Optimal strategies for an evacuation problem

Irwin, M.A.

M.A. Irwin

Optimal strategies for an evacuation problem

Bachelorscriptie, 17 juni 2009

Scriptiebegeleider: dr. F.M. Spieksma

Mathematisch Instituut, Universiteit Leiden

Contents

1 Introduction

From 2500 B.C. until 1311 the tallest structure in the world was the Great Pyramid of Giza. At construction the pyramid was 280 Egyptian cubits tall, equivalent to 146 meters. It is believed to have been built as a tomb for the Egyptian King Khufu (Cheops in Greek). This is in fact still not sure. What is sure though, is that everyday life in those days was spent on the ground. Today the tallest building in the world is the Burj Dubai at an incredible height of 818 meters! In fact, everyday taller and taller buildings are being designed and built. These buildings are no longer shrines or temples to be admired, but really serve a purpose. For instance, the Burj Dubai, although not yet completely finished, will consist of a hotel, private apartments and corporate offices.

One of the most important issues when building these massive structures is safety. People have to feel safe in order to be willing to live or work in such a supertall skyscraper. Therefore the chances of an accident occurring should not be (much) greater than in any other building. However, it is of course always possible that something will happen. Just take the events of 9/11 as a tragic example, or watch the movie 'The towering inferno'! In such cases a quick and clear evacuation strategy might be needed.

This study will make a start at solving the question:

> **For any given building: What is an optimal evacuation strategy?**

In order to answer this question we will analyze a simplified model of a random building that has just one lift. Chapter 2 will introduce some notation used in the model as well as the assumptions that were made to simplify the model. If an assumption needs clarification this will also be given. The simplified model has some intrinsic characteristics making it possible to analyze the problem. Chapter 3 introduces these characteristics, and proves their correctness. Using these characteristics an algorithm has been constructed that gives an optimal evacuation strategy for our simplified model. The details of this algorithm are given in Chapter 4. Some interesting results of our analysis are presented in Chapter 5, after which Chapter 6 will conclude with some discussion of these results. Also possible follow-up studies are discussed in this chapter.

2 The model

As mentioned in the Introduction we will only study buildings with one lift. In this study the numbering of the floors is done in the reverse order. So the top floor is numbered 0, and the ground floor is numbered n (where $n > 0$). This is done to simplify calculations. As a result a building consists of $n + 1$ floors.

To be able to study the model we will have to introduce some further notation.
First of all, we have the following three parameters of the model:

- p = time it takes a person to ascend or descend 1 floor
- l = time it takes the lift to ascend or descend 1 floor
- t = stopping period, i.e. time it takes the lift to open and close the doors

These parameters are considered to be known for each building.

Next we define the following decision variables:

- S_i = floor where the i-th stop is planned
- w_i = waiting time at the i-th stop
- $W(R)$ = total amount of waiting time under strategy $R = \sum_i w_i$
- $T(R)$ = total evacuation time under strategy R

And finally we have variables that are related to the building:

- HO_i = the highest occupied floor when planning stop i
- C_i = cutfloor belonging to stop i

These last two variables need some further explanation. As soon as we have planned where S_i will be and what w_i we can calculate from which floors it is possible to reach this floor in time to enter the lift. The lowest of these floors is called the *cutfloor*. For the remaining floors below S_i the new highest occupied floor HO_{i+1} can be determined.

Now that all parameters and variables have been defined it is possible to give a more precise definition of an evacuation strategy.

Definition 2.1. *An evacuation strategy R=$\{S,w\}$ decides where the lift stops and how long it waits at each stop.*

In fact, an evacuation strategy should also contain some information on where the people of each floor should go. However once the stops have been planned, everybody is automatically sent to the nearest stop. Therefore this information is not included. An evacuation strategy R is called an *optimal evacuation strategy* if it minimizes the time needed to clear the building.

2.1 Assumptions of the model

The problem has been studied under certain basic assumptions. Some of which have already been mentioned, but not formally stated as such. Here we will clarify which assumptions have been made, as well as some consequences of these assumptions.

a) *The number of people in the building as well as their location in the building is known.*

If this assumption is not made, a general evacuation strategy could still be determined. This strategy would presume that every floor is occupied. However this is not the goal of this study.

b) p *is constant.*

In practice, not everyone walks or runs at the same speed. Also, one could presume that it takes longer to ascend a flight of stairs than to descend this same flight. This is all not the case in this study however. Finally we do not take into account delays due to crowdedness. No matter how busy it is, a person's speed is always the same.

c) $p > l$

If this is not the case then the optimal strategy would be to just let everybody walk down. Therefore this is not really an assumption, but rather the only case we will study.

d) *The lift always starts at the top floor.*

This is in fact not really a necessary assumption, but merely done to simplify the calculations.

e) *The lift has an infinite capacity.*

A direct implication of this assumption is that it is not necessary to unload anyone at the ground floor and then return to pick up some more people.

f) t *is constant.*

No matter how many people want to enter the lift, t is always the same. This assumption combined with the fact that p is constant has a very important implication, namely

It is only important to know if a floor is occupied or not, not the number of people on a floor.

The last assumption we made in the model is

g) *Every time a person enters the lift a new stopping period of length* t *starts.*

In practice this means that it is not possible to sneak in while the doors are closing. If a person arrives at a floor where the lift is in the process of opening and closing the doors, then for him to be able to enter the lift a new process of opening and closing will have to be started. All the time that lift had already spent on this floor hereby becomes waiting time.

2.2 Characteristics of the model

From these assumptions some important characteristics of the model can be defined. These characteristics will make it possible to find an optimal evacuation strategy. Each characteristic is presented as a lemma.

Lemma 2.2. *For every evacuation strategy R with total evacuation time $T(R)$, there is a strategy R^* with all waiting time at the first stop S_1 and $T(R^*) \leq T(R)$.*

Proof:
Consider an evacuation strategy R with k stops, and waiting time $W(R)$.
Case $k = 1$: The statement is certainly true since $R^* = R$.
Case $k > 1$: If all waiting in R is done at the first stop then again the statement is trivially true. Let stop i be the first stop with waiting time $w_i > 0$, $i = 2, ..., k$. Now let strategy R' be the same as with w_i transferred to w_1 (so $w'_1 = w_1 + w_i$ and $w'_i = 0$). Since the departure time of the lift at S_i is the same in R and R' nothing changes for the floors below C_i. For stops S_j where $j = 1, .., i$ the departure time will be the same or w_i later, which means that the people from all floors assigned to S_j under R, will also be able to reach S_j under R' before departure. Therefore R' is also a feasible evacuation strategy with total waiting time $W(R') \leq W(R)$. We can repeat this process until we obtain a feasible strategy R^* where all waiting is done at the first stop S_1. □

One important consequence of this lemma is the following:

Theorem 2.3. *For every building there is an optimal strategy.*

Proof:
The total number of possible stopping strategies is $\sum_{k=0}^{n+1} \binom{n+1}{k}$ which is finite. For each stopping strategy we need only consider the cases where all waiting time is done at S_1. Because each stopping strategy has a minimum amount of waiting time needed to evacuate, there are only a finite number of evacuation strategies to be considered. Therefore there must be an optimal strategy. □

This proof is based on the fact that we can first fix the floors where the lift should stop, and then calculate the corresponding minimal amount of waiting time needed to evacuate the building. However it is also possible to fix the evacuation time first, and then determine all possible corresponding evacuation strategies. This fact is used in the following Lemma.

Lemma 2.4. *For every optimal evacuation strategy R with S_1 and w_1 there is also an optimal evacuation strategy R^* with $S_1^* = max\{k|(k - HO_1) \cdot p \leq k \cdot l + w_1, k = 0, 1, .., n\}$*

Proof:
Consider an optimal evacuation strategy R with S_1 and w_1.
If $S_1 = max\{k|(k - HO_1) \cdot p \leq k \cdot l + w_1, k = 0, 1, .., n\}$ then $R^* = R$.
Otherwise $S_1 < max\{k|(k - HO_1) \cdot p < k \cdot l + w_1, k = 0, 1, .., n\}$, because HO_1 must be able to reach floor S_1 within the w_1 waiting time. However, $S_1^* = max\{k|(k - HO_1) \cdot p < k \cdot l + w_1, k = 0, 1, .., n\}$ is also reachable for HO_1 within the w_1 waiting time. This means that all floors above S_1^* will be able to reach S_1^* in time. Because $S_1^* > S_1$ all floors i with $S_1^* < i \leq C_1$ have less distance to travel in R^*. This means they will definitely be able to arrive at S_1^* in time. For all stops and floors below C_1 everything is kept the same as in R. Because we have not changed the amount of waiting time or the number of stops, but merely replaced the stops, $T(R^*) = T(R)$ and R^* is also an optimal evacuation strategy. □

In perhaps simpler words, for every optimal strategy R with $W(R)$ we can plan S_1 as low as possible given w_1. The new strategy R^* is also an optimal evacuation strategy. As a result, when determining a possible evacuation strategy, we will always choose the stopping floors S_i, $i = 1, \ldots, M$, as low as possible given a number of stops M and a total amount of waiting time W. An important consequence of choosing the stops as low as possible is the following:

Lemma 2.5. *For any optimal evacuation strategy R^* from Lemma 2.4 with M stops and total waiting time $W(R^*)$ it is possible to distribute the waiting time in such a way that the lift waits until the people from HO_i reach S_i and then immediately continues to the next stopping floor.*

Proof:
This fact is due to the property that p is the same for everybody in the building. Suppose for some optimal strategy R^* constructed with Lemma 2.4 this is not possible. This would mean that the lift would have to wait at some stopping floor S_i after the people from HO_i have arrived at S_i. The only reason to wait any longer at S_i is to let some people from below C_i enter at S_i. If they do not enter there, the lift might as well wait at S_{i+1}.
The instant HO_i reaches S_i all other people will also exactly reach a certain floor. Nobody can be between floors. This means that if the lift would wait any longer, it would have to wait at least an amount of p time for someone from below to reach S_i. But in that case it is better to stop one floor lower, since the extra time the lift needs to wait for HO_i at that floor is $p - l$. This reduces the total waiting time by at least l. But this in contradiction with the fact that R^* is an optimal evacuation strategy.□

This last Lemma seems to be in contradiction to Lemma 2.2 a little bit. After all, Lemma 2.2 more or less states that all waiting should be done at the

first stop S_1, because this benefits everyone in the building. Now Lemma 2.5 states that this need not be the case. This is not true.
The main idea behind Lemma 2.2 is that transferring all waiting time in a given strategy to the first stop might result in an improvement, i.e. fewer stops or less total waiting time. In any case, the total evacuation time will never be larger. Lemma 2.5 on the other hand focuses on optimal evacuation strategies, therefore the total evacuation time can never be decreased by transferring all waiting time to the first stop.
A useful corollary of this lemma is the following:

Corollary 2.6. *For any optimal evacuation strategy R the total waiting time $W(R) = \sum_{i=1}^{M} (HOwt \text{ at } S_i^*)$, where S_i^* is the i-th stop in the transformed optimal evacuation strategy created using Lemma 2.4.*

3 The algorithm

3.1 Leiden's algorithm

Before presenting the algorithm proposed in this paper it is useful to first present another algorithm. This algorithm will be called the "0-waiting algorithm".

0-waiting algorithm

Initialize: $M = 0$

Determine HO_1

WHILE $((n - HO_{M+1}) \cdot p > n \cdot l + M \cdot t)$

- $S_{M+1} = max\{k | (k - HO_{M+1}) \cdot p \leq k \cdot l + M \cdot t, k = 0, 1, .., n\}$
- $C_{M+1} := 2 \cdot S_{M+1} - HO_{M+1} + 1$
- Send all floors between HO_{M+1} and C_{M+1} to S_{M+1}
- Determine HO_{M+2}
- $M := M + 1$

END(while)

This algorithm determines an evacuation strategy S where the lift does not wait at any of the stops. Essentially, it checks if the people from the HO can exit the building before the lift reaches the ground floor. Every time this is not the case an extra stop is planned and the corresponding cutfloor is determined. This leads to a new HO for which we now do the follow procedure. As soon as the people from HO are able to exit the building in time, the algorithm stops. Unfortunately, this algorithm will most likely not return an optimal evacuation strategy. Even though there is no waiting time, it is very probable that an improvement could be found by choosing a different stopping floor, where, after some additional waiting time, people from several floors can be picked up at the same time. Hereby the total number of stops could be reduced. If the gain of the reduction of stops is greater than the added waiting time, a better evacuation strategy has been found. Thus, finding a maximal number of stops required is essential to solving the problem.

This has resulted in creating a 2-step algorithm to solve the evacuation problem. A computer program with the exact details of the algorithm has been made with Matlab and is presented in the Appendix. In this paper only the basic idea of the algorithm will be discussed. Also it will be proven that the algorithm finds an optimal evacuation strategy.

Leiden's algorithm

1. (a) M=0
   (b) **WHILE** $((n - HO_{M+1}) \cdot p \geq n \cdot l + M \cdot t + t)$
   - $S_{M+1} := max\{k|(k - HO_{M+1}) \cdot p \leq k \cdot l + m \cdot t + t, k = 0, 1, .., n\}$
   - $C_{M+1} := 2 \cdot S_{M+1} - HO_{M+1} + 1$
   - Send all floors between HO_{M+1} and C_{M+1} to S_{M+1}
   - Determine HO_{M+2}
   - $M := M + 1$

   END

2. (a) $lowerbound = 0, upperbound = t$
   (b) **FOR** $i = 1 : M$
   - Determine HO_i
   - $s_i := max\{k|k \cdot l + (i-1) \cdot t + lowerbound > (k - HO_i) \cdot p\}$
   - $HOwt := max\{0, (s_i - HO_i) \cdot p - (s_i \cdot l + (i-1) \cdot t)\}$
   - Found = FALSE

   WHILE (!Found AND $HOwt < upperbound$)
   - Determine s_{i+1} up to s_M using '0-waiting'-algorithm
   - Determine HO_{M+1}

   IF $((n - HO_{M+1}) \cdot p \leq n \cdot l + M \cdot t + HOwt)$
   - upperbound := HOwt
   - Sopt := s
   - Found = TRUE

   ELSE
   - lowerbound := HOwt
   - $s_i := s_i + 1$
   - $HOwt := (s_i - HO_i) \cdot p - (s_i \cdot l + (i-1) \cdot t)$

   END(if)
   END(while)
   $s_i := s_i - 1$
   END(for)

The final step of the algorithm could be to transfer all waiting time to the first stop S_1. The way the algorithm constructs possible evacuation strategies and lemma 2.5 makes this extra step pointless.

In essence, the first step of this algorithm is almost an exact copy of the "0-waiting algorithm". The only difference is that instead of not waiting at all, an extra amount of t waiting time is added. The idea behind this is based on the following lemma:

Lemma 3.1. *For every evacuation strategy R with i stops and total waiting time $W(R) \geq t$ it is possible to construct an evacuation strategy R^* with $i+1$ stops and total evacuation time $T(R^*) \leq T(R)$.*

Proof:
The proof is quite trivial. From Lemma 2.2 we know that for every evacuation strategy R we can transfer all waiting time to the first stop. The resulting evacuation strategy R^* will never be worse than R. If the total waiting time $W(R) \geq t$ then adding a stop at any of the floors above S_1 without waiting will not increase the total evacuation time. In the worst case the lift will have to wait the remaining waiting time $W(R) - t$ at S_1.□

Obviously it is better to not just plan the extra stop anywhere, but try to plan it so that not all of the remaining waiting time $W(R) - t$ is needed.
As a result, step 1 of the algorithm returns the maximal number of stops M needed in an optimal evacuation strategy. There might be optimal strategies that consist of less stops. These will however require a large amount of total waiting time W. No optimal evacuation strategy will consist of more than M stops, since this will always increase the total evacuation time T.
Along with the maximal number of stops, an upper bound for an evacuation strategy with M stops is returned. The strategy constructed in the first step is the current optimal evacuation strategy is stored in an array called *Sopt*.
The second step in Leiden's algorithm is to determine the minimal amount of waiting time in an optimal evacuation strategy consisting of M stops. This is done by using a Branch and Bound-technique. Instead of considering all possible evacuation strategies with M stops, we define lower and upper bounds for the waiting time of an optimal strategy with M. The algorithm tries to lower the upper bound as far as possible until we have an optimal strategy.
One of the important parameters in the algorithm is $HOwt$. This is the amount of time the lift needs to wait for people from HO. As stated in Lemma 2.5 the instant that people from floor HO reach a stopping floor is crucial, since any additional waiting time at that floor is then pointless. Once we have fixed a stopping floor, the remaining floors can be seen as a new, smaller building which has to be evacuated.
It is possible for $HOwt$ to be negative. This is the case when the people from HO have to wait for the lift, instead of vice versa. For instance, this is always the case if the people from HO stay where they are. Obviously negative waiting time does not make any sense in practice. In these cases $HOwt$ is set to 0. Because we always choose the lowest floor where this is possible, this can only happen once though.
For each $HOwt$ a temporary strategy s is stored in an array. Every time it proves impossible to evacuate the building with a certain $HOwt > lowerbound$ amount of waiting time the lower bound is raised. This lower bound can be seen as a minimal amount of waiting time needed to evacuate the building with M stops. The lower bound is initially set equal to 0. As mentioned above, the first step results in $upperbound = t$.

The IF-check determines whether or not it is possible for all people that have not been sent to a stopping floor to exit the building before the lift reaches the ground floor. If this is the case, we have found an evacuation strategy with $HOwt < upperbound$ waiting time. The upper bound will then be set to $HOwt$ Also the corresponding strategy s is set to be the new optimal strategy S_{opt}. Every time the IF-check is not satisfied the lower bound gets raised until $HOwt > upperbound$. If no strategy is found in the WHILE-loop this does not imply that no improvement can be found. The algorithm uses '0-waiting' to determine the remaining stops. It might be possible though that just a little bit of extra waiting time will clear the building. This in fact also happens when a better strategy has been found, since the upper bound may still be greater than the lower bound giving us some time to 'play' with. Therefore after each WHILE-loop the short but very significant statement $s_i := s_i - 1$ has been added. If an improvement can be found this will have to be by stopping one floor earlier and then pick up the rest with a small amount of extra waiting instead of 0-waiting.

3.2 An example of Leiden's algorithm

An example might help to clarify the whole process even more.
We are given a building with the following parameters: $n = 100, p = 4$ and $t = 5$. In this case l has been set to 1. The top 25 floors of this building are shown in figure 3.

Figure 1: Top 25 floors of building with $n = 100, p = 4$ and $t = 1$

Step 1 of the algorithm would return $M = 4$ and the corresponding 't-waiting' strategy $S_{opt} = [4\ 15\ 34\ 73]$. The final HO in this case would be floor 98, and it is clear that people from this floor or below will have more than

enough time to leave the building before the lift reaches the ground floor.
Now step 2 of the algorithm starts. Floors 5 and lower will not have to be considered for S_1, since at floor 5 $HOwt = 7 > 5 = upperbound$. Also, floors 0 and 1 will not have to be considered for S_1, since floor 2 is the lowest floor where the people from HO_1 have to wait for the lift. The results of the first iteration of the second step of the algorithm are given in the table 1 below.

First stop	Strategy s	HO_5	Is HO_5 out	Lower bound	Upper bound
2	[2 9 20 42]	58	no	0	5
3	[3 10 23 49]	67	no	1	5
4	[4 15 34 73]	98	yes	1	4

Table 1: $HO_1 = 2$.

After the first step of the algorithm an evacuation strategy s has been found with $T(s) \leq T(S_{opt})$. The stopping floors are actually the same in both strategies, but the amount of waiting has been lowered. S_{opt} is therefore adjusted. If a further improvement of $Sopt$ is to be found this will have to be realized by making the first stop at floor 3. At this floor the lift will have to wait a period of 1 for the people from HO_1 to arrive, before proceeding to the next stopping floor. As a result we do not need to consider floor 9, the lowest floor where HO_2 arrives no later than the lift.

Second stop	Strategy s	HO_5	Is HO_5 out	Lower bound	Upper bound
10	[3 10 23 49]	67	no	1	4
11	-	-	-	1	4

Table 2: $HO_2 = 6$,$S_1 = 3$.

Because at floor 11 $HOwt = 4$, this does not need to be considered. There already is an evacuation strategy with $W = 4$. If an improvement of S_{opt} is to be found, this will have to been done by setting $S_1 = 3$ and$S_2 = 10$

Third stop	Strategy s	HO_5	Is HO_5 out	Lower bound	Upper bound
24	[3 10 24 51]	70	yes	1	2

Table 3: $HO_3 = 15$,$S_1 = 3$ and $S_2 = 10$.

In the final iteration the improvement would have to be found by setting $S_1 = 3, S_2 = 10$ and $S_3 = 23$. Since there is no floor where $lowerbound < HOwt < upperbound$, with $HO_4 = 33$, no strategy is constructed.
The optimal strategy $S_{opt} = [3102451]$ with $W(S_{opt} = 2$.

3.3 Proof of Leiden's algorithm

Now that the idea behind the algorithm is clear, it remains to prove that the algorithm finds an optimal solution.

Proof (Leiden's algorithm finds an optimal solution):
The proof is constructive.
Because of lemma's 2.2 and 3.1 there is always an optimal evacuation strategy R with M stops and total waiting time $W(R) \leq t$. Also this waiting time is done at the first stop S_1. (Every optimal strategy that does not possess one of these characteristics can be transformed into an optimal strategy that does possess these characteristics)
The first step of the algorithm chooses the stops as low as possible. Lemma 2.4 states that this is always the best decision. Therefore the algorithm will find the maximal number of stops M in an optimal evacuation strategy.
The second step considers all possible strategies with M stops and total waiting time between 0 and t. However, it only looks at the lowest possible stopping floors reachable within a certain amount of $HOwt$ waiting time. Since $W(R) \leq t$, this amount of waiting time is also considered by the algorithm. Due to the way the algorithm constructs an evacuation strategy and Lemma 2.4 it will find an optimal evacuation strategy with M stops and total waiting time $W(R)$.□

Finally a note on the complexity of Leiden's algorithm. It can be proven that M is $O(\log n)$. As a result step 1 is $O(\log n)$. Step (2) consists of a FOR-loop of M iterations with a WHILE-loop in it. The while loop will never make more than n loops. Therefore the algorithm seems to be $O(n \log n)$. However, due to the Branch and Bound-technique a great number of possibilities will not have to be considered. Therefore it might be the case that the algorithm is even better than $O(n \log n)$. In any case Leiden's algorithm seems to be able to find an optimal evacuation strategy in almost linear time.

4 Results

In the previous chapter a description of Leiden's Algorithm was given. As mentioned before, the exact code of this algorithm has been written in Matlab and is provided in the Appendix. The program has given us the opportunity to do multiple simulation runs and examine the optimal evacuation strategies. There were two main questions we wanted to answer by doing these simulation runs. These were,

1. How much improvement is obtained by doing step 2 of the algorithm ?
2. Can the optimal strategy be predicted beforehand, e.g. is there a formula for the stops ?

All simulation runs were done with several different parameters p, l and t. Since we are not working with real units we are only interested in the proportions of these parameters. Therefore, l is always set to 1, and only p and t are varied. Also the number of floors, parameter n, could easily be changed.

4.1 Analysis of Step 1 of Leiden's Algorithm

Since step 1 of the algorithm already returns a possible evacuation strategy, it would be nice to know something about the quality of this evacuation strategy. It is not difficult to understand that the evacuation strategy has an absolute performance guarantee of t. This means that the total evacuation time of an optimal strategy R_{opt} will never be more than t less than the total evacuation time of the evacuation time returned by step 1 of the algorithm R_1. Therefore it seems logical to presume that if t is small, than the difference between R_{opt} and R_1 will not be that great.

t-values	number of stops	percentage $T(R_{opt}) = T(R_1)$	percentage $W(R_{opt}) = 0$
$t = 1$	4	11,97	68,52
$t = 2$	3	40,43	26,18
$t = 3$	3	0,02	26,58
$t = 4$	3	0,08	29,65
$t = 5$	3	0,12	34,57
$t = 6$	3	0,79	49,55
$t = 7$	3	1,05	50,76
$t = 8$	3	0,76	49,97
$t = 9$	3	3,03	49,22
$t = 10$	3	12,08	49,34
$t = 11$	2	36,19	0,06
$t = 12$	2	0,06	0

Table 4: For each t: 10.000 runs with $n = 1000$, $p = 1.5$

As can be seen in Table 1 a strange phenomenon occurs. Initially the percentage of times $T(R_{opt}) = T(R_1)$ decreases, as was to be expected. However, this percentage starts to increase at $t = 9$ and $t = 10$. For different values of n and p a similar pattern can be witnessed. For example. Table 2, where $p = 3$, shows this pattern even more significantly.

t-values	number of stops	percentage $T(R_{opt}) = T(R_1)$	percentage $W(R_{opt}) = 0$
$t = 1$	8	31,11	68,89
$t = 2$	7	36,66	43,11
$t = 3$	7	4,51	34,92
$t = 4$	6	16,71	45,96
$t = 5$	6	22,43	23,73
$t = 6$	6	0,18	8,57
$t = 7$	6	0,57	15,26

Table 5: For each t: 10.000 runs with $n = 1000$, $p = 3$

A reason for these jumps seems to be that the number of stops needed to evacuate the building is lowered at these values of t. As t gets raised the advantage of making an extra stop gets smaller. This is due to the fact that stopping at a floor takes longer, and therefore walking loses less time. For several values of t the number of stops needed is the same, e.g. in case $p = 1.5$ we find that for $t = 2, 3, \ldots, 9$ three stops are needed. For $t = 10$ and $t = 11$ however we see that optimal strategies only need two stops.
The last columns in Table 1 and Table 2 show that it frequently occurs that no extra waiting time is needed. Therefore the error bound of t is quite often attained. In other words, the second step of the algorithm really does result in an improvement.

4.2 Analysis of optimal strategies

Another point of interest was to examine where the stops are mostly planned in optimal evacuation strategies. If some kind of pattern can be detected, this might make it possible to develop evacuation strategies which are more or less independent of the occupation of the building. Because the number of stops will be less in a small building than in a tall building, we have chosen to simulate unrealistically tall buildings, $n = 100000$. The reason being that this makes it easier to observe a pattern.

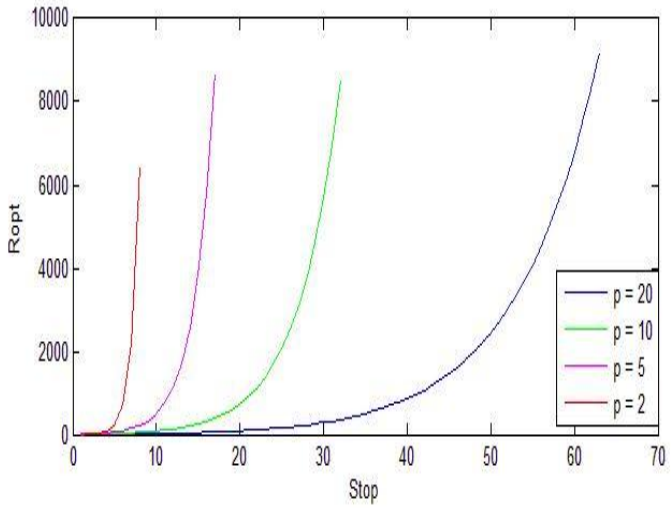

Figure 2: Optimal stop floors, $n = 10000$ and $t = 1$

In figure 1 the location of the stops in an optimal evacuation strategy R_{opt} have been plotted as a function of the number of the stop. For example, the stops in the optimal evacuation strategy when $p = 2$ were 0, 5, 20, 71, 232, 707, 2140 and 6435. The results seemed to suggest an exponential relationship between the numbers of the stops and the floors where the stops are planned. This was further examined by choosing several values of t, and then taking the logarithm of the stopping floors in R_{opt}.

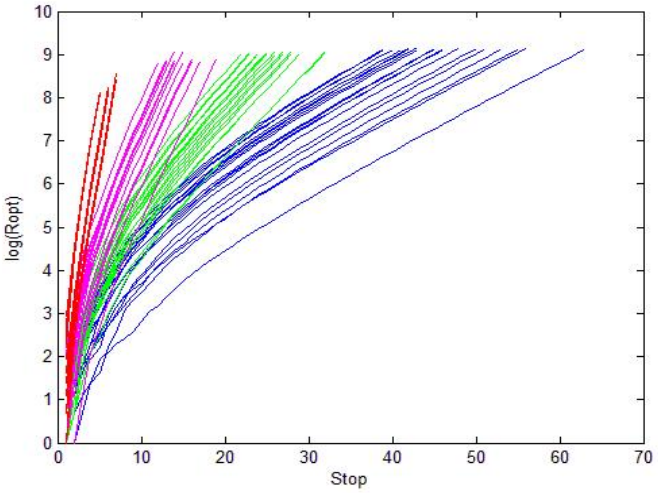

Figure 3: Logarithm of R_{opt}, $n = 10000$ and $t = 1, 2, \ldots, 20$

The results in figure 2 more or less confirmed the idea of the exponential relationship. Even more remarkable though, is the fact that the slopes of the graphs in figure 2 seem to be independent of t. Parameter p clearly does have an effect on the slope. As p is increased the slopes become less steep. However, when t is increased, while keeping p constant, the graph only shifts a little bit, but the slopes of the graphs eventually are the same. The values of these slopes, given in Table 3, indicate that as p is increased the slope tends to $\frac{2 \cdot l}{p}$. Again, since we are only interested in the relative proportions of l and p, this can also be seen as $\frac{2}{p}$.

p-values	slope of $\log(R_{opt})$
$p = 2$	1,100
$p = 5$	0,406
$p = 10$	0,201
$p = 20$	0,100

Table 6: Slopes of R_{opt}

5 Conclusion

This thesis is a first start at trying to answer the question:

> **For any given building: What is an optimal evacuation strategy?**

This has been done by making some basic assumptions which enabled us to make a simplified model of a random building. These assumptions were clarified in chapter 2. As a result of the assumptions certain specific characteristics could be witnessed. These characteristics were also explained and proved in chapter 2. For the simplified model Leiden's Algorithm has been constructed which constructs an optimal evacuation strategy for any random building. An explanation of the algorithm as well as a proof of its correctness was given in chapter 3. Finally Leiden's Algorithm was analyzed and some interesting results were set out in chapter 4.

It is clear that no definitive answer has been given yet to the main question posed. Unfortunately, a definitive answer might not be possible in reality. The biggest problem will always be that the occupation of a building is not fixed. People walk in and out of buildings all the time. Also within the building they are not confined to one floor, but travel from floor to floor. Even if it were possible to know the occupation of the building at any given moment exactly, it would still be virtually impossible to model every person's individual walking speeds. Making some basic assumptions is therefore more or less inevitable.
Still, Leiden's Algorithm is a starting point from which further analysis can be done. This analysis should focus mainly on dropping as many assumptions as possible. After all, every time an assumption is dropped the model becomes more realistic.
One of the first assumptions that should be dropped is the assumption that the lift always waits at the top floor. In fact, this will not require a major modification of Leiden's Algorithm since it only effects the location of the first stopping floor. The reason why this assumption needs to be dropped first is that if Leiden's Algorithm is modified in such a way that the lift can start from any floor, especially the ground floor, other assumptions might be dropped a lot easier. For example, dropping the assumption that the lift has an infinite capacity has a major implication. Now it is no longer possible to merely focus on occupied or unoccupied floors, but the number of people on an occupied floor will have to be taken into account as well. As a result the lift may have to drop off some people at the ground floor, after which it will have to go up again to pick up some more people. Each time the lift is unloaded at the ground floor we can consider the new situation as a new evacuation problem where the lift starts at the ground floor. Therefore it might be possible to find an optimal evacuation strategy by repeating Leiden's Algorithm several times.

The results set out in chapter 4 also warrant further analysis. The main focus here should be the fact that the stopping floors in an optimal evacuation

strategy resemble some kind of exponential structure of the form $e^{\frac{2 \cdot l}{p}}$. It seems logical that the ratio between l and p is important. Also, the factor 2 could well be due to the fact that people come from above and below to the stopping floors. The exact explanation for this structure is still unclear however. Even more of a mystery is the fact that the factor t does not seem to have an effect on this relationship.

One practical advantage of this insensitivity of t is that the algorithm could be that Leiden's algorithm is a good heuristic for real-life problems. After all, in general a stopping period will take longer if more people want to enter. Therefore the parameter t is most likely not constant. Leiden's algorithm is however not affected much by fluctuating t's.

Also, the relationship between the two steps of Leiden's Algorithm yielded some remarkable results. Especially the fact that the quality of step 1 of the algorithm is fairly erratic when t is varied has not been completely explained.

However, in order to answer the main question posed in this thesis further analysis of the results could be irrelevant. As mentioned before, Leiden's Algorithm is a first step in solving this question. The final answer might result in an algorithm that merely uses some properties of Leiden's Algorithm, but has some completely new properties of its own.

A Matlab code of Leiden's algorithm

```
clear all;
clc;

n=1000;
t=7;
p=3;
l=1;

building=unidrnd(2,n,1)-1;

HO=1;
while building(HO)==0
    HO=HO+1;
end
HO=HO-1;

esv=floor((HO*p)/(p-l))

%aantal stops bepalen
w=t;

numberofstops=0;
numberisknown=0;
highestoccupied=HO;

while numberisknown==0

     Stops(numberofstops+1)=floor((w+numberofstops*t+highestoccupied*p)/(p-l));

    if Stops(numberofstops+1)>n
        numberisknown=1;
    else
        n2=floor(2*Stops(numberofstops+1)-highestoccupied+1);
        highestoccupied=n2;
        while highestoccupied<n && building(highestoccupied+1)==0
            highestoccupied=highestoccupied+1;
        end


        if highestoccupied==n
            numberisknown=1;
        else
            if (n-highestoccupied)*p<=n*l+w+numberofstops*t
                numberisknown=1;
            end
        end

        numberofstops=numberofstops+1;

    end
end
```

```
Sopt=Stops'
numberofstops

lowerbound=0;
upperbound=w

h=HO*ones(numberofstops,1);
s=zeros(numberofstops,1);

for i=1:numberofstops

    s(i)=floor((h(i)*p+lowerbound+(i-1)*t)/(p-l));
    for counter=i+1:numberofstops
        s(counter)=floor((h(counter)*p+lowerbound+(i-1)*t)/(p-l));
    end

    lw=0;
    ew=(s(i)-h(i))*p-(s(i)*l+lowerbound+(i-1)*t);
    if ew<0
        ew=0;
    end

    found=0;
    while found==0 && ew<upperbound-lowerbound

        for j=i+1:numberofstops
            h(j)=floor(2*s(j-1)-h(j-1)+1);

            while h(j)<=n && building(h(j)+1)==0
                h(j)=h(j)+1;
            end

            s(j)=floor((lowerbound+ew+(j-1)*t+h(j)*p)/(p-l));
        end

        finalhighest=floor(2*s(numberofstops)-h(numberofstops)+1);
        while finalhighest<n && building(finalhighest)==0
            finalhighest=finalhighest+1;
        end

        if (n-finalhighest)*p<n*l+lowerbound+ew+numberofstops*t
            found=1;
        else
            s(i)=s(i)+1;
            lw=ew;
            ew=(s(i)-h(i))*p-(s(i)*l+lowerbound+(i-1)*t);
            if ew<0
                ew=0;
            end
        end
    end
```

```
    if found == 1
        upperbound=lowerbound+ew
        Sopt=s
        hopt=h
        if s(i)>esv
            s(i)=s(i)-1;
        end

        omlaag = 0;
        test=floor(2*s(i)-h(i)+1);
        while building(test)== 0
            test=test+1;
            omlaag = 1;
        end

        if omlaag == 1
            test = test-1;
        end
        h(i+1)=test;
    end

    lowerbound=lowerbound+lw

end

Sopt
waitingtime=upperbound
```