MedicineChest lift algorithm design

Matthew Bayfield

July 30, 2023

Contents

1 Defining efficiency **2**

2 Devising a strategy **2**
2.1 Initial considerations and assumptions 2
2.2 On the way to a solution 4
2.2.1 Static Scenario 4
2.2.2 Dynamic Scenario with no lift calls 5
2.2.3 Dynamic Scenario with lift calls 7
2.2.4 Complete Final Scenario 8

3 The Algorithm Outline **9**
3.0.1 Global variables, Events 9
3.0.2 A major function 9
3.0.3 The main loop 11

1 Defining efficiency

One way to measure the efficiency of the lift algorithm can be described as the ability of the algorithm to move the maximum number of people between floors in the shortest amount of time. This is a holistic measure of efficiency, in that it measures the total time to transport differing groups of people to a range of floors. However this measure of efficiency ignores the user experience, and the efficiency as measured by the distribution of journey times for individual users of the lift. Users expect the lift to operate fairly from their perspective as an individual, and are not necessarily concerned as to the benefit to the collective user journey times. Additionally users expect a degree of predictability when using a lift, that allows them to estimate when they likely will be able to disembark from or board the lift at a given floor. If a lift algorithm fails to be somewhat predictable, or at least fair — in that it guarantees a user will board or disembark at their desired floor within a reasonable amount of time — then it will lead to dissatisfied users, and perhaps a journey time distribution skewed towards higher values, if not at least for a significantly large minority of users.

Thus an optimal lift algorithm should aim to try minimise the total journey time and the mean/mode/median journey time for a user. Of course these two times will likely be correlated, since a lift performing unnecessary travel by going to floors when not required or in a way where a shorter route is possible for a fixed number of journeys to a set of floors, would naturally prolong the journey time for every user.

2 Devising a strategy

2.1 Initial considerations and assumptions

It is the case that people working in the offices are equally distributed across the 10 floors. The implications of this are that — ignoring for now any dependency on time of day and lift usage patterns — journeys to and from a given floor are equally likely relative to all other floors. In reality although not specified in the brief, certain floors will be more frequented than others depending on the need of workers to travel between floors as dictated by the purpose of each floor. If anything it would not be surprising if the most common journeys are between the ground floor and the floors above, since this is a journey necessary to leave and return to work. This then leads naturally in to the next factor in being the times at which people arrive and leave the building.

At times when most people are arriving to work (8am), pretty much all lift journeys will be from the ground floor to the upper floors, and vice versa at the end of the work day (6pm). This then will affect where the lift is when empty at these times of day, and also perhaps whether the lift will prioritise journeys from the ground to the upper floors and vice versa.

With no direct way to know with full certainty how many people are in the lift or waiting for the lift at every point in time, there is no way to fully incorporate the number of users travelling between two floors, and also the lift capacity in to the decisions made by a candidate algorithm. One way in which it is possible to use the lift capacity, is if eight floors have been selected to travel to or called; for then there is no point in stopping

to pick up additional users from further called floors, until some users have disembarked at a destination floor to make room.

A limitation of having only a single non-directional external lift call button, is that no information can be fed in to the algorithm indicating whether users calling the lift from a floor desire to travel up or down or both (if both buttons can be pressed). This then impedes the ability to make decisions with respect to route planning, and thus when to stop at the floor. At best a probabilistic assumption can be made as to the most likely direction of desired travel. For example assuming as was stated earlier that each floor — aside from perhaps at the start, end and middle of the day — is equally frequented, the relative number of floors above and below the called floor, will give a probability that the user is going up or down. If a user is on the 8th floor, it wil be assumed they are going down; if there are an equal number of floors above and below, then the downward direction will be assumed, because people will often be leaving the building throughout the day. At the start of the day it may be assumed that all journeys are up, whereas at lunch and the end of the day it will perhaps be assumed the direction of travel is downward.

It will be assumed that all floors are equally spaced, and that the average speed (v) at which the lift travels between adjacent floors is approximately constant (there will naturally be slight variations depending on the number of occupants), so that the time taken to travel between adjacent floors (Δt_{adj}) is constant. Additionally it will be assumed that the time during which the lift stops at a floor is approximately constant (Δt_{stop}). With these assumptions the total time travelled by the lift will be proportional to the number of stops and the number of floors travelled up and down. Thus the travel time between two floors numbered $F_{initial},\ F_{final}\ \in [1, 10]$, will be

$$\Delta t(F_{initial}, F_{final}) = n \cdot \Delta t_{stop} + |F_{final} - F_{initial}| \cdot \Delta t_{adj},$$

where $n -$ is the number of scheduled stops between the two floors.

If given a starting floor, and a known fixed set of destination floors with no additional calls occurring, then an optimal route could be calculated without the need for any dynamic adjustments to real time changes. Such a scenario is far easier to handle, and although possible on occasion during the lift operation, unfortunately is not the only scenario that will occur. Far more often it is likely that new destination floors and calls to new floors, will be added as the lift travels between and stops at its floors. This then will alter which route between floors is optimal, and may mean the current route taken thus far in hindsight is not the best. This of course is in part an inherent consequence of the unpredictability of the user actions as well as an incomplete knowledge of the common journey patterns. Decisions will need to be made as to how to update the route taken by the lift in response to real time changes. A further practical consideration in doing this, is the minimum response time of the lift and thus the update frequency of the algorithm. For example if the lift is about to pass a floor as it is called to that floor, it is not possible to stop the lift in time. Also there is a recalculation time required to re-optimise the route taken every time a change to the input data occurs. It is for these reasons that it has been decided to only update the route when the lift is stopped and is stationary at a floor; any new destination floors added by newly boarded users, or calls made since the last stop, will be used to re-optimise the route taken by the lift.

2.2 On the way to a solution

2.2.1 Static Scenario

In order to come up with an algorithm to optimise the efficiency of the lift, it is useful to first solve a simplified version of the problem and then modify the solution in response to increasing the problems complexity. Thus a solution will first be considered for a static scenario, where starting at some floor, all destination floors are known, and no calls occur or further new destination floors are added at each stop.

In this scenario, trying to minimise the total time taken for the lift to reach all destination floors will be a function of the route/path taken, namely the order each floor is stopped at. In this static scenario, if the lift starts at some initial floor $F_{initial}$, and given a set of destination floors $S = \{ F_i \}_{i=1}^{n}$, where $n \leq 9$, then a route or path through the floors corresponds to a permutation of n objects, of which there are $n!$. For each permutation of floors or path, the total number of floors traversed can be calculated as the sum of the difference of adjacent terms in each permutation, with the initial floor adjoined at the start:

The symmetric group S_n denotes the set of all possible permutations of n objects. Each path P corresponds to a sequence of floors starting with the initial floor.

$$\therefore \; P = F_{initial}, F_{\sigma(1)}, \ldots, F_{\sigma(n)}, \text{ where } \sigma \in S_n.$$

The total number of floors traversed D for a path P is then:

$$D(P) = \sum_{i=1}^{n} \left(P_{i+1} - P_i \right)$$

where $i + 1$ and i are sequence indices of the sequence corresponding to the path P.

In any path the total number of stops n is the same. Therefore from the perspective of minimising the total journey time of the lift, picking the path that has the smallest $D(P)$ will correspond to the shortest total journey time. However even in this static scenario the shortest overall journey time, may not correspond to the path that minimises the mean journey time of the individual user, which as mentioned before is another measure of the lift efficiency. (The mean journey time is proportional to the sum of the individual user journey times). To illustrate this consider the case where $F_{initial} = 5$, and where $S = \{ 1, 3, 9, 6, 10 \}$. By performing the calculations it can be shown that the route with the fewest floors traversed, travels 13 floors. The number of floors travelled for each user, for this same route, when summed is 40. The smallest sum of combined floors travelled for each user for a route is 36; and the total floors travelled for this route by the lift is 14. Therefore it is clear that although they will be correlated, the optimal route with respect to total lift journey time will not always be the optimal route for the mean user journey time. Note the stoppage time has not been included in estimating the user journey times. Also again it is worth stating that the distribution of user journey times may not be symmetric about the mean.

It is necessary to make a choice between our two types of lift efficiency. A route that travels more floors, and thus takes slightly longer to transport all groups of people to all n floors, may actually be better for the average user satisfaction. Thus where the route is not optimal for both types of efficiency, minimising the mean user journey time will be favoured.

The lift stoppage time will now be incorporated in order to calculate and minimise the total user journey time as a measure of the mean user journey time. It should first be noted that although we are trying to calculate the user journey time, in fact what we end up calculating is the time taken to reach and stop at a given floor since it was first selected as a destination. Thus this time will set the upper bound of the user journey time for a single user when travelling to a certain floor. The reason we cannot calculate the journey time for every user heading to a given floor, is because we do not know how many users are travelling to each floor, or whether (in the non-static scenario) some of the users boarded the lift when their destination floor has already been selected, thus leading to a relatively shorter journey time for those users.

Now that it is understood what is being calculated, the user journey time for reaching their destination floor, having selected it when boarding, and where their floor is the jth visited floor, corresponds to the jth partial sum (Q_j) of the sequence of the cumulative times elapsed, calculated at each stop along the lifts path. To be clear Q_j is the time taken for the lift to reach the jth stop along its route, since leaving its initial floor in this static scenario. The total user journey time as a function of the lift path ($T(P)$) is then the sum of all the n partial sums:

$$T(P) = \sum_{j=1}^{n} Q_j = \sum_{j=1}^{n} \Bigg(\overbrace{\Big(\sum_{i=1}^{j} |(P_{i+1} - P_i)|\Big) \cdot \Delta t_{adj}}^{\text{time travelling between floors}} + \underbrace{(j-1) \cdot \Delta t_{stop}}_{\text{total time stopped}} \Bigg)$$

The optimal path P with respect to minimising the total user journey time, is then the path that minimises $T(P)$. In the event that there exists multiple paths that minimise $T(P)$, then a path which minimises the total lift journey time can be chosen; this corresponds to a path that minimises Q_n. The expression for Q_n can be seen to contain a $D(P)$ term, with a term for adding the total time stopped.

2.2.2 Dynamic Scenario with no lift calls

Having derived a strategy for maximising the efficiency of the lift in the static scenario, the complexity of the problem shall be incremented by introducing the possibility that additional users may board the elevator during stops, and so select additional destination floors. This complicates things by introducing somewhat random changes, since there is no way of knowing which additional or how many floors will be added to the set of destination floors. In the static scenario choosing the optimum path was done when the number of stops was fixed from the start. In this new scenario as was discussed previously, the optimal route for the remaining floors will likely change. One option is to simply reset the route by effectively treating the current stop as the initial stop again, with a new fixed

set of destination floors, and optimising the path as if it were again the static scenario. A potential downside of this option is that it fails to account for the time elapsed for users who started on the actual initial floor, and who have yet to reach their destination stop. These users could possess a significantly longer journey time if the route deviates from the route leading to their stops. Therefore any increase in their journey time by deviating the route for the original yet reached stops, will need to be offset by sufficient reductions in the journey times of newly boarded users, so that the total user journey time for newly boarded users and existing lift riders is minimised. What's more even if a new route is chosen that minimises the total user journey time, as this addition of floors can happen at every stop, successive route deviations affecting the same users, could perhaps lead to unacceptable journey times for these users. Additionally if a user experiences many route deviations, particularly if the lift is changing directions many times, the lift will become unpredictable to the user who will not be able to predict when the lift will reach their stop. Ideally a scenario leading to such extended journey times for some users will not occur frequently.

One way to perhaps handle this extreme scenario, is to introduce some sort of term that results in the lift travelling to certain floors if they have not been visited for a long time. Although even this solution could itself pose problems, by significantly increasing the total user journey time, and also in effect causing the extreme scenario to become more common, requiring the behaviour enforced by this term to become more common. It does seem to be the case that for any strategy there will exist edge cases where the strategy is non ideal for some users. Ultimately in the absence of a completely provable/derivable best strategy, the actual practical efficiency of the lift resulting from a chosen strategy relative to some standard strategy, will have to be measured through lift testing and or simulations in order to see whether a better strategy exists. It is also apparent that there may be under certain circumstances a trade-off between lift efficiency and how predictable it is to a user.

It has been decided that at the cost of lift efficiency under some circumstances, that individual user journey times should be bounded, and thus in effect more predictable, in that after so many stops the lift will proceed to travel to those stops that have yet to been reached for this threshold number of stops, in an optimal way, before returning to its previous behaviour.

Consequently during most circumstances the devised option to re-optimise the route using the static scenario solution, with the initial floor becoming the current floor, will be used to periodically update the route at each stop, in a way that still optimises the mean user journey times. Therefore at each stop and after S has been updated by adding all new floor destinations, the time taken to reach each floor in S in a route must be calculated and all the times summed. This will be done using $T(P)$, remembering to replace $F_{initial}$ with $F_{current}$. Whilst the lift is following this strategy, the number of stops since a selected floor was selected will be counted; if at any point, the counts for a set of floors exceed this limit, the lift will alter its strategy by travelling only to these floors in an optimal way. It will then revert to its normal strategy. A limit of 9 floors travelled will be set, as this could correspond to the lift stopping at every other stop except the desired destination stop once. The optimal path for travelling to this subset of floors in this limit strategy will be determined using the static scenario solution.

2.2.3 Dynamic Scenario with lift calls

The next increment in complexity involves allowing users to call the lift from any floor. This again complicates the process of updating the route in response to real time changes, and also adds another source of user journey time, namely the user waiting time for the lift to stop for them. It subsequently introduces the need to balance the journey times of users already on the lift with the waiting times of the users calling the lift, whose journey time begins when waiting. If a user is made to wait excessively for the lift, regardless of how quickly their ride in the lift is, they will be dissatisfied. In contrast adding extra stops/floors to a user's ride in the lift, will increase their journey times.

It again should be emphasised that there is no way of knowing how many people are waiting for the lift to arrive on a certain floor, or whether they have been waiting the same time, or which destinations floors they want, or even the direction they are heading. All that can be tracked is the time since a user called the lift, and this then sets the upper bound of the waiting time of a single user.

All users except those that board the lift when empty, will have to wait in all likelihood some noticeable amount of time to board the lift, as it stops at other stops first. Thus rather than try to incorporate the waiting time directly in to the journey time of a user, the waiting times will attempt to be minimised separately. This will be achieved differently depending on whether there is a stronger probability that the user is travelling up or down, and if the lift is passing the called floor. There are two scenarios:

1.
   - The lift departs the current stop. Before travelling to the next scheduled destination stop, a called floor must be passed.
   - The subsequent destination floor after the next destination floor has the lift travelling in the same direction as it has been doing so.
   - The waiting user has a stronger probability of going to a floor in the same direction.
   - In this case it is fine to stop at the called floor, as there is a better chance the user is going to a next floor or a floor near to a next floor in the direction the lift is travelling. Their journey time and waiting will then be shorter, at only the initial cost of a single extra stop to other users.

2.
   - The lift departs the current stop. Before travelling to the next scheduled destination stop, a called floor must be passed.
   - The subsequent destination floor after the next destination floor has the lift travelling in the same direction as it has been doing so.
   - The waiting user has a stronger probability of going in the opposite direction.
   - In this case it is better to not stop at the called floor, as there is a better chance the user is going to a floor not in the direction the lift is travelling. Their increased waiting time will be equal to their journey time if they were picked

up; and a single extra stop for some other users will be removed.

It should be noted that relying on this probabilistic interpretation for predicting whether a user is going up or down will not work in each case, but for the long term average it should. The probability will be assigned based on the number of floors above and below the called floor. For example if there are more floors above, it will be taken that the user is more likely to be going up. This assumes as each floor is equally distributed, and in the absence of any other information, that there is an equal chance of movement between floors. If there are an equal number of floors above and below, then the user will be assumed to be going down.

For called floors that are skipped, they will not be considered to be added to the route again until the next stop. For called floors that are not being passed-by, they will be treated as if they were a destination floor in the static solution. Again like with destination floors, if a called floor has not been visited for 9 stops since called, it will be prioritised as was discussed previously. Finally if a called floor is already a destination floor, or becomes a destination floor, then two identical contributions involving this floor will feature in the total user journey time, as calculated in the static scenario; this makes sense as at least 2 users are affected. Having discussed how to handle called floors, the expression for $T(P)$ needs to be updated:

$$T(P) = \sum_{j=1}^{n} Q_j \cdot \eta(j) = \sum_{j=1}^{n} \left(\overbrace{\left(\sum_{i=1}^{j} |(P_{i+1} - P_i)|\right) \cdot \Delta t_{adj}}^{\text{time travelling between floors}} + \underbrace{(j-1) \cdot \Delta t_{stop}}_{\text{total time stopped}} \right) \cdot \eta(j),$$

$$\text{where } \eta(j) = \begin{cases} 2, \text{ if } P_{j+1} \text{ is a called floor and a selected destination floor;} \\ 1, \text{ otherwise.} \end{cases}$$

2.2.4 Complete Final Scenario

All that is left to incorporate in to the solution for the dynamic scenario with lift calls, is how the lift behaves when empty, and also how the lift capacity can be used to improve the efficiency.

Starting with the lift capacity, the only way to tell that definitely 8 people are in the lift is if 8 floors have been selected. If this occurs then called floors (the 2 unselected floors left, since there are 10 floors total) will not be stopped at, since no one can get on until some people have disembarked at a destination floor. Once fewer than 8 selected floors remain, called floors will be stopped at again. One issue with this is that the lift could still be full even with fewer than 8 selected floors, leading to the lift stopping at called floors, and users not being able to get on. To get around this issue, the lift weight sensor (which lifts have for controlling the lift pulling force) could be used to detect total weight changes, which would indicate whether users boarded. If no one boarded at a called stop, then the lift would be assumed to be full, and act as just described above. The called floor would not be removed from the set of floors to visit, as it would need to be revisited.

With regard to the lift behaviour when empty, this pertains to at which floor the lift waits. Through most times of the day, it makes sense for the lift to wait at floor 5 or 6, roughly half way, in that there are 5 and 4 floors either side. As each floor is equally distributed in people, with assumed equal movement between floors, a call to the lift could happen above or below with equal probability, and so its all about having an equal travel time to an above or below floor. At the start of the day at 8am, people begin arriving to work, and so for a certain time window starting then, it would make sense for the lift to wait at floor 1 (Ground floor). Likewise at the end of the day, for a time window before 6pm, it would make sense for the lift to wait at floor 10, as most people will be heading down to the floor 1, and the longest journey is from floor 10, with the potential to pick up users on the way down, also heading down.

3 The Algorithm Outline

Having now devised a strategy for optimising the lifts route and behaviour, I will now outline the structure of the algorithm, making use of the strategy, in its pseudocode form.

3.0.1 Global variables, Events

There will need to exist global variables tracking the sets of selected destination floors. Rather than just S, there will be S_{live} and S_{copy} denoting the real time updated set of selected floors, and a deep copy at a certain point in time respectively. Likewise there will be C_{live} and C_{copy} for the sets of called floors. There will also need to be sets of variables $\{ X_f \}_{f=1}^{10}$ and $\{ Y_f \}_{f=1}^{10}$, that count the number of stops since a called or destination floor were called or selected. Finally there needs to be a global variable indicating the current floor stopped at.

Events such as a user pressing a floor's lift call button, or users pressing buttons to select a floor as a destination, will need to be continuously monitored and used to update in real time S_{live} and C_{live}.

3.0.2 A major function

A major function will be
calculate_optimal_path(selected_floors, called_floors):
““““
Function to calculate an optimal path using the set of selected destination floors passed as a parameter, and the set of called floors passed as a parameter.
Returns:
A Path P as a sequence of floors.
”””
Function body:

1. If $selected\_floors$ is not empty, check for each floor $f \in selected\_floors$, whether $X_f \geq 9$; if so add f to a set named selected_limit_floors.

2. If $called\_floors$ is not empty, check for each floor $f \in called\_floors$, whether $Y_f \geq 9$; if so add f to a set named called_limit_floors.

3. If $called\_floors$ is empty:

   (a) If selected_limit_floors is empty, calculate $T(P)$ for each possible path involving the selected_floors. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |selected\_floors|$. Return calculated optimal P.

   (b) If selected_limit_floors is not empty, calculate $T(P)$ for each possible path involving the selected_limit_floors. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |selected\_limit\_floors|$. Return calculated optimal P.

4. If $called\_floors$ is not empty:

   (a) If selected_floors is empty:

      i. If called_limit_floors is empty, calculate $T(P)$ for each possible path involving the called_floors. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |called\_floors|$. Return calculated optimal P.

      ii. If called_limit_floors is not empty, calculate $T(P)$ for each possible path involving the called_limit_floors. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |called\_limit\_floors|$. Return calculated optimal P.

   (b) If selected_floors is not empty:

      i. If selected_limit_floors is not empty and called_limit_floors is empty, calculate $T(P)$ for each possible path involving the selected_limit_floors. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |selected\_limit\_floors|$. Return calculated optimal P.

      ii. If selected_limit_floors is empty and called_limit_floors is not empty, calculate $T(P)$ for each possible path involving the called_limit_floors. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |called\_limit\_floors|$. Return calculated optimal P.

      iii. If selected_limit_floors and called_limit_floors both not empty, calculate $T(P)$ for each possible path involving the selected_limit_floors. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |selected\_limit\_floors|$.

      For this optimum path P, extract the subset of called_limit_floors that must be passed-by. For this subset calculate whether the associated user on each floor has a stronger probability of going in the same direction as the lift in P, as discussed in the final strategy. Then extract the subset of floors of this subset, for which waiting users do have a stronger probability of going in the same direction as the lift. Of the floors left, if any, insert the first floor

to be passed-by in P, in to P in its position between the two consecutive floors in P. Return this P as the optimal path.

If there are only called_limit_floors not being passed-by in P, recall these will be treated as destination floors. Add them to an expanded selected_limit_floors set. Then calculate $T(P)$ for each possible path involving this expanded set. Extract paths which minimise T(P); from these choose any path that minimises Q_n, where $n = |selected\_limit\_floors| + |called\_limit\_floors|$. Return calculated optimal P.

iv. If selected_limit_floors and called_limit_floors both empty, repeat steps 4b(i), and 4b(ii), but replace selected_limit_floors with selected_floors, and replace called_limit_floors with called_floors.

3.0.3 The main loop

The code will need to contain a main loop, that starts when the lift stops at a floor:

1. Arrived/waiting at current floor.
2. Check if S_{live} and C_{live} are empty.
3. If both are empty, then check the current time.
4. $7.30am < current\_time < 9.30am$, then go to or stay at floor 1. $current\_time > 5pm$, then go to or stay at floor 10. Else go to or stay at floor 5. Lift will travel to or stay at floor, and so restart the loop with 1.
5. If S_{live} and C_{copy} are not both empty, open the lift doors.
6. If the current floor is in S_{live}/S_{copy} then remove it, and set $X_{current} = 0$.
7. Set $X_f += 1$, $\forall\ f \in S_{copy}$.
8. If current floor in C_{copy}:
   (a) If total weight has changed, remove the current floor from C_{live}/C_{copy}. Set $Y_{current} = 0$.
   (b) If total weight has not changed, indicate lift is full.
9. Set $Y_f += 1$, $\forall\ f \in C_{copy}$.
10. If doors open, allow time for users to board and disembark the lift. Close doors and wait 15 seconds or so for new users to select new floors if necessary.
11. Update S_{copy} by creating new deep copy of S_{live}.
12. Update C_{copy} by creating new deep copy of C_{live}.

13. If S_{copy} and C_{copy} are both empty, check the current time, and go back to step 4..

14. If $|S_{live}| \geq 8$, indicate lift is full.

15. If lift is full, call function $calculate\_optimal\_path(selected\_floors = S_{copy},\ called\_floors = \{\ \})$, to calculate path when not stopping at called floors.

16. If lift is not full, calculate path by calling function $calculate\_optimal\_path(selected\_floors = S_{copy}, called\_floors = C_{copy})$.

17. Using path P returned by *calculate_optimal_path*, move lift to P_2, the first floor in P after the current floor. Once arrived, loop restarts at 1.