that increment everyone's wait times. When an individual gets in the lift, they are set a new wait time (which will be the amount of time they have been in the lift for). Each floor will be assigned a priority based on the collective waiting times of people. The lift will then travel in the direction with the greatest priority.

Person

I've added the following attributes:

- floorMoves – int
- movesInLift – int

I also added the following methods:

- incrementMovesInLift
- getMovesInLift
- incrementFloorMoves
- getFloorMoves

16/03/20

Improved

I have noticed that the lift is getting stuck on certain floors when the priority is equal for both directions. To attempt to resolve this I have created two 'forced' move functions, one for up and the other for down. If the floor with the highest priority has a greater priority than all the floors in the opposite direction, the lift will be forced to travel to that floor.

Lift

The two forced functions are within the lift class. They are called forcedUp and forcedDown.

17/03/20

Improved

I have decided to scrap this control system as my previous attempts to solve it have not worked. The control system does finish the simulation; however, it takes double the amount of moves that the mechanical control system does. Therefore, I will now be creating and new control system. To make my 2 attempts distinguishable this new control system will be under ImporvedMK2 from now on.

ImprovedMK2

In this attempt I am going to get the lift to fill itself in an efficient way. It shall move to the nearest floor until it is full, then travel to the furthest destination and repeat.

Lift

To make the move functions for Improved and ImprovedMK2 distinguishable, the new move function is called improvedMove2.

Student Number: 690030709

20/03/20

LiftAlgorithm

To try and reduce code redundancy I have created an abstract class that will store all the methods and attributes used by both control systems. The methods are:

- removeFromLift
- removePeopleFromBuilding

The attributes are:

- l - Lift
- b - Building

ImprovedMK2

The methods and attributes that are now in LiftAlgorithm have been removed, and ImprovedMK2 now extends LiftAlgorithm. I also managed to complete the movement function for this control system, the next step shall be testing whether it is more efficient.

Baseline

Same as with ImprovedMK2.

22/03/20

LiftAlgorithm

I've added a saving mechanism in a method called saveResult. It takes the number of floors in the simulation and the amount of move it took to complete as arguments. These are then saved to a csv file, if the csv already exists the results are appended to the end without overwriting existing results (with help from reference 4). These results will be used to create graphs to demonstrate the efficiency of both control systems, and to act as a comparison.

ImprovedMK2

This attempt has also had to be scrapped as, again, it is less efficient than the mechanical control system. It is a better attempted than Improved as it is only about 50% worse.

ImprovedMK3

My next attempt at, hopefully, a control system that is more efficient than the mechanical control system. This system will work on the same basis as the baseline, the lift will only change direction at the boundary floors. The difference will come when people get into the lift, they only get in when their destination is in the direction the lift is travelling.

Lift

improvedMove and improvedMove2 have been removed from my lift class as there are no longer of any use. improvedFillLift is going to remain as it is going to be used by ImprovedMK3.

Person

The wait time attributes and methods used by my first attempt have been deleted as they are no longer of any use. For reference, these were the methods and attributes listed under 'Person' for 15/03/20.

Student Number: 690030709

27/03/20

DisplayGraphics

I have begun my graphics for the project. This will be the class containing all of the graphics, although I am planning to have the updates occur within my lift class, so it can happen in tandem with the movement of the lift. This class will extend the JPanel class (information gathered from reference 5). I am hoping to use the built-in paint and repaint functions to make my lift graphics run like an animation.

29/03/20

DisplayGraphics

I have overridden the base paint function (although the base function is going to run at the beginning of the overridden function). The DisplayGraphics object needs to be added to a JFrame to be viewed (see reference 6), so I added a static main method to the DisplayGraphics class to run everything.

Lift

I have created a function called updateGraphics which will run the repaint function for the initialised DisplayGraphics object.

LiftAlgorithm

I have added two functions, returnLift and returnBuilding, which allow the building and lift specific to the initialised simulation to be accessed by DisplayGraphics. This should hopefully allow me to run the graphics in tandem with the simulation.

29/03/20

DisplayGraphics

I have created and implemented the simulation running with both my baseline and improved control systems. There are no buttons or input systems yet, I am hoping to implement this later.

Lift

The updateGraphics function is going to be run each time the lift changes a floor, so it is currently within my baselineMove and firstMove functions.

01/04/20

DisplayGraphics

I have added buttons to my simulation with a menu frame that demonstrates them. One will be for the baseline while the other for my improved control system. Currently the simulation does not run when the button is pressed. Hopefully, I will be able to find out why.

Lift

The updateGraphics function is going to be run each time the lift changes a floor, so it is currently within my baselineMove and firstMove functions.

ImprovedMK3

Once again, this method has proven to be less efficient than the mechanical control system. So I am once again back to the drawing boards to try and create and efficient algorithm. My next attempt will be recorded under ImprovedMK4.

Student Number: 690030709

ImprovedMK4

My next algorithm is going to try to follow the idea that the lift shall carry on moving until it is both empty and there are no more customers to serve beyond that point. I have begun the implementation of this method. This class will also inherit LiftAlgorithm.

06/04/20

DisplayGraphics

I have found that there is a compatibility issue with using JButtons and painting a JPanel (see reference 7). For this reason, I am just going to run the simulation from the main method and have the user use the command line to input what they want to customise the simulation. The user can select how many floors they want (any more than 10 require can only be run in the background); the number of people and which simulation they want and whether they want to animate it or run it in the background.

Lift

The lift class now has the following attribute:

- animate – Boolean

This will run the updateGraphics method if true. I have finished implementing and testing this system, it is indeed more efficient and has been tested.

References

[1] Placing a JButton in the desired location - https://stackoverflow.com/questions/3195666/how-to-place-a-jbutton-at-a-desired-location-in-a-jframe-using-java

[2] Appending to a csv rather than overwritting - https://stackoverflow.com/questions/8256389/appending-to-the-last-line-of-csv-file-in-java

[3] How to use a JButton - https://docs.oracle.com/javase/tutorial/uiswing/components/button.html

[4] JButton Documentation - https://docs.oracle.com/javase/7/docs/api/javax/swing/JButton.html

[5] JPanel Documentation - https://docs.oracle.com/javase/10/docs/api/javax/swing/JPanel.html

[6] JFrame Documentation - https://docs.oracle.com/javase/10/docs/api/javax/swing/JFrame.html

[7] JButton and painting JPanel aren't compatible - https://bytes.com/topic/java/answers/16314-repainting-within-actionlistener-doesnt-work

[8] Drawing string inside rectangle for animation - https://stackoverflow.com/questions/8279975/how-to-draw-a-string-inside-a-filled-rectangle

[9] Inputting data to the system, through the command line - http://www.scit.wlv.ac.uk/~in8297/CP4044/workshops/w03.html

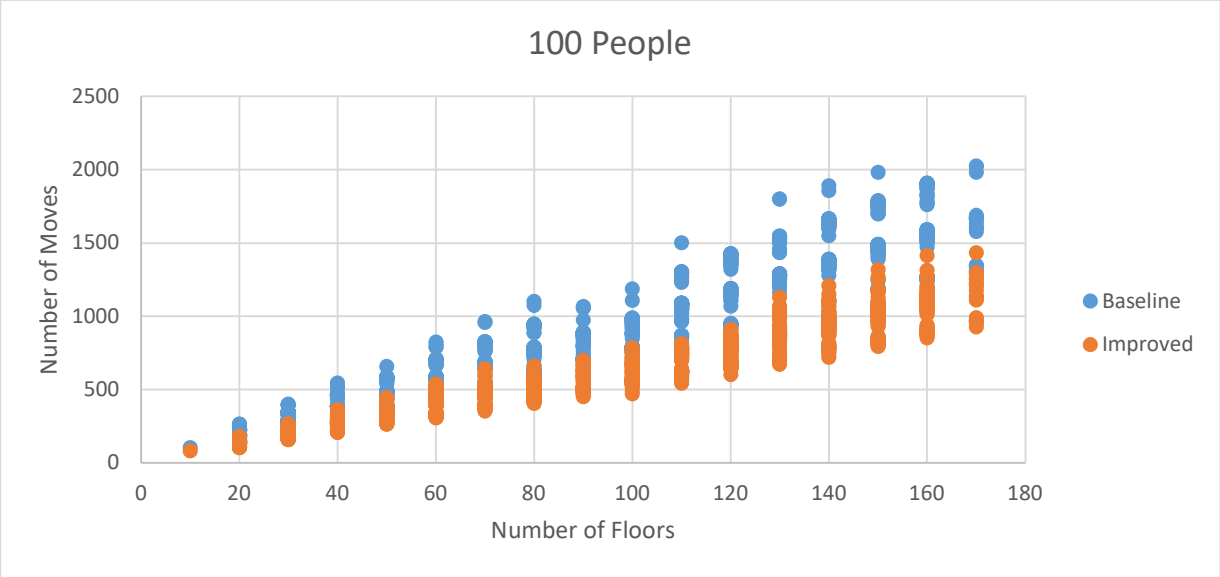
Student Number: 690030709

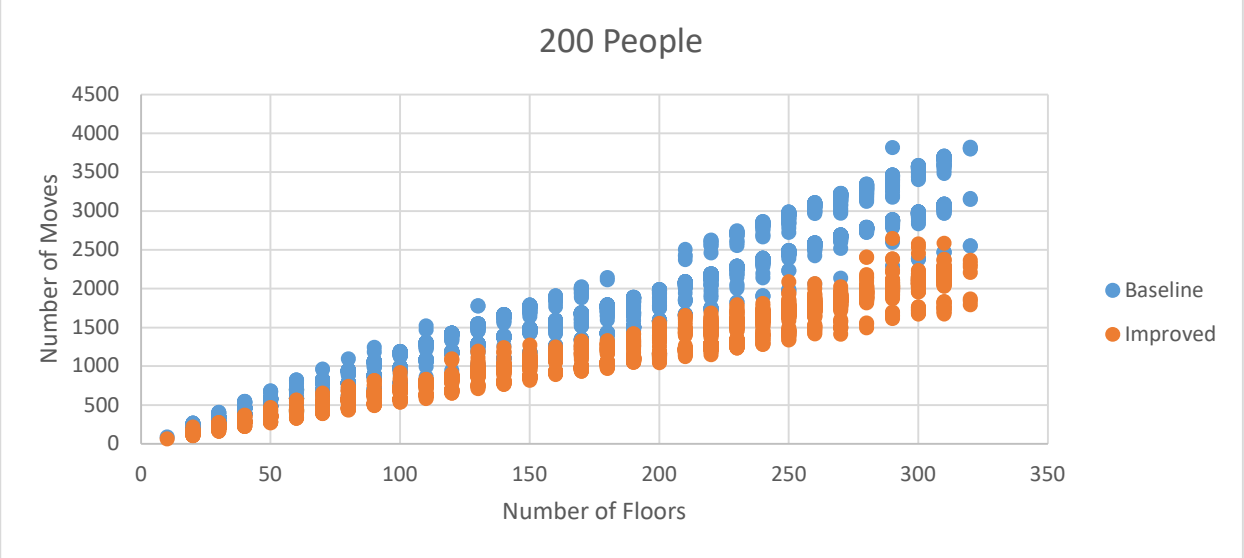
Performance Analysis & Graphs

For both system I have used a 10% rule for the lift's capacity, the capacity of the lift will always be 10% of people in the building. This is to try and allow the simulation to run faster, while still allowing comparability between my improved control system and the mechanical control system.

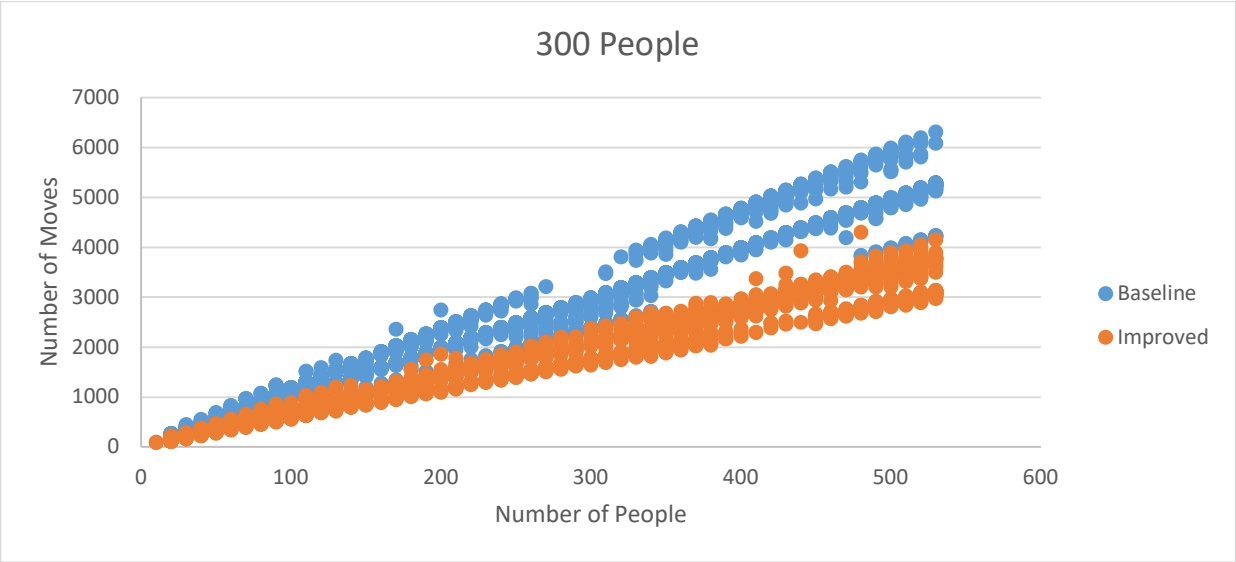
The variable that I have decided to measure and optimise is the total moves it takes for the lift to finish the simulation, where a move is a change in the lift's floor. As all people are generated at the start of the simulation, this variable will be equal to the longest wait time (both waiting for the lift and waiting to reach their destination). This would be a complaint of customers for a lift, this individual would not be interested in the average wait time, only their wait time. If I can reduce their maximum wait time this guarantees that the customer will not have to wait longer than this maximum wait time.

See the graphs below to see a comparison of the 2 control systems.





Student Number: 690030709



As the graphs show, my improved control system is much more efficient than the mechanical baseline. This is especially true the more floors there are in the simulation.

Video

The video can be found by following this link:
https://youtu.be/TxNHDECU6KY