Elevator Algorithm

Literature Review:

1. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fwww.cs.huji.ac.il%2F~ai%2Fprojects%2F2014%2FThe_intelevator%2Ffiles%2Freport.pdf&clen=297977&chunk=true
2. https://towardsdatascience.com/elevator-optimization-in-python-73cab894ad30
3. https://en.wikipedia.org/wiki/Elevator_algorithm
4. https://www.geeksforgeeks.org/smart-elevator-pro-geek-cup/

Offline Algorithm: the elevator will focus on one call until it reaches to its destination. The offline algorithm gets all the calls and their destinations so, generally the system will check if there is an available elevator if so, the system will send this elevator to the call. Then in the case when all elevators work, the system will look for the closest elevator to the call and check its destination and its direction, if the direction is the same the system will calculate the time for it to take this call, if it will be fast the elevator will take this call. In case that there is a farther elevator, but its destination is to this call, or if this elevator's time calculation for this call is faster the system will send this elevator.
The time calculation depends on:
1. Number of stops
2. Number of floors the elevator pass (by going up/down) before getting to the destination
3. Door opening + door closing
4. Wait time for each stop
5. Slow before stop + acceleration after stop
6. Velocity (before slowing for stop and after acceleration)

First the time for the elevator to open + stay + close, then we consider the time to slow before stop + acceleration after stop and take care of any other stops in the middle and finally multiply it all by the number of floors the elevator should pass. So,
Time = (destination floor – current floor) * velocity + ((number of stops) * (open + stay + close + Slow before stop + acceleration after stop))
The time calculation also takes care of the people inside the elevator by checking the destination floor and the number of stops needed.

Online Algorithm: In this algorithm we will break the problem into cases:
Case 1: All the elevators are "resting" at different floors then we will send the closest elevator. (Let us set the resting floors such that 1/3 elevators will rest at the middle floor of the building).
Case 2: Some of the elevators are moving, and some are "resting", then we will check if there is a moving elevator towards the floor (the calling floor is on the way, or the destination of the call) , and also check if there is a "resting" elevator close enough and then check for which one it will be faster.
Case 3: All the elevators are moving. In this case we will check all the elevators that are "resting" or in the correct direction and the current call is in a floor that in the range of the other calls.
In this case we will calculate according to these parameters:
* A – Acceleration and declaration, time to open, close.
* B – Number of stops.
* C – Number of floors.
* V – Velocity.

And we will use the next formula.
Let D be the destinations floors and S the source floors

A = acceleration + declaration + open + close
B = (position – source) / speed
Time = C + (A * B)
The sum is for each 2 floors without a stop between them(d for destination, s for source).

Simple Junit cases:

1. New call inserted to the right place in the calls list
2. New call enters to the end of the calls list
3. Up call enters – check if the system allocates the right elevator (same down)
4. Command while list of the allocated elevator is NOT empty
5. Command while list of the allocated elevator is empty
6. Enter a lot of up calls
7. Enter a lot of down calls
8. Enter up then down – check if elevator changes direction (shouldn't)
9. Enter a call where the source floor is low, and the destination floor is high
10. Enter few calls and check if the system allocates the fastest elevator

Cases table:

Case:	Total Waiting Time:	Average Waiting Time:	Number on incomplete calls	All results (long String)
0	218.9897426188186	21.89897426188186	1	Case,0, Total waiting time: 218.9897426188186, average waiting time per call: 21.89897426188186, unCompleted calls,1, certificate, -748934023
1	327.9897426188186	32.79897426188186	4	Case,1, Total waiting time: 327.9897426188186, average waiting time per call: 32.79897426188186, unCompleted calls,4, certificate, -1177892402
2	5831.792822120195	58.317928221201946	6	Case,2, Total waiting time: 5831.792822120195, average waiting time per call: 58.317928221201946, unCompleted calls,6, certificate, -1889034546
3	21386.538284333124	53.46634571083281	5	Case,3, Total waiting time: 21386.538284333124, average waiting time per call: 53.46634571083281, unCompleted calls,5, certificate, -1800748993
4	29768.455368642095	59.53691073728419	3	Case,4, Total waiting time: 29768.455368642095, average waiting time per call: 59.53691073728419, unCompleted calls,3, certificate, -2125107200
5	136319.12115705173	136.31912115705174	36	Case,5, Total waiting time: 136319.12115705173, average waiting time per call: 136.31912115705174, unCompleted calls,36, certificate, -4735726620
6	98202.88209694841	98.20288209694841	14	Case,6, Total waiting time: 98202.88209694841, average waiting time per call: 98.20288209694841, unCompleted calls,14, certificate, -3470000520
7	274169.12115705165	274.1691211570516	81	Case,7, Total waiting time: 274169.12115705165, average waiting time per call: 274.1691211570516, unCompleted calls,81, certificate, -5329741023
8	206270.8820969488	206.2708820969488	26	Case,8, Total waiting time: 206270.8820969488, average waiting time per call: 206.2708820969488, unCompleted calls,26, certificate, -6992173382
9	83036.34007431248	83.03634007431248	6	Case,9, Total waiting time: 83036.34007431248, average waiting time per call: 83.03634007431248, unCompleted calls,6, certificate, -2744399872