KTH Computer Science
and Communication

Elevator Control Strategies

Simulating different algorithms to find the most efficient strategy

By

Frederick Ceder, fceder@kth.se

Alexandra Nordin, alnordin@kth.se

Course: Degree Project in Computer Science, First level (DD143X)

Supervisor: Johan Boye

Faculty: Computer Science and Communication, KTH

Stockholm, Sweden

30th May, 2013

Abstract

This paper investigates the efficiency of known elevator control strategies by simulating these in an own made apartment simulator. Efficiency will be determined by the lowest product of the energy consumption (Watt/second), average waiting time, average transfer time and the maximum waiting time of a passenger, which is the output by the simulator. The apartment simulator will simulate the elevator behavior, according to a respective control strategy, and the passenger flow on each respective floor in a specific test scenario. In this test scenario, passengers always travel either to the ground floor or to their respective home floor to simulate an apartment complex on a workday.

The outcome of the investigation was that a control strategy that would prioritize elevator orders, i.e. calls made from inside the elevator, remember calls and collect passengers that are on route was the most efficient, both in terms of low energy consumption and passenger satisfaction (low transit and waiting times).

Table of Contents

1. Introduction

Development of energy efficient technologies is of high importance in today’s society and should be considered due to moral and financial aspects. The usage of different smart strategies and algorithms in different technologies can lead to low energy efficiencies and is therefore of high importance in the world today. One way of reducing energy consumption in apartment buildings is to have an elevator which operates using a smart and energy efficient control strategy, but at the same time tries to minimize the waiting time for passengers. This is the topic of this paper.

1.1 Problem statement

The goal of this research is to investigate different elevator control strategies and to find the most efficient, in terms of energy consumption and waiting time, for a certain apartment building by programming a simulator in which different control strategies can be implemented. The goal is to make the simulator as modular as possible so that many different scenarios can be simulated and to find the best control strategy.

1.2 Hypothesis

A control strategy which collects passengers going to the same direction will be the most efficient in time and energy. This is because when collecting all calls in one direction the travel distance will be reduced and therefore also the travelling time. Presumably this will lead to lower energy consumption and a serving of many passengers in one trip, which will lead to lower passenger waiting time.

2. Background

2.1 Types of elevators

There are three commonly used elevators today: Hydraulic elevators, gearless traction elevators and geared traction elevators. Traction elevators are pulled up and down with a rope connected to a motor with a counterweight. Hydraulic elevators are used today in both passenger and freight services in buildings from two to six stories high and has a speed from 0.125 to 1.0 mps(meters per second). Geared and gearless elevators work similarly, with the only difference being that a gearbox is built in between the motor and the sheave in a geared elevator (Strakosch & Carporale, 2010, pp. 3-11)**.**In this essay the focus will be on gearless non-hydraulic traction elevators.

2.2 Elevator control strategies

2.2.1 Automatic operation

When electrical elevators were introduced, the most common and simple control strategy was the so called "Automatic Operation". This system only requires one button on each floor, which, when pressed, sends a call to the elevator to come to that specific floor. The elevator car is equipped with one button per floor, so that the passenger can press the respective button for the desired floor. This system can only handle one trip at a time, making it quite inefficient if there are passengers wanting to travel in the same direction since one passenger will have to finish his trip first. As buildings started to become taller the demand on more controlled strategies increased (Strakosch & Carporale, 2010, pp. 159-160). This control strategy will be tested and implemented in the simulator built for this research.

2.2.2 Elevator Algorithm

In order for elevators to be able to travel at a higher speed it was necessary for the elevator to know in advance when to start decelerating. In this case the floor, at which the passenger wishes to stop, is registered by the elevator operator so that the decrease in speed wouldn't be too uncomfortable for the passengers. Due to human constraints the acceleration of an elevator should preferably not exceed 1,5 m/s^2 (Barney & dos Santos, 1985, p. 3).

Since elevator traffic kept on increasing the control system "Collective Operation" was introduced. These systems most commonly include an up- and down-button on each floor so the passenger can inform the operating system in which direction she wishes to travel. Collective Operation implies collecting and storing all calls in one direction, then reversing the elevator and collecting all the calls in the other direction (Strakosch & Carporale, 2010, p. 160). This algorithm of elevator control is called the elevator algorithm (Appendix 9.2.1) and will be implemented and tested in this research.

2.2.3 Computerized Control

The most dynamic changes in elevator control started when the microprocessor was introduced with programmable features, making it possible to use computers for running the elevator control logic. This made it possible to store more information at a much faster rate. It is this usage of computers that allows different control strategies to be implemented in an easy way in the same elevator system. The usage of such a control system requires different types of input information to function. The elevator calls generated by passengers is considered as the primary input (Strakosch & Carporale, 2010, p. 165)as it is because of this information that the elevator acts in a certain way to fulfill its purpose. Other input needed is information about the elevator itself, i.e. current location, direction, speed and state (acceleration, constant speed, etc.) and also, for some systems, the time of day.

As more advanced systems were developed, elevator control systems started using strategies which allowed them to learn, reason and solve problems. The most common priority for implementing self-learning algorithms in these control systems is to minimize the engine cost (Strakosch & Carporale, 2010, p. 172), but there are of course other parameters that should be considered to be optimized, like for example the average passenger waiting time. In a control system these parameters can be weighed differently by importance by which the programmer chooses. In the case of this thesis, the energy cost is to be considered as most important since the goal is to minimize the power consumption by the entire system. The average waiting time, the maximum waiting time of an individual and average travelling time will also be measured and taken into consideration when comparing results.

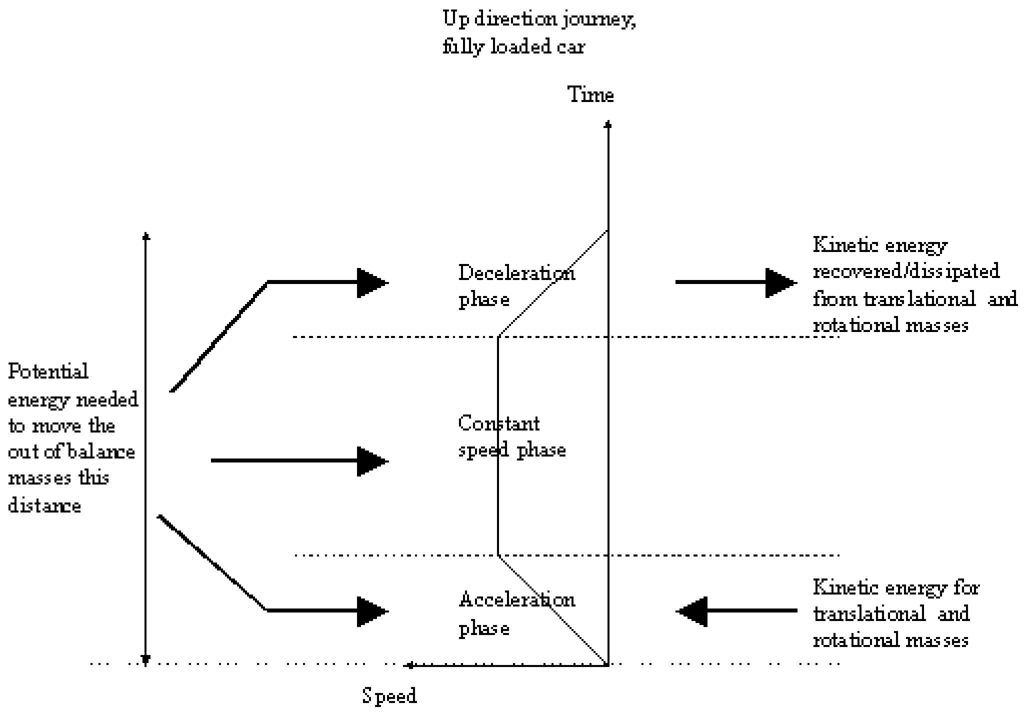
2.3 Energy consumption model

In an ideal world, without energy losses or friction, where passengers use a counterweighted traction elevator to travel to a certain floor and to travel back down, there wouldn't be any energy losses in the long run. The idea is that an entity resting at ground level will require an elevator system to do a specific amount of work in order to move the elevator to floor at height h. The entity will store the potential energy that was used to make the elevator move the person up and likewise give it back when going back down again. (Peters, et al., 2004)

If one imagines two masses M1 and M2 that have the same mass for the sake of simplicity. Then, ideally, if neglecting all external forces, these two masses would balance each other out and not move. If M1 would carry a person up to a floor it would require the system to do a specific amount of work. If the person was to return to the elevator to go back down, then the weight of the person would force M1 down, requiring no energy from the system, which supports the idea of an entity "giving back the energy" to the system. Ideally, the system would not consume any energy in the long run. However, in reality this doesn't fully apply. For example, if the weight if an elevator cart travelling downwards is greater than its counterweight the elevator engine would have to apply energy to the system so the cart doesn't travel too fast.

2.3.1 Energy calculation model

If we implement this concept, that there are no external forces, except gravity, acting on our elevator system, yet with constant acceleration and deceleration of the elevator cart, it can be interpreted as seen in figure 1.



Up direction journey, fully loaded car
Time
Kinetic energy recovered/dissipated from translational and rotational masses
Deceleration phase
Potential energy needed to move the out of balance masses this distance
Constant speed phase
Acceleration phase
Kinetic energy for translational and rotational masses
Speed

Figure 1 - (Peters, et al., 2004)

The idea is to divide the calculation of energy into to three major parts, where these can be classed into the following states:

1. Acceleration
    - Elevator engine will exert force on cart so that it accelerates.
2. Constant speed
    - Elevator engine will exert force on cart so that it goes at constant speed.
3. Deceleration

- Elevator engine will exert force on cart so that it decelerates.

Work done is related to the kinetic energy of moving objects. Thus, in order to calculate the energy of a moving Cart one can use the relationship described in Equation 1 between kinetic energy (K.E.), velocity and mass.

$$K.E. = \frac{1}{2}mv^2$$

Equation 1

It can also be described in terms of force such as in Equation 2.

$$K.E. = F \times s$$

Equation 2

Equation 2_allows us to calculate the work required to push a cart a certain distance (s). Thus, the force exerted on a moving mass must be calculated in order to be able to calculate the work done. This calculation model assumes a traction elevator, which means that a counter mass is always working against or with the cart, depending on the cart direction. To be able to calculate the work done when an elevator engine pulls a cart one must calculate the effective gravity force on the cart due to gravity and the pull/push of the counter mass, as seen in Equation 3. F_{system} in Equation 4 is the the force exerted by the elevator engine, whose magnitude is affected by the effective gravity force, which will push the elevator cart with a certain resulting force making it accelerate at a certain speed or move it at a uniform velocity (see Equation 4).

$$F_{gravity} = |m_{Cart} - m_{CounterMass}| * a_{gravity}$$

Equation 3

$$F_{result} = F_{gravity} + F_{system}$$

Equation 4

Note: See 9.1 Equations

Important to note is that all forces should be relative to the elevator cart and a negative or positive force will indicate the direction of the force – either down or up.

2.4 Passenger flow

Assuming that most of the residents in an apartment building work or go to school during the weekdays, it is important to simulate a passenger flow which takes this into consideration. A realistic scenario would be that the elevator traffic going down to the ground floor would peak during morning rush hour, whereas the traffic going up back from the ground floor would peak during the afternoon (Susi, et al., 2004, p. 5). Traffic from each floor can differ from each other depending. If a person who lives on the first floor is leaving the building, the chances of this person using the stairs is higher than for someone living on the sixth floor. When describing the process of passenger arrivals on each floor in an apartment building it is generally accepted to use a Poisson process such as in Equation 5.

$$P(n) = \frac{(\lambda T)^n}{n!} e^{-\lambda T}$$

Equation 5

$P(n)$ is the probability of n calls being registered within the time interval T and λ is the intensity of the passenger flow (Strakosch & Carporale, 2010, p. 46). For generating higher or lower passenger flow on different floors the intensity can be changed.

3. Simulator

For carrying out research on different elevator control strategies a simulator program, written in Java, was developed to generate results. The following subsections describe the requirements on the simulator, its general structure, implemented control strategies and the simulation of passenger flow.

3.1 Description

The whole simulator consists of many components, such as the elevator system, the different floors and all the passengers, that together create an apartment complex. The simulator will simulate a collection of residents living on their respective floor, using the elevator to travel to the ground level or up to their home from the ground floor.

The state of the entire simulator program is dependent on the states of every subcomponent, which is updated every time step simulating one second.

The following subsections will define important subcomponents and their states, which are constantly updated, that are required to understand the fundamental concept of this simulator.

3.1.1 Passenger

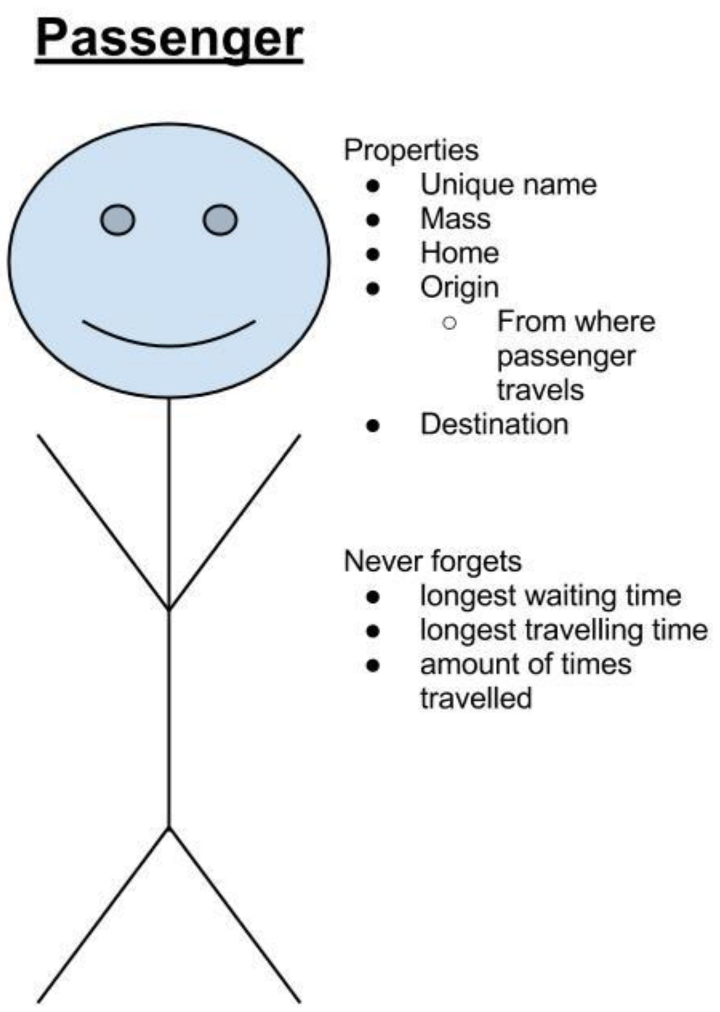


Figure 2 – Passenger entity

Figure 2 – Passenger entity gives a good overview of the different properties, or states, that a passenger has. Unique name, mass and the passengers home state are values that will stay constant throughout the whole simulation, whereas the rest can change during the simulation. Longest waiting time will not exceed a waiting tolerance time, which simply means if the passenger reaches the tolerance level then the passenger will take the stairs instead of the elevator.

The purpose of the passenger is to live as a resident on a specific floor and, if queuing in the waiting queue of the specific floor, request an elevator and travel in it to another floor (see 3.1.2 Floor and Figure 4 - Elevator system).

3.1.2 Floor

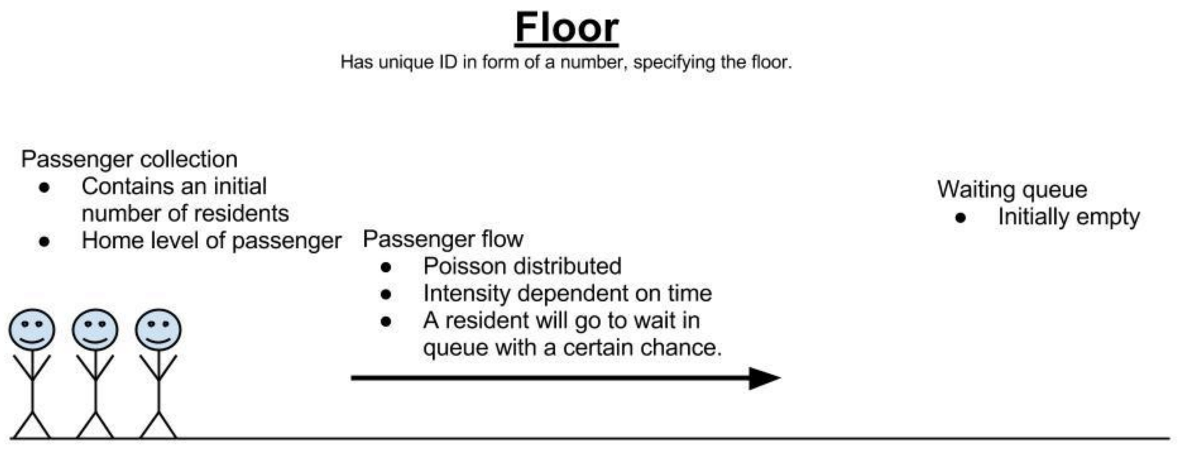

Floor
Has unique ID in form of a number, specifying the floor.
Passenger collection
● Contains an initial number of residents
● Home level of passenger
Passenger flow
● Poisson distributed
● Intensity dependent on time
● A resident will go to wait in queue with a certain chance.
Waiting queue
● Initially empty

Figure 3 - Floor

Apart from having a unique ID, which is constant throughout the simulation run, a floor will have three important state variables (see figure Figure 3 - Floor) that will change throughout the simulation. The first is a collection of passengers (see 3.1.1 Passenger) that exists on each floor. If passengers travel to this floor the size of this collection will increase and if they travel from this floor it will decrease. The waiting queue is the second state variable of the floor, where passengers wait for an elevator to come and pick them up. The third state variable is the current intensity of the passenger flow to the waiting queue of the floor, which influences the probability that a passenger will go to the waiting queue.

3.1.3 Elevator System

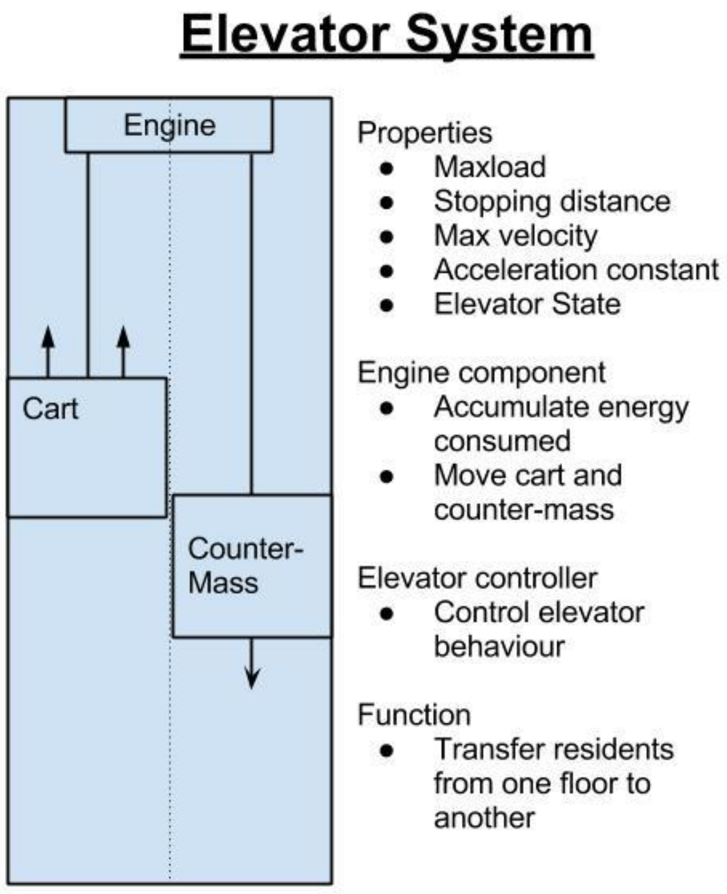


Figure 4 - Elevator system

The elevator system's (illustrated in Figure 4 - Elevator system) function is to transfer passengers (see 3.1.1 Passenger) from one floor (see 3.1.2 Floor) to another. The illustration also shows the relationship between the cart and the counter mass and also the role of the engine. The elevator engine will either pull or push the cart down, which will have the reverse effect on the counter-mass. Both the cart and counter-mass will have several states (see 9.6 Documentation) changing throughout the simulation which includes a lot of kinematic values (see 9.6 Documentation), yet also from the carts perspective its current mass load and similar.

Generally, the elevator system can be considered as a specific set of general properties that are constant throughout the simulation and states that can change during the course of the simulation. The elevator system's elevator state is probably the most significant one, as this determines how the elevator should behave for the current time step (see Figure 1 -).

The elevator controller (see 3.4 Elevator controller) is responsible for the elevator's state changes as it actually controls the elevator's behavior according to requests made to the elevator to pick up passengers or transport them to a specific floor.

3.1.4 Apartment complex

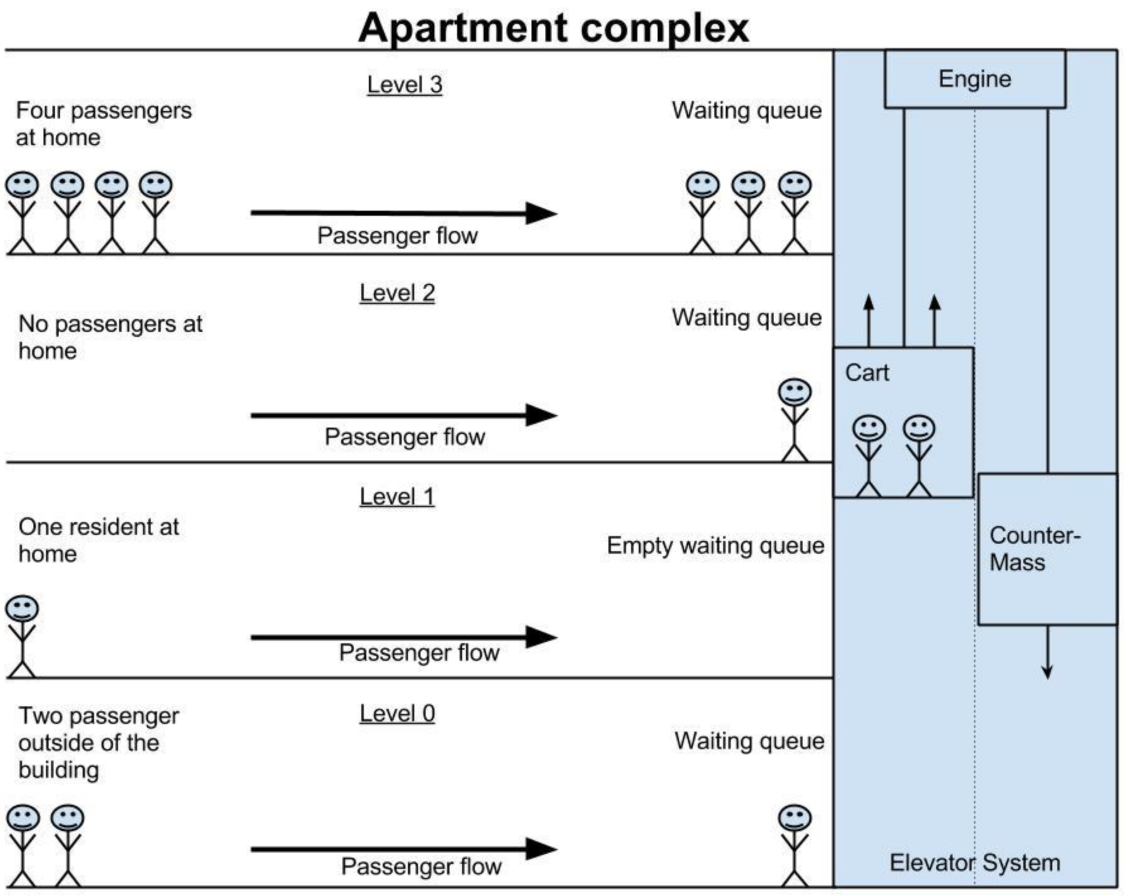


Figure 5 - Apartment complex

The apartment complex does not contain any particular states except for being a wrapper of the components mentioned in 3.1.1 Passenger, 3.1.2 Floor and 3.1.3 Elevator System. Figure 5 - Apartment complex illustrates a scenario of the whole system compiled together, creating the apartment complex.

3.2 Requirements

The following requirements were used as guidelines to implement a framework that is going to be used to build the simulator on:

- Output:
  - Total energy consumption in Mega Joules
  - Average waiting time
  - Peak waiting time
  - Total number of transits
- Input:
  - Control strategies (see 3.4 Elevator controller)
  - Elevator system (described in 3.1.3 Elevator System and requirements in 3.2.1 Elevator system requirements
  - List of Floors (described in 3.1.2 Floor

3.2.1 Elevator system requirements

An elevator system is defined by an elevator cart, its counter mass, an elevator engine and a controller. The elevator cart is used to transport passengers, while the counter mass will simply be used to utilize the advantages of a traction elevator. The elevator engine is used to apply force to the system, which will be directly translated into work done. The controller plays a major role in the elevator system as this can be interpreted as an elevator strategy, which is considered as being the elevator's logic – what behavior to apply in specific situations. It is described more deeply in 3.4 Elevator controller.

The following requirements apply for the elevator system:

- Travel in a 1-dimensional vertical space.
- Be able to switch between the 5 states:
  - Acceleration
    - The state when the elevator accelerates (see Figure 1 -)
  - Deceleration
    - The state when the elevator decelerates(see Figure 1 -)
  - Constant
    - The state when the elevator travels at uniform velocity (see Figure 1 -)
  - Idle
    - The inactive state of the elevator.

    - Transfer
        - The state when passengers enter or exit the elevator.
- Only stop at specific floor levels.
- Respond to passenger requests at a specific floor.
- Pick up passengers at their respective floors.
- Transfer passengers to their destination.
- Only able to transfer a specified max load.
- Idle at a specific floor during times of no traffic.

3.3 Energy Calculation

The calculation of energy in the elevator system is done in every time step by using the described calculation model in section 2.3.1 Energy calculation model. The calculation is dependent on the current state of the elevator, its direction and the force the engine has to apply to the system in order to make the elevator cart, carrying all the passengers, move at the desired velocity for the specific time step. This is done by calculating a force magnitude which will, when summed up with the effective gravitational force on the elevator cart (see Equation 3 and Equation 4), equate to a resulting force affecting the elevator cart. This resultant force multiplied by the displacement in one time step will result in the work done in that specific time step (See Equation 2). The energy consumed is accumulated throughout the whole simulation generating a total energy consumption by the engine. This value is summed up in the end of the simulation with all other energy consuming components' respective energy consumption value.

3.4 Elevator controller

The elevator controller requires different properties variables to be set to certain values which decide which control strategy is to be used. There are three variables which decide the behavior of the controller; *GENERAL_Queuebehavior, IDLE_Position* and *IDLE_TIMEOUT*.

GENERAL_Queuebehavior is an integer which can be set to 5 different values. This value decides how the list of all calls from the floors and from inside the elevator will behave. The following behaviors are available, represented by their respective integers:

0. *No queue*

   In this case the Elevator controller will not use a queue to store calls. The first call to be registered will be treated until finished. When a trip is finished the first registered call will be the next one to be treated. This strategy works like the "Automatic operation", described in section 2.3.1.

1. *Queue with no sorting*

   The Controller will use a queue to store all calls. The queue will contain all calls, both from inside the elevator and from the floors, in the order in which they were registered. The queue will be updated in each time step.

2. *Intermediate stops when knowing direction of calls from floors*

   In this strategy the controller will use a queue for the calls from inside the elevator and a hashmap with the calls from the floors. The hashmap contains two queues with calls from the different floors with the direction (UP or DOWN) of the desired trip as a key value. Depending on the current travelling direction of the elevator the controller will merge the calls from the floors, which are mapped to the same direction, and the calls from inside the elevator into a new queue. This queue will be sorted in ascending or descending order, depending on the current travelling direction and will be used by the controller as the new list of calls. This strategy allows calls, which have been registered later, to be treated first if they are on the route that the elevator will be travelling. The queue is updated in every time step, taking new calls into account. This strategy is an implementation of the elevator algorithm (See appendix 9.2.1 Elevator Algorithm and section 2.2.2 Elevator Algorithm

3. *Intermediate stops without knowing direction of call*

   This strategy works in the similar way as strategy nr 2. The difference lies in the treatment of the calls from the floors. Instead of storing calls in a hashmap a queue is used for all calls from the floors, resulting in no sorting depending on the desired travel direction.

4. *Prioritizing Calls from inside the elevator over calls from floors*

   For this strategy the controller uses two queues, one for the calls from inside the elevator and one for the calls from the floors. If there is no current call the queue of calls from inside the elevator will be checked first and take the first call. If there exists calls from floors which lie on the planned route of the elevator, these will be prioritized. Otherwise if there are no calls from inside the elevator the current call will be the first one in the queue of calls from the floors.

The variable *IDLE_Position* is an array containing integers which represent the floors at which the elevator will idle when it goes into idle mode. The size of the array will determine during which periods the elevator shall idle at a specific floor. If for example the array is of size 3 and is going into idle mode it will during the first third of the simulator lifetime idle at the floor at index 0. During the second third it will do the same but instead idle at the floor at index 1, and so on. The time an elevator has to be inactive for it to go into idle mode is determined by *IDLE_TIMEOUT.*

The three variables mentioned above can be combined to create different control strategies.

3.5 Passenger flow simulation

The simulation of the passenger flow is updated in each time step in which the probability of a passenger sending a call for the elevator is updated on each floor. The probability is dependent on the intensity which is set for the current time interval and floor number and is calculated in the following way (See also Equation 5):

$$P(1) = \lambda\, e^{-\lambda}$$

Equation 6

$P(1)$ is the probability of one person arriving to the elevator in one time step.

4. Methodology

Different defined elevator strategies have been compared in order to evaluate which is the best for a specific apartment complex. Data has been collected by setting up different case scenarios for the simulator program and then compare them to find the most effective strategy.

4.1 Calculation of effectiveness

The mentioned output in section 3.2 Requirements are needed to compare the different elevator strategies by calculating an effectiveness value, see Equation 7, derived below:

Outputs:

E	=	total **E**nergy consumption (converted to kilo Watt/hour)
WT	=	average **W**aiting **T**ime per passnger (converted to hours)
T	=	average **T**ransit time per passenger (converted to hours)
PWT	=	**P**eak **W**aiting **T**ime (converted to hours)

Leading to the effectiveness value (EV):

$$EV = E * PWT * T * WT$$

Equation 7 - EV equation

This allows a determination of the effectiveness of an elevator in terms of an energy and passenger satisfaction perspective. A control strategy with a low effective value is a good and efficient strategy.

4.2 Simulation scenarios

The simulator allows a user to simulate a certain scenario for testing different control strategies. Below follows a list of constant and varying values for the different scenarios. The constant values are the same for every simulation whereas the varying values are the ones that distinguish one scenario from another.

4.2.1 Constant values

4.2.1.1 Floors

- Floor intensities (see appendix)
- Number of residents on each floor : 10

4.2.1.2 Passengers

- Passenger destination either ground level or their home destination.
- Passenger mass : 75 kg
- Passenger waiting tolerance : 120 s

4.2.1.3 Elevator

- Elevator Mass : 700 kg
- Max load capacity : 7 * average passenger weight
- Mass of counter mass : Elevator Mass + 0,5*Max load capacity
- Maximum velocity : 1,0 m/s
- Maximum acceleration : 0,5 m/s^2
- Transfer delay per passenger : 4 s

4.2.2 Varying values

4.2.2.1 Apartment complex

- Number of floors

For each scenario the number of floors will vary. Simulations, testing all control strategies, have been carried out in simulated apartment buildings with 5, 7 and 9 floors.

4.2.2.2 Simulation length

- Duration of the simulation

The control strategies will also be tested during a 24-hour and 1-hour simulation. The 1-hour simulation will have a high intensity and resemble the most passenger flow intensive hour from the 24-hour simulation. The intensities of passenger flow on each floor during the 24-hour simulation will resemble the traffic during a normal working day in an apartment building (as described in 2.4 Passenger flow).

4.3 Control strategies

Each control strategy described in 3.4 Elevator controller has been tested for each scenario. The idle timeout variable is set to 60, representing 60 seconds, for all strategies. The idling positions for the elevator during the 24-hour simulation will be the following:

- 00:00 – 08:00 : Top floor
- 08:00 – 20:00 : Ground floor

- 20:00 – 24:00 : Top floor

The reason for choosing these idle positions is because it is efficient when simulating a regular working day when people leave home in the morning and come back during the afternoon. During the 1-hour simulation the idling position is the top floor.

4.4 Testing

The testing has been carried out by running simulations on the described scenarios (see The control strategies will also be tested during a 24-hour and 1-hour simulation. The 1-hour simulation will have a high intensity and resemble the most passenger flow intensive hour from the 24-hour simulation. The intensities of passenger flow on each floor during the 24-hour simulation will resemble the traffic during a normal working day in an apartment building (as described in 2.4 Passenger flow).) implementing the different control strategies(see 4.3 Control strategies). The resulting efficiency value (see 4.1 Calculation of effectiveness for each simulation have been compared and presented in section Result

5. Result

5.1 Simulating 1 hour

These are the results for the simulation of 1 hour carried out on three apartments of consisting of 5, 7 and 9 floors respectively.

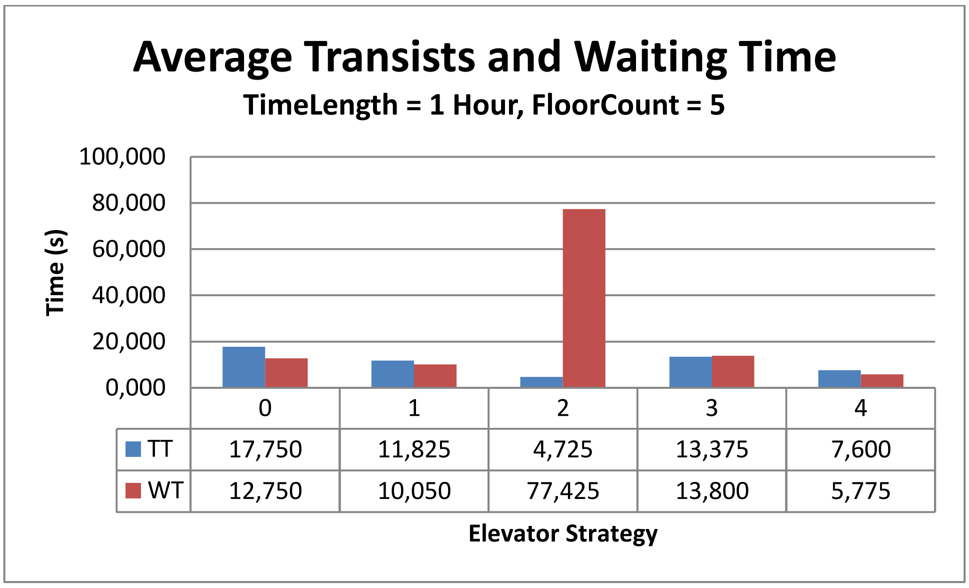
5.1.1 FloorCount = 5

Elevator Strategy	Energy Consumed (EE)	Average Transit Time (TT)	Average Waiting Time (WT)	Peak Waiting Time (PWT)	Effectiveness Value (EV)	Effectiveness increase (EI)
0	170919.647	17.750	12.750	65.000	698411.505	1.000
1	165829.095	11.825	10.050	36.000	197073.370	3.544
2	77799.541	4.725	77.425	120.000	948721.645	0.736
3	152821.445	13.375	13.800	120.000	940233.939	0.743
4	63310.240	7.600	5.775	33.000	25471.292	27.420

Table 1

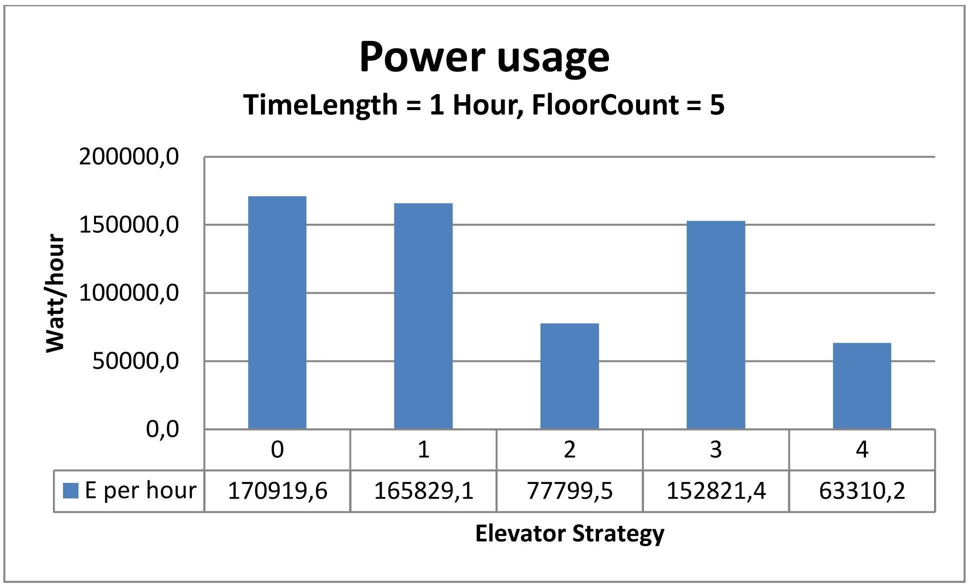
Elevator Strategy	Transits made (TT)
0	36
1	36
2	13
3	35
4	36

Table 2



	0	1	2	3	4
🟦 TT	17,750	11,825	4,725	13,375	7,600
🟥 WT	12,750	10,050	77,425	13,800	5,775

Figure 6



	0	1	2	3	4
🟦 E per hour	170919,6	165829,1	77799,5	152821,4	63310,2

Figure 7

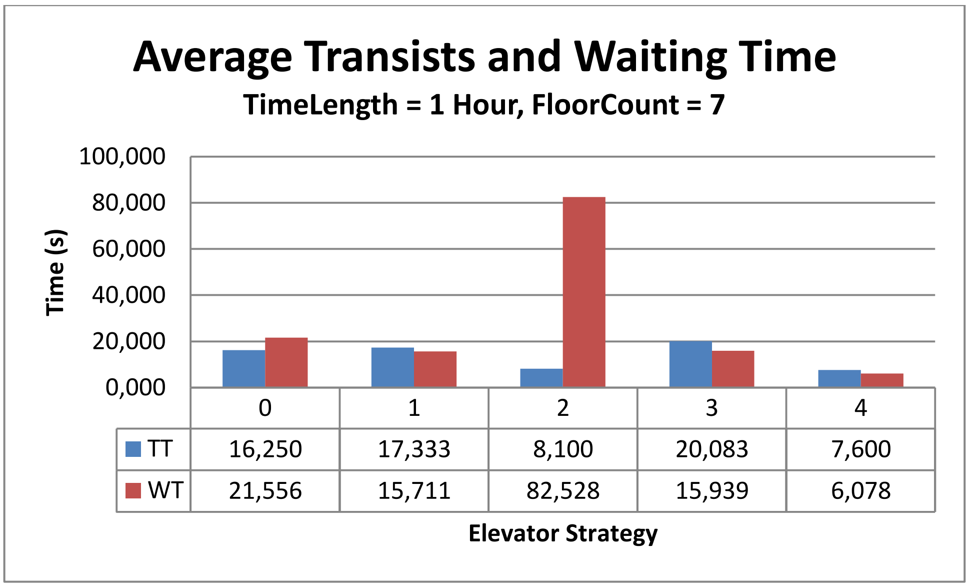
5.1.2 FloorCount = 7

Elevator Strategy	Energy Consumed (EE)	Average Transit Time (TT)	Average Waiting Time (WT)	Peak Waiting Time (PWT)	Effectiveness Value (EV)	Effectiveness increase (EI)
0	367120.246	16.250	21.556	87.000	3107689.876	1.000
1	328589.290	17.333	15.711	41.000	1019116.302	3.049
2	169571.285	8.100	82.528	120.000	3778472.155	0.822
3	320986.010	20.083	15.939	114.000	3253735.863	0.955
4	100438.093	7.600	6.078	26.000	33506.396	92.749

Table 3

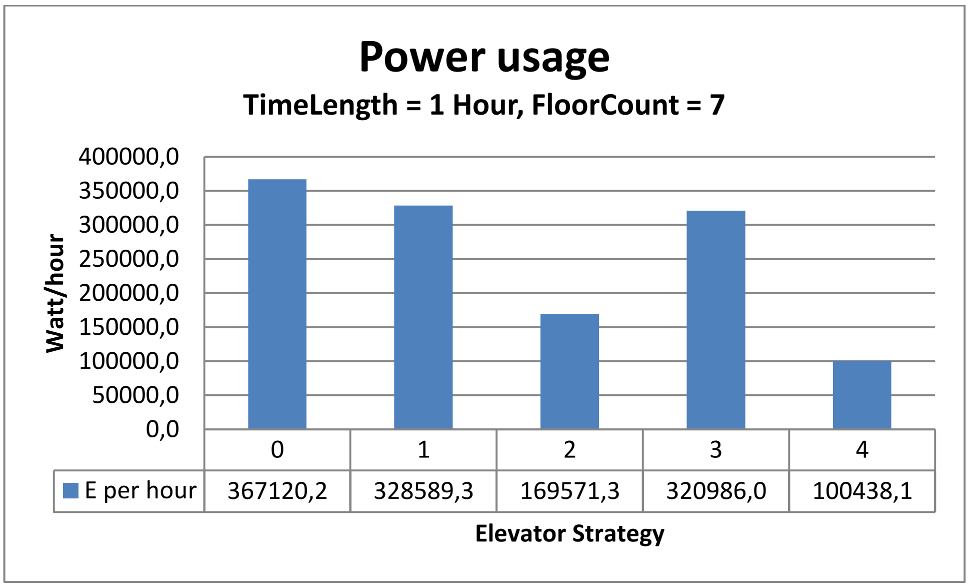
Elevator Strategy	Transits made (TT)
0	59
1	59
2	26
3	59
4	59

Table 4



	0	1	2	3	4
🟦 TT	16,250	17,333	8,100	20,083	7,600
🟥 WT	21,556	15,711	82,528	15,939	6,078

Figure 8



	0	1	2	3	4
■ E per hour	367120,2	328589,3	169571,3	320986,0	100438,1

Figure 9

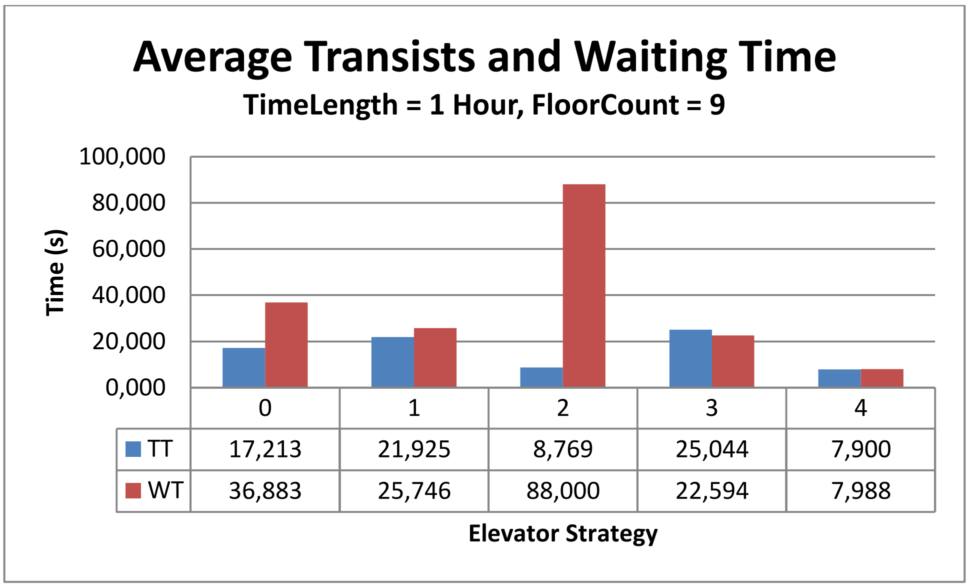
5.1.3 FloorCount = 9

Elevator Strategy	Energy Consumed (EE)	Average Transit Time (TT)	Average Waiting Time (WT)	Peak Waiting Time (PWT)	Effectiveness Value (EV)	Effectiveness increase (EI)
0	496007.311	17.213	36.883	120.000	10496413.724	1.000
1	440805.063	21.925	25.746	77.000	5322079.454	1.972
2	183049.483	8.769	88.000	120.000	4708337.781	2.229
3	411395.210	25.044	22.594	89.000	5754855.396	1.824
4	121941.205	7.900	7.988	47.000	100457.832	104.486

Table 5

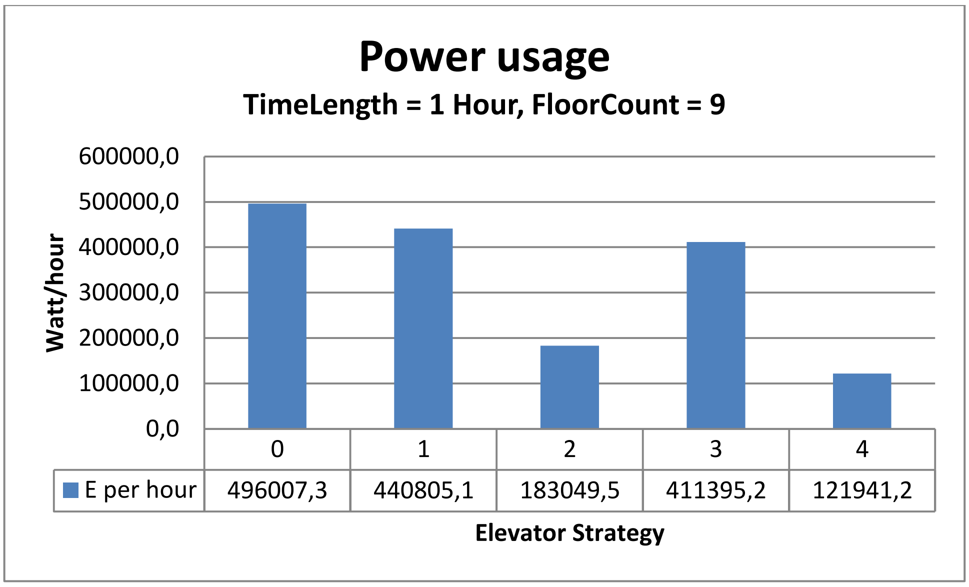
Elevator Strategy	Transits made (TT)
0	70
1	76
2	30
3	75
4	76

Table 6



	0	1	2	3	4
🟦 TT	17,213	21,925	8,769	25,044	7,900
🟥 WT	36,883	25,746	88,000	22,594	7,988

Figure 10



	0	1	2	3	4
🟦 E per hour	496007,3	440805,1	183049,5	411395,2	121941,2

Figure 11

5.2 Simulating 24 hours

These are the results for the simulation of 24 hour carried out on three apartments of consisting of 5, 7 and 9 floors respectively.

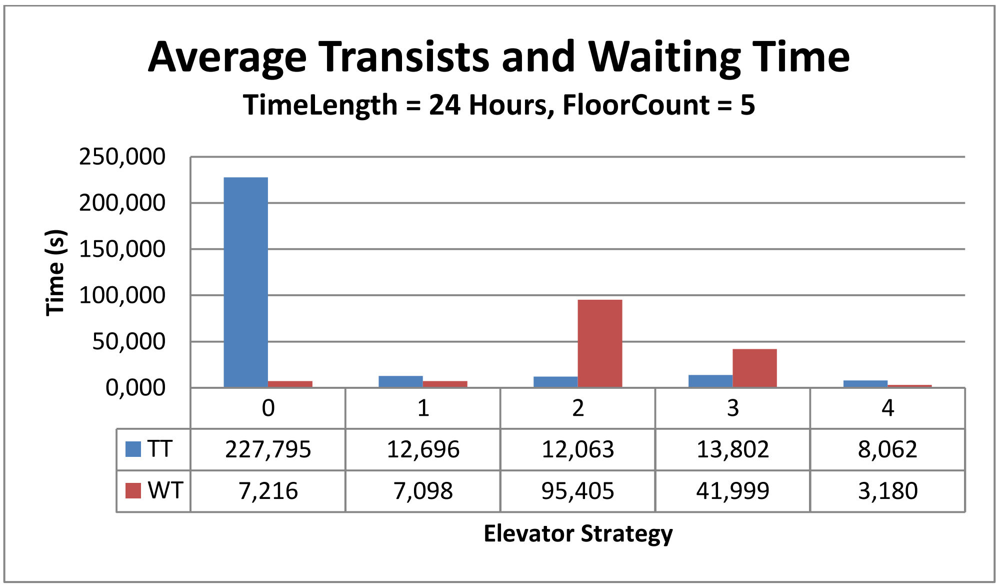
5.2.1 FloorCount = 5

Elevator Strategy	Energy Consumed (EE)	Average Transit Time (TT)	Average Waiting Time (WT)	Peak Waiting Time (PWT)	Effectiveness Value (EV)	Effectiveness increase (EI)
0	930428.863	227.795	7.216	28.000	495617.853	1.000
1	1177153.512	12.696	7.098	25.000	30692.601	16.148
2	332504.191	12.063	95.405	120.000	531461.740	0.933
3	908495.401	13.802	41.999	120.000	731404.319	0.678
4	404621.458	8.062	3.180	13.000	1560.959	317.509

Table 7

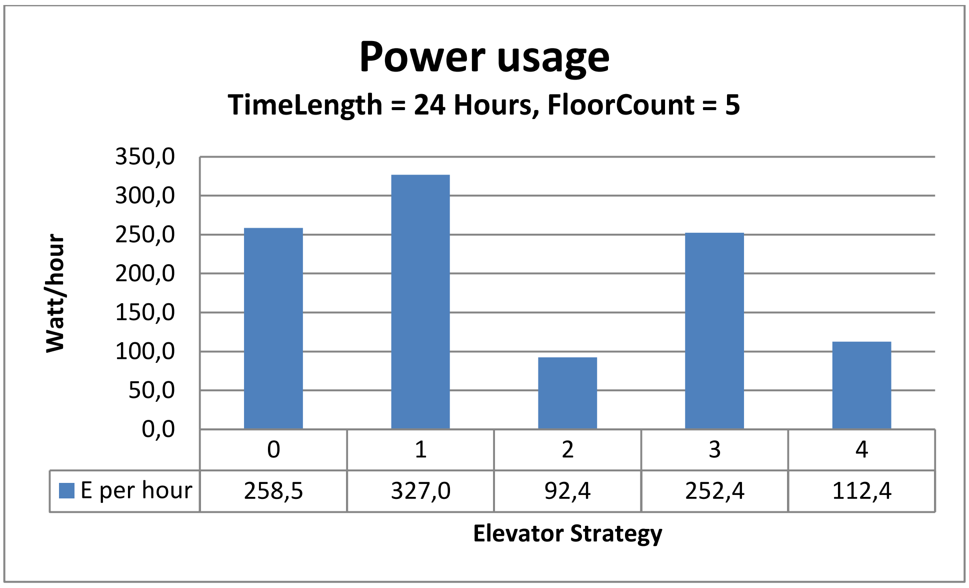
Elevator Strategy	Transits made (TT)
0	213
1	213
2	53
3	154
4	213

Table 8



	0	1	2	3	4
🟦 TT	227,795	12,696	12,063	13,802	8,062
🟥 WT	7,216	7,098	95,405	41,999	3,180

Figure 12



	0	1	2	3	4
🟦 E per hour	258,5	327,0	92,4	252,4	112,4

Figure 13

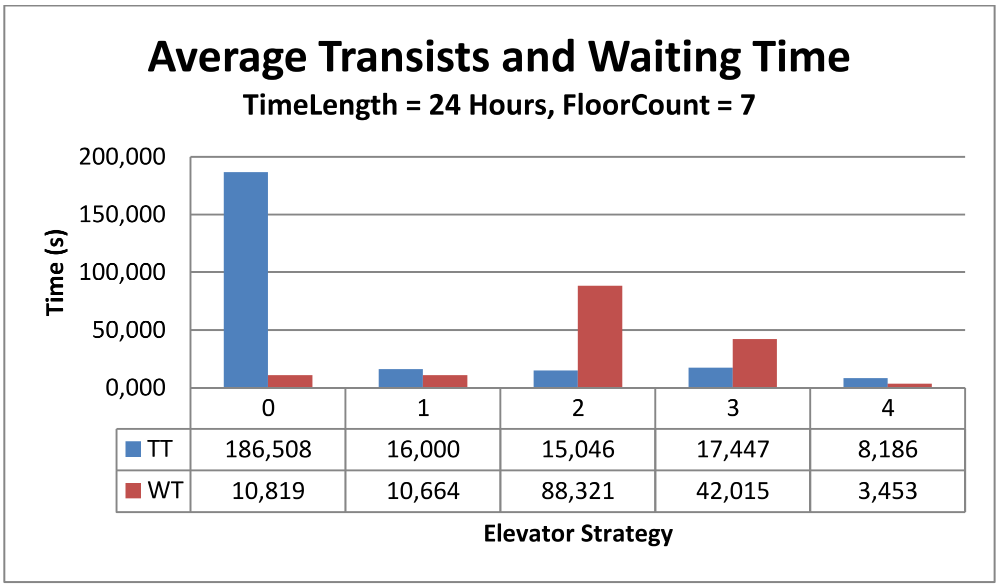
5.2.2 FloorCount = 7

Elevator Strategy	Energy Consumed (EE)	Average Transit Time (TT)	Average Waiting Time (WT)	Peak Waiting Time (PWT)	Effectiveness Value (EV)	Effectiveness increase (EI)
0	1694414.141	186.508	10.819	70.000	2769981.805	1.000
1	2060414.215	16.000	10.664	47.000	191235.415	14.485
2	799130.925	15.046	88.321	120.000	1474903.734	1.878
3	1634789.733	17.447	42.015	120.000	1664390.175	1.664
4	501197.180	8.186	3.453	19.000	3114.801	889.297

Table 9

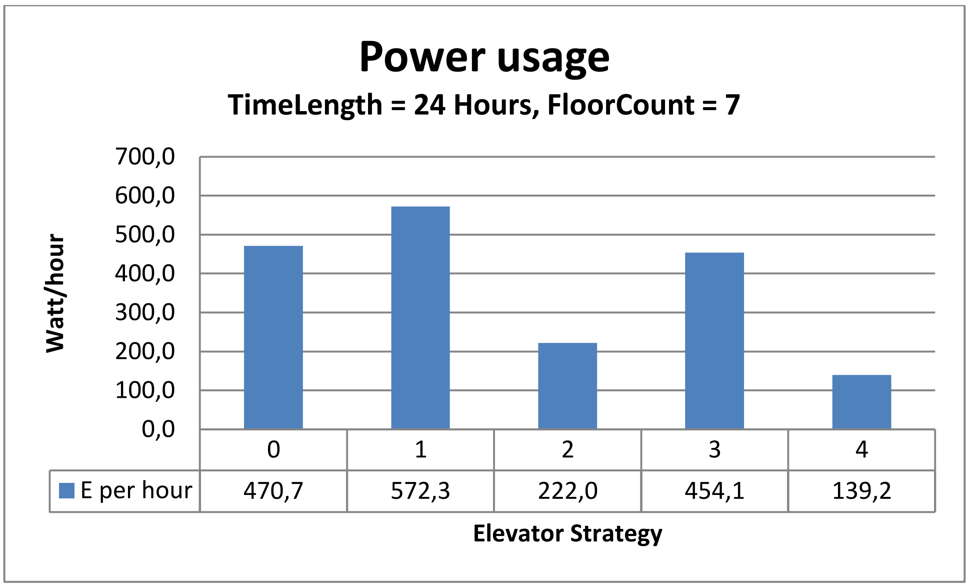
Elevator Strategy	Transits made (TT)
0	262
1	263
2	90
3	198
4	263

Table 10



	0	1	2	3	4
🟦 TT	186,508	16,000	15,046	17,447	8,186
🟥 WT	10,819	10,664	88,321	42,015	3,453

Figure 14



	0	1	2	3	4
🟦 E per hour	470,7	572,3	222,0	454,1	139,2

Figure 15

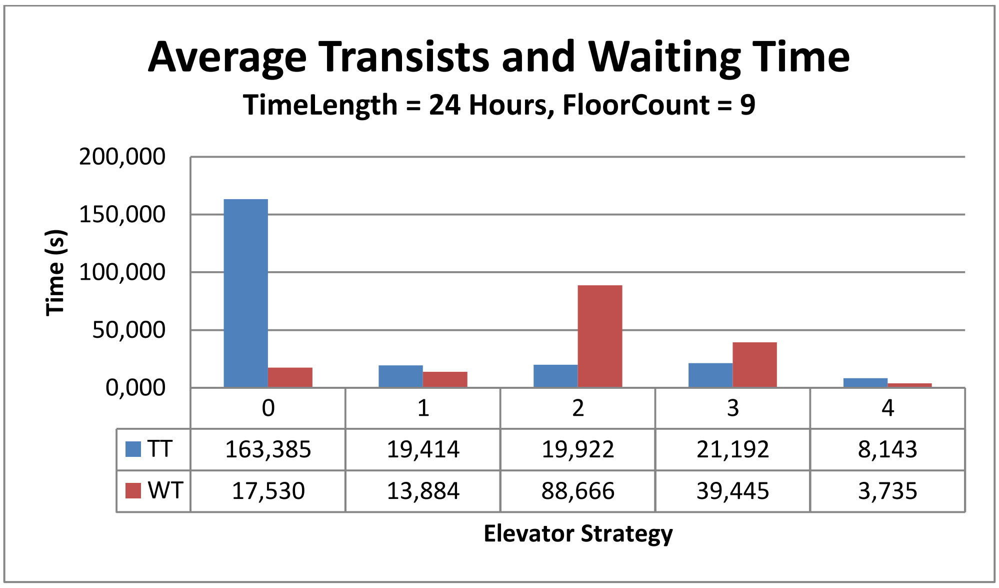
5.2.3 FloorCount = 9

Elevator Strategy	Energy Consumed (EE)	Average Transit Time (TT)	Average Waiting Time (WT)	Peak Waiting Time (PWT)	Effectiveness Value (EV)	Effectivenes s increase (EI)
0	2585343.960	163.385	17.530	120.000	10284409.93 3	1.000
1	2896046.513	19.414	13.884	57.000	514965.581	19.971
2	1258775.479	19.922	88.666	120.000	3088170.487	3.330
3	2465449.290	21.192	39.445	120.000	2862394.636	3.593
4	555813.050	8.143	3.735	29.000	5674.848	1812.279

Table 11

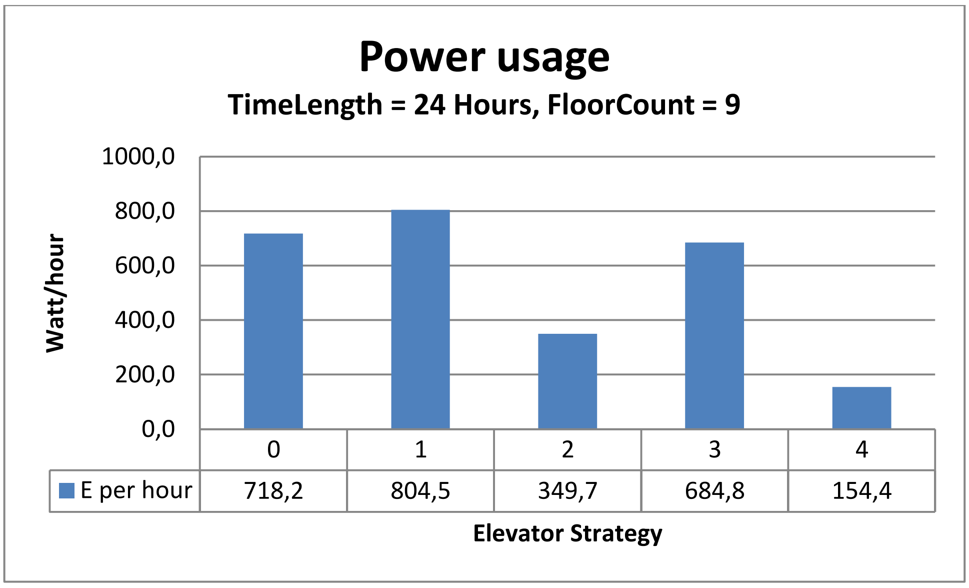
Elevator Strategy	Transits made (TT)
0	288
1	292
2	116
3	240
4	292

Table 12



	0	1	2	3	4
🟦 TT	163,385	19,414	19,922	21,192	8,143
🟥 WT	17,530	13,884	88,666	39,445	3,735

Figure 16



	0	1	2	3	4
🟦 E per hour	718,2	804,5	349,7	684,8	154,4

Figure 17

5.3 Energy increase over floor size

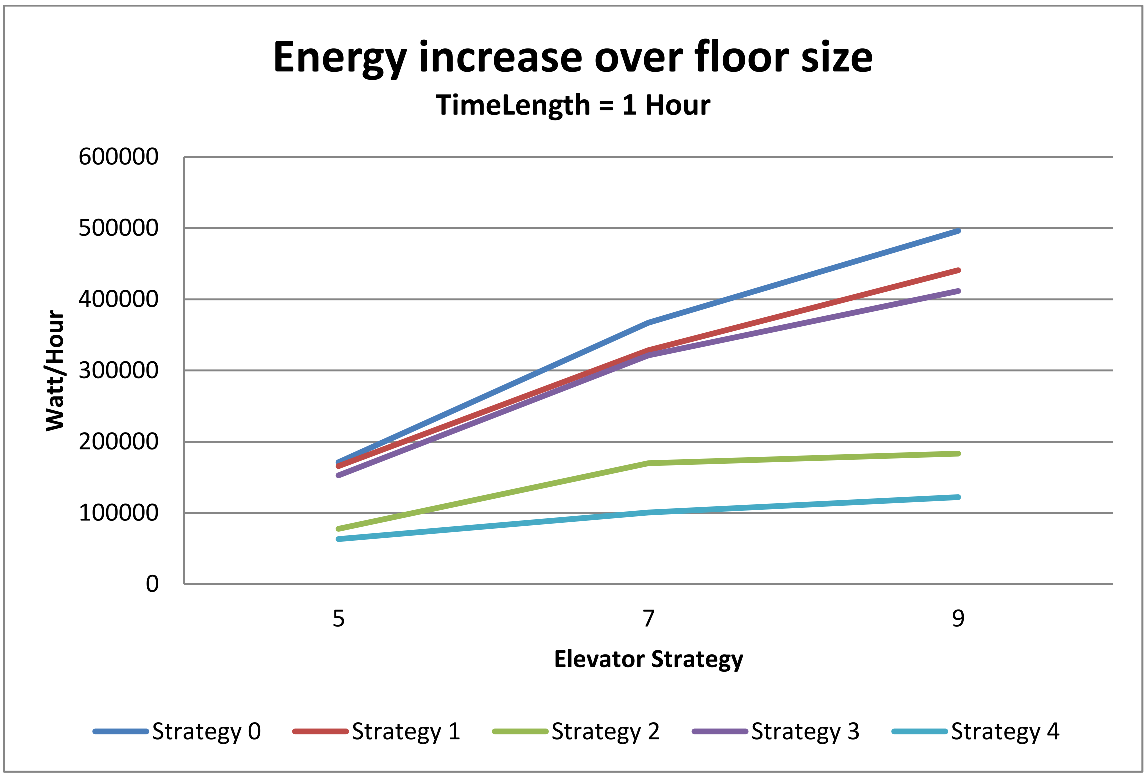


Figure 18

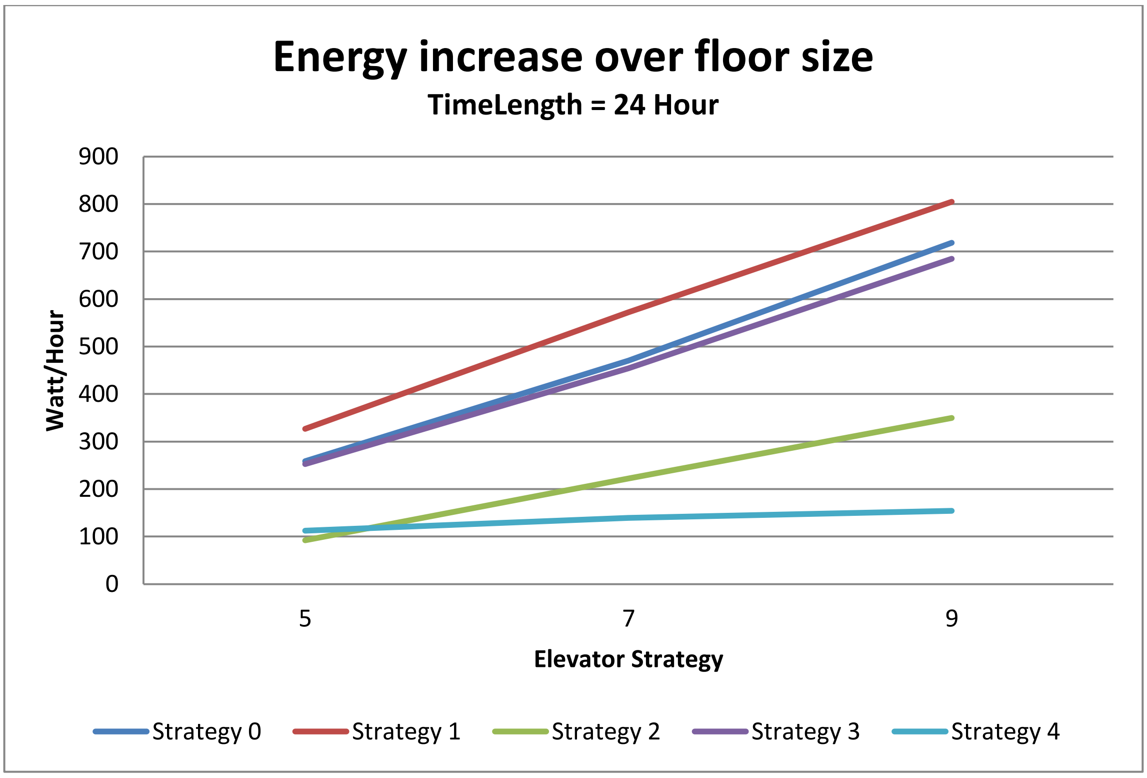


Figure 19

6. Discussion and Analysis

6.1 Simulations for 1 hour

By looking at the simulations simulating an intensive 1-hour period it seems obvious that control strategy number 4 (see 3.4 Elevator controller) is the most efficient, in both energy consumption, waiting time and of course its EV (see Table 1, Table 3 and Table 5). Another interesting result is the high average waiting time when using strategy number 2. The waiting time, total energy consumption and the average transit time all increase when the number of floors increases. However, the increase is a lot higher for control strategy 0 than for the other strategies (see Figure 18).

Strategy 2 consistently has, compared to the others, a high average waiting time, which might be because it will miss certain passengers on certain floors when the elevator is travelling up from the ground floor. This is most probably because it will pick up all passengers requesting to travel in the same direction as the elevator. This is done using a queue of requests evaluated using a merge and sort algorithm, which is recalculated in every time step (see 3.4 Elevator controller). Thus, passengers that receive a bad position in the merge and sorted queue of requests and orders will not be able to enter the elevator, since it is full when it passes the respective passengers floor. In this case, where passengers only travel between their home and the ground floor, the growth of number of floors will lead to an increase in probability that the elevator is full the closer a passenger floor is to the ground level. This of course will probably make the passenger exceed its tolerance time for waiting, meaning that they will take the stairs instead, explaining the low count of transits made, compared to the other strategies (see Table 2, Table 4 and Table 6). Interestingly, the energy consumed is very low, despite the fact that this strategy will make the elevator travel to empty requests, as passengers have already left, (due to high waiting time, see Figure 6, Figure 8 and Figure 10) just as all the other strategies. The low energy consumption is due to the elevators behavior in this test-case, when travelling up, and not the low transit count as might be the first initial thought. When travelling up the elevator will pick up passengers on all floors requesting to go up, which is none, except for the ones on the ground level, meaning that the elevator will only accelerate once to get to its destination. Accelerating against gravity is the most energy consuming action an elevator can do and is thus the explanation for the low energy consumption, as the elevator mostly accelerates by going in the direction of the force of gravity.

When reaching the lower floors the elevator cart will be full and not able to take more passengers, and is therefore not suitable for the apartment buildings tested.

Strategy 3 is very similar to strategy 2, except that it will sort passenger requests after the closest floors due to the direction the elevator currently travels (see 3.4 Elevator controller). This means, that unlike strategy 2, that strategy 3 will in this test-case, where everybody travels between ground floor and home, allow the elevator to stop when either going down or up. It will probably accelerate upwards, against gravity, from an idling state very often, when compared to strategy 2. This is probably also a reason why it is in the upper region, when it comes to energy consumption, in Figure 18 (together with strategy 1 and 0).

The high increase in energy consumption for strategy 0 is because of the fact that this strategy only does single trips and doesn't remember calls, meaning that it may travel considerably more than strategy 1 (being fairly similar except for its waiting queue, see 3.4 Elevator controller) or in fact all the others. Since the number of residents in the apartment building higher when the number of floors is higher, this means that the elevators travelling distance will increase rapidly with an increase of floors and thus passengers living in the apartment (see Figure 18), which explains that strategy 0 has the highest growth in Figure 18.

The exceptional behavior and results of strategy number 4 is because of the prioritizing of calls inside the elevator, which means prioritizing transporting passengers to their destination, unlike the other strategies. Strategy 4 tries to prevent that the elevator cart becomes full and thus reduces the travelling distance, since it won't reach floors when it's full as often as the other strategies. This is very well reflected in Table 2, Table 4 and Table 6, which illustrate a very high transit value that contributes to the low energy value and growth over increased number of floors (see Figure 18). Figure 6, Figure 8 and Figure 10 also illustrate very low waiting times and this is due to passengers most likely being picked up by the elevator.

6.2 Simulations for 24 hours

The results from the 24 hour simulation are in general similar to the ones from the simulation of 1 intensive hour. One of the big differences though is the high average transit time for strategy 0(see Figure 12, Figure 14, and Figure 16). Another change is that during the 24-hour simulations strategy number 1 is the one that consumes most energy (see Figure 19). It appears that the strategy to prefer from this simulation is again strategy number 4.

The reason behind the high average transit time for strategy number 0 is inconclusive. This can be because of a bug in the simulator leaving passengers in the elevator during idling. Another reason could be that the passengers’ orders might simply be lost as other passengers make requests or orders before them resulting in a long elevator trip for a certain passenger.

By comparing the data from strategy 0 and strategy 1 it can be seen that they manage to transfer the same number of passengers during one day(see Table 8, Table 10 and Table 12) and have similar average waiting time values(see Table 7, Table 9 and Table 11). From this information it can be deduced that the high energy consumption for strategy 1 is probably caused by a high number of accelerations, similar to the phenomenon seen in 6.1 Simulations for 1 hour about strategy number 3. The behavior of the elevator with control strategy 1 is very dependent on the passenger flow from each floor since it will serve the orders and requests in the order in which they were called. This can cause the elevator to travel up and down between floors, causing the problem just stated.

7. Conclusion

From the results and the analysis from this research it can be concluded that strategy number 4 is the most effective for apartment buildings with 5, 7 or 9 floors considering energy effectiveness, average transit time, peak waiting time and lowest waiting time. This strategy has by far, compared with the other strategies, the lowest effectiveness value (see   Equation 7 - EV equation). Strategy number 4 is the only one that prioritizes passengers in the elevator which seems to have been a key factor when implementing these elevator control strategies for the simulated test cases resembling apartment buildings.

The hypothesis was wrong at least for the test case in where the simulations where carried out, which can be seen in the poor results from strategy 2. For the specific passenger flow, representing a typical working day in an apartment building, these strategies were not sufficient. However, the control strategy two might perform better in other test scenarios.

8. Bibliography

Barney, G. C. & dos Santos, S. M., 1985. *Elevator Traffic Analysis and Control.* 2nd ed. London, UK: Peter Peregrinus.

Edinburgh, I. a. U. o., n.d. *Disk scheduling, Internet archive, University of Edinburgh.* [Online]
Available at:
http://web.archive.org/web/20080606005055/http://www.dcs.ed.ac.uk/home/stg/pub/D/disk.html
[Accessed 09 04 2013].

Peters, R., Al-Sharif, L. & Smith, R., 2004. Elevator Energy Simulation Model.

Strakosch, G. R. & Carporale, R. S., 2010. *The Vertical Transportation Handbook.* 4th ed. s.l.:John Wiley & Sons.

Susi, T., Sorsa, J. & Siikonen, M.-L., 2004. *Passenger behaviour in elevator simulation,* s.l.: KONE.

Wikipedia, n.d. [Online]
Available at: http://en.wikipedia.org/wiki/Elevator_algorithm
[Accessed 09 04 2013].

9. Appendix

9.1 Equations

Equation 1

$$K.E. = \frac{1}{2}mv^2$$

Definitions:

K.E = Kinetic Energy, *m* = mass, *v* = velocity

Equation 2

$$K.E. = F \times s$$

Definitions:

K.E = Kinetic Energy, *F* = Force, s = displacement

Equation 3

$$F_{gravity} = |m_{Cart} - m_{CounterMass}| * a_{gravity}$$

Definitions:

$F_{gravity}$ = Gravity force, m_{cart} = mass of cart, $m_{counterMass}$ = mass of counter mass, $a_{gravity}$ = gravity acceleration(= 9,81 ms^{-2})

Equation 4

$$F_{result} = F_{gravity} + F_{system}$$

Definitions:

F_{result} = resultant force, $F_{gravity}$ = gravity force, F_{system} = Force of the system

Note: F_{system} is the force exerted on the cart by the engine, whereas F_{result} is the actual force affecting the elevator Cart.

Equation 5

$$P(n) = \frac{(\lambda T)^n}{n!}\, e^{-\lambda T}$$

Definitions:

P(n) = Probability of n events within time T, *λ* = intensity(expected number of events in time *T*), *T* = Time elapsed, *n* = number of events

Equation 6

$$P(1) = \lambda\, e^{-\lambda}$$

Definitions:

P(1) = Probability of 1 events in 1 time unit, *λ* = intensity(expected number of events in 1 time unit)

Equation 7

$$EV = E * PWT * T * WT$$

Definitions:

E = total **E**nergy consumption (converted to kilo Watt/hour)

WT = average **W**aiting **T**ime per passnger (converted to hours)

T = average **T**ransit time per passenger (converted to hours)

PWT = **P**eak **W**aiting **T**ime (converted to hours)

9.2 Definitions

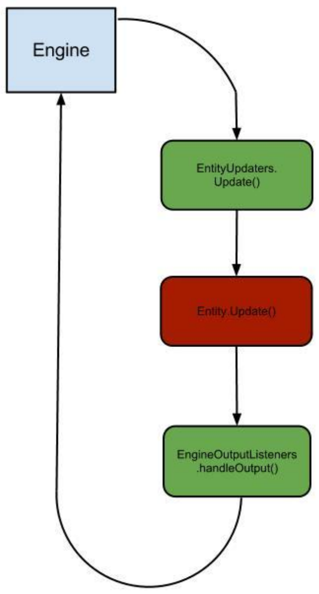
9.2.1 Elevator Algorithm

The Elevator Algorithm is primarily a disk scheduling algorithm to determine how the disk arm should behave when writing and reading from a hard drive. When the arm is moving in a direction it will only treat requests in the same direction. It will only change direction when idling and there are no more calls in the same direction. (Edinburgh, u.d.).

9.3 Simulator architechture

9.3.1 Simulator Engine

The simulator engine will update itself for a specified amount of times, which is defined during initialization of the program. This amount is defined as the number of time steps that the engine will run before terminating and one time step implies one update.



The simulator engine will make all instances implementing EntityUpdater or Entity interface run their respective update method and all implementing the EngineOutputListeners interface will invoke their handleOutput method in every timestep.

9.3.2 Entity interface

An instance that implements the Entity interface will have to implement an update method. The update method is usually used and required for changing the state of an Entity according to external changes, often manipulated by instances implementing the EntityUpdater interface (see 9.3.3 EntityUpdater interface)9.3.3 EntityUpdater interface .

In practice an Entity could be anything that requires an update and is considered to be one of the primitive interfaces in this simulator implementation.

Many entities used in this simulator implement interface extend the Entity interface or even implement several Entity sub-interfaces.

9.3.2.1 Entity interface extensions

- EntityEnergyConsumer
    - An entity that consumes energy.
    - Implementations (see 9.6 Documentation):
        - Engine
        - Cart
        - EntityFreightSystem
- EntityMass
    - An entity that has a mass.
- EntityMovingMass
    - An entity that extends EntityMass and is able to be influenced by forces.
    - Implementations (see 9.6 Documentation):
        - Mass (Abstract class)
            - Cart
            - CounterMass
- EntityLoad
    - An entity that extends EntityMass and can live on an EntityFloor and be transported by an EntityFreightSystem.
    - Implementations (see 9.6 Documentation):
        - Passenger
- EntityFloor
    - An entity that can hold a collection of EntityLoad instances in a “residents” queue or a “waiting” queue. It also holds information needed for the calculation of current EntityLoad flows (to the queue).
    - Implementations (see 9.6 Documentation):
        - Floor
- EntityFreightSystem

    - An entity that represents a transport system that can transport EntityLoad between different EntityFloor.
    - This also extends the EntityEnergyConsumer.
    - Implementations (see 9.6 Documentation):
        - Elevator

9.3.2.2 ElevatorControllerFramework

ElevatorControllerFramework is an abstract class that directly implements the Entity interface. Its sole purpose is to set a framework for different implementations of elevator controllers, thus making the controller part very modular. The idea is that the simulator will call the update method of an ElevatorController instance, which will handle the different calls received using the implemented logic.

9.3.3 EntityUpdater interface

Any instance that implements the EntityUpdater interface can be considered to be a specific rule that is used to update an Entity instance, therefore the name Entity-Updater.

The implementations in this simulator will manipulate entities by changing states of the different entities within the system (For easier reference we will refer to the implementations mentioned in 9.3.2.1 Entity interface extensions):

- Gravitational force applied on Cart and Countermasses
    - Affects resultant force.
- Elevator requests or calls made on specific floors or within the elevator cart itself by Passenger instances.
    - Invokes calls on the ElevatorAI implementation.
- The placement of Passengers from a Floor to an Elevator or vice versa.
    - Affects mass of cart, which changes the resultant force experienced by the elevator system.
- The flow of Passenger on each floor that comes to make elevator requests.
    - Affects the number of requests made.

The current simulator engine implementation is very general and modular, which makes it very easy to add new EntityUpdater implementations. The implementation is simply added to a list of EntityUpdater and then invoked using the update method inherited from the EntityUpdater interface.

9.3.4 EngineOutputListeners interface

Any instance that implements the EngineOutputListener interface can be used to use the information currently present in the current timestep of the simulator engine. This can be as simple as taking a snapshot of specific data and printing it out to the terminal or as complex as presenting the current context in a graphical user interface.

The only implementation currently used is OutputStatistics, which reviews elevator and passenger statistics at a certain interval during the simulation. It will also make a final snapshot in order to compile the final statistics after one simulation run. The key values that this implementation will output is for example the elevator energy consumption, average passenger waiting time and the maximum waiting time.

Similarly like with the EntityUpdater implementations, EngineOutputListener implementations can simply be added to a list of EngineOutputListener and then be invoked using the handleOutput method inherited from the EngineOutputListener interface.

9.4 Glossary

Direction state

The state describing the travel direction of the elevator cart. This variable has the value “UP” or “DOWN”.

Effective gravitational force

In a traction elevator it is the resulting force of the balancing of the elevator cart and its counter mass. This means that if the elevator cart is heavier, then the resultant force will point down from the cart yet in opposite direction for the counter mass and vice versa.

Elevator behavior

What the elevator should do depending on its different states and its implemented elevator control strategy.

Elevator state

Is a state that defines how an elevator should generally behave. An elevator can be in an IDLE, ACCELERATION, DECELERATION, CONSTANT and TRANSFER state.

Framework

In this case another word for structure of the simulator.

Order

An order is a call made to the elevator from a passenger inside the elevator.

Peak waiting time

The maximum time that a passenger has waited.

Property

A state that stays constant throughout a simulation. One may also simply consider it as a constant value.

Resultant force

The force that result from two or more forces pushing or repelling each other.

Request

A request is a call to an elevator made by passengers waiting for the elevator on a specific floor.

Snapshot

A view of states in the same time step.

Time step

Defines the frequency of when the program does updates. In our case it is equivalent to the time unit seconds. That is, 1 time step == 1 second.

Transit

The transfer of a passenger from one floor to another is considered as a transit.

Waiting time

The time that a passenger has waited for an elevator to arrive to a floor.

9.5 Data recordings

Each recording is an average output from 100 simulations.

9.5.1 Simulation time = 1 hour

9.5.1.1 FloorCount = 5

```
Floors: 5
Time: 1 h
Control strategy: 0
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 12.75
AVERAGE_TRANSIT_TIME: 17.75
AVERAGE_ENERGY: 4747.7679715277245
AVERAGE_TRANSITS: 0.9
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 170919.6469749981
TOTAL_TRANSITS: 36.0
PEAK_WAITING_TIME: 65.0
EFFECTIVE_VALUE: 698411.5053866393
-------------------------------------------------

Floors: 5
Time: 1 h
Control strategy: 1
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 10.05
AVERAGE_TRANSIT_TIME: 11.825
AVERAGE_ENERGY: 4606.363760416617
AVERAGE_TRANSITS: 0.9
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 165829.0953749982
TOTAL_TRANSITS: 36.0
PEAK_WAITING_TIME: 36.0
EFFECTIVE_VALUE: 197073.36980734006
-------------------------------------------------

Floors: 5
Time: 1 h
Control strategy: 2
```

```
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 77.425
AVERAGE_TRANSIT_TIME: 4.725
AVERAGE_ENERGY: 5984.58010384615
AVERAGE_TRANSITS: 0.325
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 77799.54134999996
TOTAL_TRANSITS: 13.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 948721.64452124
-------------------------------------------------

Floors: 5
Time: 1 h
Control strategy: 3

-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 13.8
AVERAGE_TRANSIT_TIME: 13.375
AVERAGE_ENERGY: 4366.3269942856705
AVERAGE_TRANSITS: 0.875
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 152821.44479999845
TOTAL_TRANSITS: 35.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 940233.9391319904
-------------------------------------------------

Floors: 5
Time: 1 h
Control strategy: 4
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 5.775
AVERAGE_TRANSIT_TIME: 7.6
AVERAGE_ENERGY: 1758.6177833333347
AVERAGE_TRANSITS: 0.9
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 63310.24020000005
TOTAL_TRANSITS: 36.0
PEAK_WAITING_TIME: 33.0
```

```
EFFECTIVE_VALUE: 25471.29238846502
--------------------------------------------------
```

9.5.1.2 FloorCount = 7

```
Floors: 7
Time: 1 h
Control strategy: 0
--------------------------------------------------
### End result ###
--------------------------------------------------
AVERAGE_WAITING_TIME: 21.555555555555554
AVERAGE_TRANSIT_TIME: 16.25
AVERAGE_ENERGY: 6222.37704533888
AVERAGE_TRANSITS: 0.9833333333333333
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 367120.2456749939
TOTAL_TRANSITS: 59.0
PEAK_WAITING_TIME: 87.0
EFFECTIVE_VALUE: 3107689.875946493
--------------------------------------------------


Floors: 7
Time: 1 h
Control strategy: 1
--------------------------------------------------
### End result ###
--------------------------------------------------
AVERAGE_WAITING_TIME: 15.711111111111112
AVERAGE_TRANSIT_TIME: 17.333333333333332
AVERAGE_ENERGY: 5569.309995338894
AVERAGE_TRANSITS: 0.9833333333333333
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 328589.28972499474
TOTAL_TRANSITS: 59.0
PEAK_WAITING_TIME: 41.0
EFFECTIVE_VALUE: 1019116.3015338229
--------------------------------------------------


Floors: 7
Time: 1 h
Control strategy: 2
--------------------------------------------------
### End result ###
--------------------------------------------------
```

```
AVERAGE_WAITING_TIME: 82.52777777777779
AVERAGE_TRANSIT_TIME: 8.1
AVERAGE_ENERGY: 6521.9724951922335
AVERAGE_TRANSITS: 0.43333333333333335
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 169571.28487499806
TOTAL_TRANSITS: 26.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 3778472.1552271447
-------------------------------------------------


Floors: 7
Time: 1 h
Control strategy: 3
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 15.938888888888888
AVERAGE_TRANSIT_TIME: 20.083333333333332
AVERAGE_ENERGY: 5440.440842372795
AVERAGE_TRANSITS: 0.9833333333333333
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 320986.0096999949
TOTAL_TRANSITS: 59.0
PEAK_WAITING_TIME: 114.0
EFFECTIVE_VALUE: 3253735.8628983777
-------------------------------------------------


Floors: 7
Time: 1 h
Control strategy: 4
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 6.0777777777777775
AVERAGE_TRANSIT_TIME: 7.6
AVERAGE_ENERGY: 1702.340558474574
AVERAGE_TRANSITS: 0.9833333333333333
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 100438.09294999986
TOTAL_TRANSITS: 59.0
PEAK_WAITING_TIME: 26.0
EFFECTIVE_VALUE: 33506.395803411186
-------------------------------------------------
```

9.5.1.3 FloorCount = 9

```
Floors: 9
Time: 1 h
Control strategy: 0
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 36.88333333333333
AVERAGE_TRANSIT_TIME: 17.2125
AVERAGE_ENERGY: 7085.818735357016
AVERAGE_TRANSITS: 0.875
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 496007.3114749911
TOTAL_TRANSITS: 70.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 1.0496413724062858E7
-------------------------------------------------


Floors: 9
Time: 1 h
Control strategy: 1
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 25.745833333333337
AVERAGE_TRANSIT_TIME: 21.925
AVERAGE_ENERGY: 5800.06661611832
AVERAGE_TRANSITS: 0.95
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 440805.0628249923
TOTAL_TRANSITS: 76.0
PEAK_WAITING_TIME: 77.0
EFFECTIVE_VALUE: 5322079.453579791
-------------------------------------------------


Floors: 9
Time: 1 h
Control strategy: 2
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 88.0
AVERAGE_TRANSIT_TIME: 8.76875
```

```
AVERAGE_ENERGY: 6101.649427499925
AVERAGE_TRANSITS: 0.375
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 183049.48282499774
TOTAL_TRANSITS: 30.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 4708337.780730317
-------------------------------------------------


Floors: 9
Time: 1 h
Control strategy: 3
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 22.59375
AVERAGE_TRANSIT_TIME: 25.04375
AVERAGE_ENERGY: 5485.269467666572
AVERAGE_TRANSITS: 0.9375
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 411395.2100749929
TOTAL_TRANSITS: 75.0
PEAK_WAITING_TIME: 89.0
EFFECTIVE_VALUE: 5754855.396259828
-------------------------------------------------


Floors: 9
Time: 1 h
Control strategy: 4
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 7.9875
AVERAGE_TRANSIT_TIME: 7.9
AVERAGE_ENERGY: 1604.4895342105197
AVERAGE_TRANSITS: 0.95
TOTAL_TIME: 3600.0
TOTAL_ENERGY: 121941.20459999949
TOTAL_TRANSITS: 76.0
PEAK_WAITING_TIME: 47.0
EFFECTIVE_VALUE: 100457.83181333021
-------------------------------------------------
```

9.5.2 Simulation time = 24 hours

9.5.2.1 FloorCount = 5

```
Floors: 5
Time: 24 h
Control strategy: 0
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 7.215654761904761
AVERAGE_TRANSIT_TIME: 227.79476190476188
AVERAGE_ENERGY: 4368.210622183018
AVERAGE_TRANSITS: 5.325
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 930428.8625249829
TOTAL_TRANSITS: 213.0
PEAK_WAITING_TIME: 28.0
EFFECTIVE_VALUE: 495617.85315048706
-------------------------------------------------

Floors: 5
Time: 24 h
Control strategy: 1
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 7.097559523809522
AVERAGE_TRANSIT_TIME: 12.69595238095238
AVERAGE_ENERGY: 5526.5423095069755
AVERAGE_TRANSITS: 5.325
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 1177153.5119249858
TOTAL_TRANSITS: 213.0
PEAK_WAITING_TIME: 25.0
EFFECTIVE_VALUE: 30692.601243228608
-------------------------------------------------

Floors: 5
Time: 24 h
Control strategy: 2
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 95.4047619047619
AVERAGE_TRANSIT_TIME: 12.062500000000002
```

```
AVERAGE_ENERGY: 6273.663976414997
AVERAGE_TRANSITS: 1.325
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 332504.1907499948
TOTAL_TRANSITS: 53.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 531461.7402874903
-------------------------------------------------

Floors: 5
Time: 24 h
Control strategy: 3
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 41.998690476190475
AVERAGE_TRANSIT_TIME: 13.801666666666668
AVERAGE_ENERGY: 5899.320788636253
AVERAGE_TRANSITS: 3.85
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 908495.401449983
TOTAL_TRANSITS: 154.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 731404.3188415689
-------------------------------------------------

Floors: 5
Time: 24 h
Control strategy: 4
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 3.180446428571428
AVERAGE_TRANSIT_TIME: 8.061666666666667
AVERAGE_ENERGY: 1899.6312586854224
AVERAGE_TRANSITS: 5.325
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 404621.458099995
TOTAL_TRANSITS: 213.0
PEAK_WAITING_TIME: 13.0
EFFECTIVE_VALUE: 1560.9588068397547
-------------------------------------------------
```

9.5.2.2 FloorCount = 7

```
Floors: 7
```

```
Time: 24 h
Control strategy: 0
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 10.818730158730158
AVERAGE_TRANSIT_TIME: 186.5076984126984
AVERAGE_ENERGY: 6467.22954751972
AVERAGE_TRANSITS: 4.366666666666666
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 1694414.1414501665
TOTAL_TRANSITS: 262.0
PEAK_WAITING_TIME: 70.0
EFFECTIVE_VALUE: 2769981.805135254
-------------------------------------------------


Floors: 7
Time: 24 h
Control strategy: 1
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 10.663928571428572
AVERAGE_TRANSIT_TIME: 15.999722222222223
AVERAGE_ENERGY: 7834.274579944072
AVERAGE_TRANSITS: 4.383333333333334
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 2060414.2145252908
TOTAL_TRANSITS: 263.0
PEAK_WAITING_TIME: 47.0
EFFECTIVE_VALUE: 191235.415166203
-------------------------------------------------


Floors: 7
Time: 24 h
Control strategy: 2
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 88.32059523809524
AVERAGE_TRANSIT_TIME: 15.045833333333333
AVERAGE_ENERGY: 8879.23250305539
AVERAGE_TRANSITS: 1.5
TOTAL_TIME: 86400.0
```

```
TOTAL_ENERGY: 799130.925274985
TOTAL_TRANSITS: 90.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 1474903.7342903435
-------------------------------------------------



Floors: 7
Time: 24 h
Control strategy: 3
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 42.01452380952381
AVERAGE_TRANSIT_TIME: 17.447222222222216
AVERAGE_ENERGY: 8256.513802399764
AVERAGE_TRANSITS: 3.3
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 1634789.7328751534
TOTAL_TRANSITS: 198.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 1664390.1745693285
-------------------------------------------------



Floors: 7
Time: 24 h
Control strategy: 4
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 3.452500000000001
AVERAGE_TRANSIT_TIME: 8.185555555555556
AVERAGE_ENERGY: 1905.6927015208873
AVERAGE_TRANSITS: 4.383333333333334
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 501197.1804999934
TOTAL_TRANSITS: 263.0
PEAK_WAITING_TIME: 19.0
EFFECTIVE_VALUE: 3114.801142573285
-------------------------------------------------
```

9.5.2.3 FloorCount = 9

```
Floors: 9
```

```
Time: 24 h
Control strategy: 0
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 17.53
AVERAGE_TRANSIT_TIME: 163.38479166666667
AVERAGE_ENERGY: 8976.888749741242
AVERAGE_TRANSITS: 3.6
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 2585343.9599254774
TOTAL_TRANSITS: 288.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 1.0284409932517512E7
-------------------------------------------------


Floors: 9
Time: 24 h
Control strategy: 1
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 13.883750000000001
AVERAGE_TRANSIT_TIME: 19.41354166666666
AVERAGE_ENERGY: 9917.967509505423
AVERAGE_TRANSITS: 3.65
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 2896046.512775583
TOTAL_TRANSITS: 292.0
PEAK_WAITING_TIME: 57.0
EFFECTIVE_VALUE: 514965.5810812095
-------------------------------------------------

Floors: 9
Time: 24 h
Control strategy: 2
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 88.66562499999998
AVERAGE_TRANSIT_TIME: 19.921875000000004
AVERAGE_ENERGY: 10851.512747845107
AVERAGE_TRANSITS: 1.45
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 1258775.4787500324
```

```
TOTAL_TRANSITS: 116.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 3088170.48744593
-------------------------------------------------

Floors: 9
Time: 24 h
Control strategy: 3
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 39.445416666666674
AVERAGE_TRANSIT_TIME: 21.191875000000003
AVERAGE_ENERGY: 10272.705375106007
AVERAGE_TRANSITS: 3.0
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 2465449.2900254414
TOTAL_TRANSITS: 240.0
PEAK_WAITING_TIME: 120.0
EFFECTIVE_VALUE: 2862394.636111238
-------------------------------------------------

Floors: 9
Time: 24 h
Control strategy: 4
-------------------------------------------------
### End result ###
-------------------------------------------------
AVERAGE_WAITING_TIME: 3.735416666666667
AVERAGE_TRANSIT_TIME: 8.143333333333334
AVERAGE_ENERGY: 1903.4693498287413
AVERAGE_TRANSITS: 3.65
TOTAL_TIME: 86400.0
TOTAL_ENERGY: 555813.0501499925
TOTAL_TRANSITS: 292.0
PEAK_WAITING_TIME: 29.0
EFFECTIVE_VALUE: 5674.848338237037
-------------------------------------------------
```

9.6 Documentation

Some javadocs dump of the simulator program’s major classes and important interfaces.

In order to view these mail Frederick or Alexandra to their respective email addresses:

Frederick:	Fceder@kth.se
Alexandra:	Alnordin@kth.se

Javadoc consists of:

- Cart
- CounterMass
- DirectionState
- Elevator
- ElevatorControlFramework
- ElevatorState
- Engine
- EngineOutputListener
- Entity
- EntityComponent
- EntityEnergyConsumer
- EntityFloor
- EntityFreightSystem
- EntityLoad
- EntityMass
- EntityMovingMass
- EntityUpdater
- Floor
- Passenger