GENETIC ALGORITHM FOR CONTROLLERS IN ELEVATOR GROUPS: ANALYSIS AND SIMULATION DURING LUNCHPEAK TRAFFIC

P. Cortés[1†], J. Larrañeta[1] and L.Onieva[1]

[1] Seville University
Ingeniería Organización.
Escuela Superior Ingenieros, Camino de los Descubrimientos s/n.
Sevilla 41092. SPAIN
Tel. +34 95 448 72 05
Fax +34 95 448 73 29
[†] E-mail: pca@esi.us.es
[†] URL: http://io.us.es/P.Cortes/main.htm

Abstract.- A genetic algorithm (GAHCA) is proposed to control elevator groups of professional buildings. The genetic algorithm is compared with the universal controller algorithm in industry applications. In other to do so an ARENA simulation scenario has been generated during heavy lunchpeak traffic conditions. The results allow us to affirm that our genetic algorithm reaches a better performance attending to the system waiting times than traditional duplex algorithms.

Keywords.- vertical traffic, genetic algorithm, elevator, controller, simulation, lunchpeak.

1. INTRODUCTION

The progressive price increase in the urban centres of the larger cities makes the necessary intensive ground exploitation by means of the construction of high buildings. Today the installation of synchronized elevator groups in professional use buildings (offices, hospitals or hotels) is an usual practice.

The elevator system research is quite recent and has followed the technology development. The late eighties and the nineties decade can be considered as the start point of the industrial investigation, especially in USA and Japan ([1], [2] and [3]). After that the research experimented the impulse of the largest multinational companies ([4], [5], [6] and [7]). By the end of the nineties the research in vertical transportation was a reality and the collaborations among the private companies and the research centres were reinforced, some examples are the Systems Analysis Laboratory in the Helsinki University of Technology with the KONE Corporation [8], the Konrad-Zuse-Zentrum für Informationstechnik of Berlin [9] or the Seville University with MAC PUAR, S.A. [10].

In elevator systems the use of the system waiting time is the priority objective to attain an efficient system performance, at the same time as having a bounded maximum waiting time. The system waiting time includes the waiting time for the lift in the hall plus the trip time inside the lift. Also, other secondary criteria are used as the queue sizes or the system energetic consumption.

The more general problem assumes the following hypothesis in the elevator system performance. Each hall call is attended by only one cabin. The maximum number of passengers being transported in the cabin is bounded by its capacity. The lifts can stop at a floor only if it exists a hall call or a cabin call in that floor. The cabin calls are sequentially served in accordance with the lift trip direction. A lift carrying passengers cannot change the trip direction.

Usually, the controller implements dispatch rules that make use of an IF-ELSE logical commands set. Among these dispatch rules, a simple lift group supervisory control system, suitable for groups of two or three in not very high rise buildings, is simulated in the computer-aided design suite LSD (Lift Simulation and Design), implemented at UMIST (University of Manchester Institute of Science and Technology), under the designation of the THV algorithm [11]. This algorithm collects the most common rules in duplex or triplex algorithms. The THV algorithm assigns the hall call to the nearest lift in the adequate trip direction (see appendix 1 for pseudocode).

Recently, more advanced methods have gained better performance. So, the *Optimal Routing algorithm*, the *Dynamically Adaptive Call Allocation* (DACA) and the *Adaptive Call Allocation* (ACA) [12] are all based on Dynamic Programming. Also, previous research related to Soft Computing techniques in elevator systems has been responsible of important advances.

For example, algorithms based on learning have been developed with success. The controller *Neuros-I* [13] of Fujitec is a neural network where the group elevator state and the lifts state are inputs for the neural network. The network has a previous learning and subsequent adaptive auto-tune online learning. Also, in the framework of the learning, Reinforcement Learning algorithms [14] have shown an accurate behaviour. It consists of a semi-Markovian process and uses an agent-team where each agent controls one lift. Under these conditions two architectures are used: a parallel architecture where the agents share the network (RL_p, Parallel Reinforcement Learning) and a decentralised architecture where each agent have its own network (RL_d, Decentralized Reinforcement Learning).

Fuzzy Logic has been proved as a valuable alternative when evaluating a large amount of criteria in a flexible manner. The fuzzy elevator group control system [15] and the Fuzzy Elevator Group Controller with Linear Context Adaptation [16] are some examples where diverse criteria are used as the $HCWT_i$ (Hall Call Waiting Time for the i-lift), the $maxHCWT_i$ (maximum Hall Call Waiting Time), the CV_i (capacity of coverability for next calls for the i-lift), and the minimum distance between new calls and the last calls allocated GD_i (Gathering Degree). Also in this line, genetic algorithms [17] and [18] have been used with success to adjust the control settings (a set of criteria) in order to give robustness to the elevator group control system, within a set of great variety of control parameters. These works allow adjusting the control settings according to individual floor utilization situations making use of a combination of car and floor attributes.

Also evolutionary systems have revealed successful capacities in order to maximize the efficiency of the elevator system call allocation. Genetic algorithms [19] and [20] have been designed within a discrete event simulation trying to predict the optimal decisions for the car dispatch. Both are short-papers with a non-wide explanation of the methods

and with an additional difficulty when trying to identify the criterion used for assessing the quality of the solutions (by means of a performance index). However the authors state the validation and success of the implementation by the representation of diverse figures and graphics. Also, in this paper we have developed a genetic algorithm to maximize the call allocation efficiency and to reduce the overall system waiting time. Here, we propose a genetic algorithm based on a hall call allocation strategy (GAHCA) to identify the chromosomes of the population individuals and we compare our proposal with conventional duplex controllers of the industry in a discrete events simulation scenario.

As the elevator systems include uncertainty due to the future behaviour of the passengers is unknown, optimisation approaches are not totally suitable. Instead of this, the simulation becomes a key factor to demonstrate the validation and accuracy of the methods and techniques as previous step to the physical implementation (see [21] for a wide perspective).

The rest of the paper follows with the second section dealing with the simulation model definition to specify the accurate elevator system performance according to the rules previously stated. The third section states the genetic algorithm characteristics. The fourth section shows the main results of the simulations and the comparison between our algorithm and the traditional duplex algorithm. Finally, we highlight the main conclusions in the final section.

2. SIMULATION MODEL

We have made use of the ARENA v.5.0 software to simulate the possible event set. ARENA is a powerful interactive visual modelling system that makes use of the SIMAN language. The initial model consists of an animation zone and a module logical zone that can be divided into one controller zone, one passenger zone and two elevator zones for each of the cabins. The controller, passenger and elevator zones are the responsible of the IF-ELSE rules that manage the group elevator system. The optimisation algorithm (that we will see in section 3) is called in the passenger zone for the call allocation.

2.1. Animation Zone

This zone is defined by the `Arrive` and `Depart` modules, which regulate the arrivals and departures of the passengers at the system.

The `Arrive` modules include the passenger arrival rate in the floor, the passenger arrival time (loaded into the `Time_Arrival` attribute), the passenger origin floor (loaded into the `Origin` attribute) and the passenger destination floor (loaded into the `Destination` attribute).

The `Depart` modules carry out the increase of one unit in the floor departure counter. Also they include the `Time_System` as a tally buffering the passenger system waiting times, as well as two queues defined by floor (one for passengers going down and one for passengers going up, at exception of the ground floor and the highest floor where only one queue exists).

We are attaching one videoclip for each of the simulated algorithms (THV and GAHCA). The videoclips include the simulation under the traffic and building conditions of section 4.1. Moreover, the graphical animation zone can be observed in videoclip1 and 2.

"Video clip 1. Genetic algorithm.wmv"

"Video clip 2. THV duplex algorithm.wm

2.2. Controller Zone

One entity has been created by lift to travel around the logical zone. When the passengers come into the lift, the passengers are joined to the `controller` entity shaping one only entity at the same time as holding all the particular individual entities attributes.

2.3 Passenger Zone

The passenger zone consists of the allocation of the `UpDown` attribute (1 if the passenger goes up and 2 otherwise) that is stated as function of the `Origin` and `Destination` attributes. So, the passenger is sent to the waiting queue if it exists. Otherwise the hall call allocation procedure is done by means of the correspondent optimisation algorithm (our genetic algorithm by the case). The next figure 1 represents the controller and passenger zone ARENA modules.

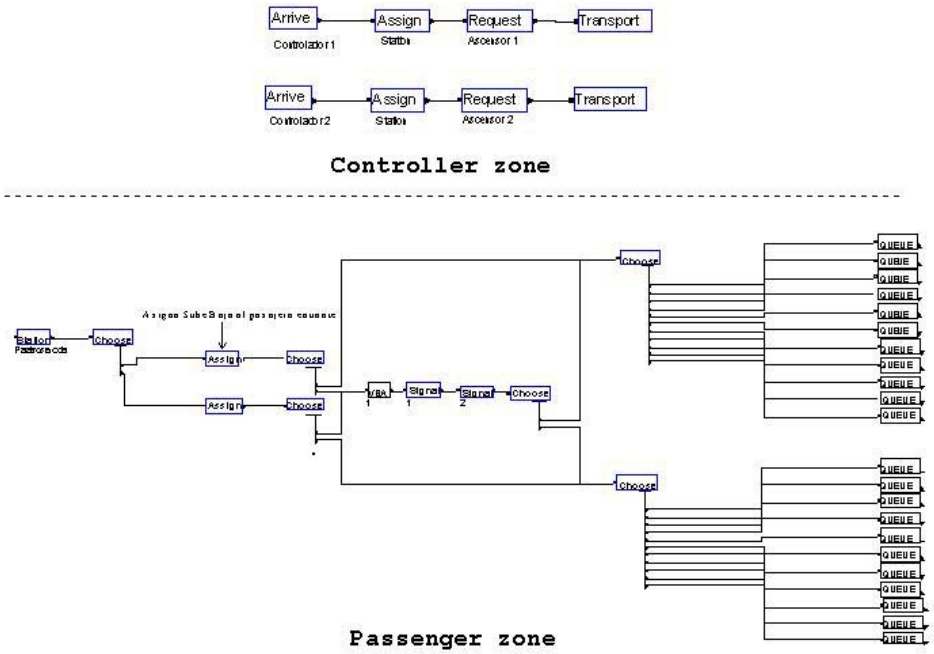


Figure 1. Controller and Passenger Zone ARENA modules

2.4. Elevator Zone

When the lift arrives at a floor the subsequent actions must be checked and done if necessary: lift waits for calls, passengers leaves the lift, passengers come into the lift, lift allocation in case of full capacity, cabin call allocation and call evaluation.

When the lift arrives at a floor, the `state` of the lift is evaluated. If the lift `state` is set to zero, the lift is stopped and will have access to the `Waiting_for_Calls` submodule. If the lift is not stopped and it is carrying passengers, it inputs into the `Leaving_the_Lift` submodule. If it is not carrying passengers, it inputs into the `Taking_Passengers` submodule after a `Delay` to simulate the opening doors time (we use the delay variable `Time_Doors` (2.5 seconds).

When the lift arrives at a floor, the `Arrival_Evaluation` submodule presents three options: the lift continues up, the lift continues down or the lifts starts the deceleration process (preparing to stop). We use the `LDX (Transporter ID, unit number)` as an ARENA proprietary variable allowing to know the floor in which the lift is. We load this data in the variable `Level`.

After updating `Level`, if the lift is going up and the lift is in the ground floor, the lift is sent to the first floor; if the lift is going down and the lift is in the highest floor, the lift is sent to the last but one floor. Otherwise the simulation model checks if the lift has stopped in the floor that `Level` indicates (this data can be checked by means of the variable `Last_Visited_Floor`), in this case the lift is sent up or down depending on the trip direction. Otherwise the lift stops at the floor if there exists a cabin call or a hall call and the capacity is not full. If the lift is full capacity and it does not exist cabin calls, the lift is sent up or down depending on the trip direction. Next figure 2 depicts the main ARENA modules and submodules for the elevator zone.

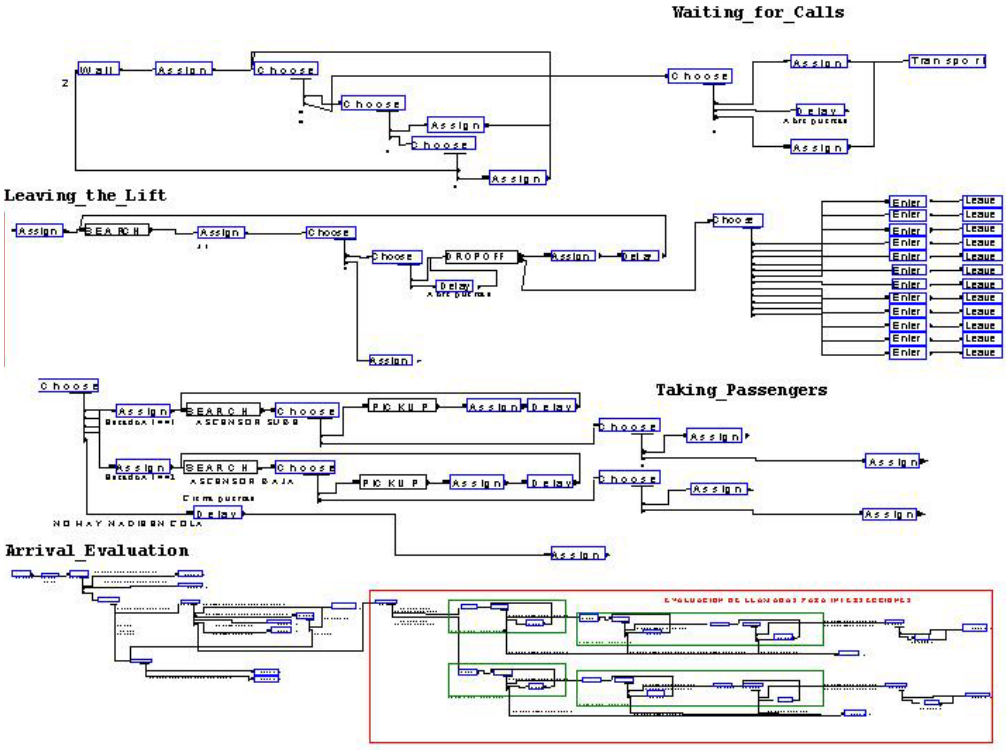


Figure 2. Elevator zone ARENA module

3. GENETIC ALGORITHM FOR THE CONTROLLER

We propose a genetic algorithm that makes use of a hall call allocation strategy (GAHCA) to perform the elevator group controller (see appendix 2 for pseudocode). For each time, *t*, the hall calls and the cabin calls of the system are evaluated, allocating the hall calls to one specific lift. Each time, *t*, the set of hall calls are reallocated allowing the subsequent modification if the system performance improves. Each time the set of decisions is taken managing all the available information (planning for the long term) but only carrying the immediate action out for each lift of the group: stop, upwards or downwards displacement.

So, each time, *t*, the simulation model makes a call to the controller optimization module (the genetic algorithm) that returns the overall call allocation. The genetic algorithm is defined by the following characteristics.

3.1. Individuals and population

Two arrays of size [2·`Number_of_Floors`-2] define the individual chromosome. Each of the arrays defines the system `state` for each one of the lifts. The array is divided into two parts; the first refers to the up traffic and the second one to the down traffic.

The first `Number_of_Floors`-1 integers correspond to the hall calls in the upward direction from the ground floor to the highest floor. The second `Number_of_Floors`-1 integers correspond to the hall calls in the downward direction from the highest floor to the ground floor. Figure 3 depicts the chromosome individuals:

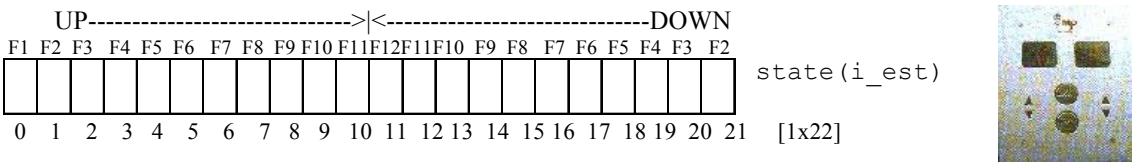


Figure 3. Individual chromosome for a twelve floors case building corresponding to one specific elevator of the group and its associated physical button box

The array holds the information referring to the hall calls by means of a binary codification. The bit 0 indicates no hall call at the floor, and the bit 1 indicates an existing hall call at the floor.

The population size is a major factor in the effectiveness of genetic algorithm. It has been proved [22] that relatively small populations allow reaching successful solutions with little computation effort. Our experiments show that increasing the population size beyond 20, although increasing the computational effort, is not rewarded by a corresponding increase of performance. So, we have maintained a population size of 20 individuals in our tests, although in real implementations this population size could be reduced to ten in order to gain in computational speed (with little loss of efficiency).

3.2. Fitness

We have used an approximate function (in seconds) to estimate the individual fitness. The fitness function returns the expected time in which the elevator group would serve the entire allocated hall calls and cabin calls. Obviously, it will be estimation because of the incapability of predicting the passenger future behaviour. The passenger arrival to the floor is random and their destinations are unknown.

The fitness estimation procedure depends on the elevator state (going up, down or stopped). However in every case it can be calculated by means of four peak values that we will note as P1, P2, P3 y P4.

Every time, the procedure has in account the overall allocated hall calls stating each new hall call to be allocated. Figure 4 shows the options depending on the up or down traffic.

The elevator is stopped or going up

- P1. Current floor.
- P2. Highest floor to take passengers up. In figure 4: displacement *a.*
- P3. Lowest floor to take passengers down. In figure 4: displacement *b.*
- P4. The highest floor among the floors lower than P1 to take passengers up, always P4<P1. In figure 4: displacement *c*.

$$Fitness = [(P2-P1)+(P2-P3)+(P4-P3)]\times[\text{estimated interfloor trip time}]$$

It includes the maximum known upward trip plus the maximum known downward trip plus the subsequent maximum known not-served upward trip in the first up traffic because P4<P1. We have to note that the mathematical expression do not include the passenger destination trips because we unknown it until the passengers come into the cabin.

The elevator is going down

- P1. Current floor
- P2. Lowest floor to take passengers down. In figure 4: displacement *a.*
- P3. Highest floor to take passengers up. In figure 4: displacement *b.*
- P4. The lowest floor among the floors higher than P1 to take passengers down, always P4>P1. In figure 4: displacement *c.*

$$Fitness = [(P1-P2)+(P3-P2)+(P3-P4)]\times[\text{estimated interfloor trip time}]$$

It includes the maximum known downward trip plus the maximum known upward trip plus the subsequent maximum known not-served downward trip in the first down traffic because of P4>P1.

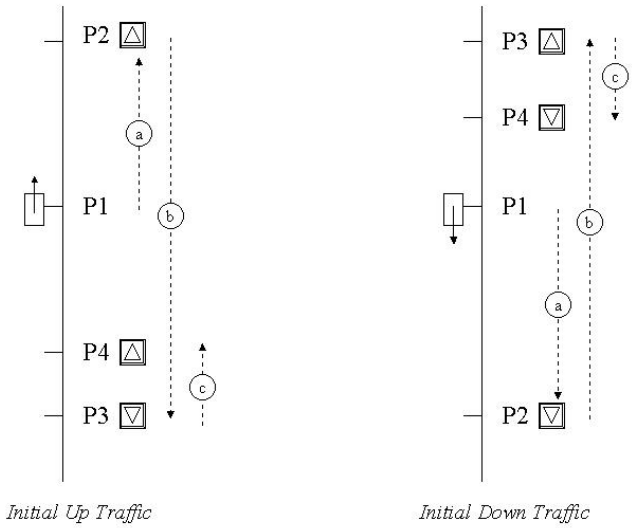


Figure 4. Possible elevator streams to estimate the fitness

Additionally, we have to consider a series of delays to estimate the total process times. All them are associated to the cabins and include the deceleration process in the elevator travel speed, the opening doors delay, the passenger incoming/outcoming process, the closing doors delay and the acceleration process in the elevator travel speed until taking the cruiser speed. Usual values are 2 seconds for every delays excepting for the incoming/outcoming time (taking 5 seconds for this delay). Moreover, we are taking 5 seconds as estimated interfloor trip time at cruiser speed.

3.3. Operators

The genetic operators used are crossover and mutation. We have used an uniform crossover operator that randomly selects two individuals (parents) from the population and generates the offspring by crossing the individual genes. The offspring inherits an exact copy of those genes that are equal in the parents' chromosome and, in other case; it inherits each gene with probability of 50%. Although the parents' selection is random, the algorithm includes an incest prevention control when parents differ in less than a gene pair. The mutation operator replaces a hall call allocation from the individual chromosome by changing the genes from 01 to 10 or viceverse. The selection of the individual is random.

Tests were carried out with different probabilities of applying crossover and mutation to the select parents. With crossover, it was found that varying the probability from 50% to 100% had little effect on performance, with a value of 85-90% being marginally optimal for the tests carried out. A value of 85% is used in the main runs. For mutation, values between 5% and 15% were seen to be giving better results than typically smaller values. A value of 15% is used in the main runs in order to enrich the genetic variety of the population. However, it is to be noted that GAHCA is robust in the sense that the solutions to the test problems were achieved on the whole with a wide range of parameter values, and with no fine-tuning required to achieve efficiency.

3.4. Replacement rule and generations

We propose the use of a hypergeometric function allowing more probability of replacement for individuals with worse fitness and less probability of replacement for individuals with better fitness. So, the individual in ranking position-i, have a replacement probability equal to $q(1\text{-}q)^i$, being q the replacement probability of the worst individual. We obtained the better performances setting a value for q between 55-65%. The main tests are run with a value of 60%.

Additionally to the replacement rule, we incorporate an individual duplicity control in the population generation.

The number of generations (or iterations) of the genetic algorithm can be a critical parameter when we try to reach efficiency of the solution and short time execution. Genetic algorithms are iterative and therefore they can take very much time of execution. In real cases the number of generations will be bounded by the exigencies of the real implementation. However the advantage of genetic algorithms is that they can be stopped at any time having the better solution at the moment. We experimented with diverse parameters for the number of generations: similar values were obtained for the interval between 50 and 100 iterations and the increases on the quality of the solutions were moderated between 20 and 50. In any case, at least 20 iterations should be done.

4. SIMULATION RESULTS

4.1. Data for the tests: building and lunchpeak traffic

We have tested the algorithms in a twelve floors building. There are 30 workers in each of the building floors excepting the 7th floor (the administration department with 60 workers) and the 12th floor (the manager department with 15 workers). There are two 20 persons capacity elevators in the hall.

The interfloor travel probabilities are defined within a lunchpeak traffic situation:

From the ground floor:
- To the 7th floor: 15%
- To the 12th floor: 4%
- To the rest of the floors: 9%

From other floors:
- To the ground floor: 95%
- To the rest of the floors: 5%

The next figure 5 depicts the arrival rate during lunchpeak traffic. Most of the workers go out for lunch during the interval [14:00,15:00] hours, returning to the building during [15:20,16:00] hours. We have taken these data from direct real life inspection in a such case building.

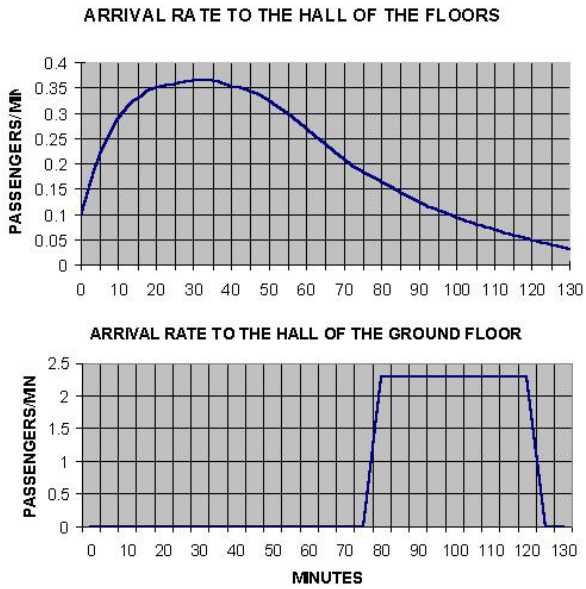


Figure 5. Arrival rate to the halls

It is important to note that lunchpeak traffic is the most critical situation in vertical traffic, because it includes the uppeak and downpeak traffic effects.

4.2. Comparison of algorithms

Our genetic algorithm has been put in competition against the well-known THV duplex algorithm. We have simulated 20 replications and we have analysed the system waiting time, the queues in the halls of the floors and the state of occupation of the elevators.

System waiting time

The table 1 summarises the analysis of passengers system waiting time.

Table 1. System waiting time analysis

ARENA Simulation Results
PCA - License #8910593
Summary for Replication 20 of 20

Project: THV DUPLEX lunchpeak Run execution date: 16/07/2002
Analyst: PCA Model revision date: 16/07/2002
Replication ended at time: 7800.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Time_System	195.60	11.157	20.194	2899.3	44239

Project: GENETIC ALGORITHM lunchpeak Run execution date: 16/07/2002
Analyst: PCA Model revision date: 16/07/2002
Replication ended at time: 7800.0

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
Time_System	149.98	(Corr)	20.333	662.75	44237

The average system waiting time is 195.60 seconds for the THV algorithm. Every replication holds the minimum waiting time between 20 and 25 seconds. However it is worthwhile to highlight the maximum waiting time reaching 2899.3 seconds, which cannot be considered an isolated peak moreover. This is a heavy value and represents an approximation to the worst-case eventuality for the case: a building with 375 workers and two only lifts during the lunchpeak traffic.

The THV maximum waiting times are due to several reasons. Firstly the THV takes the higher floors passengers down, after that the algorithm takes the rest of passengers in the other floors downwards. This phenomenon increases the waiting in the lower floors heavily. Moreover, the lunckpeak traffic refocuses this characteristic. During the lunchpeak traffic, the lifts saturate their capacity in the upper floors being not capable of taking additional passengers in lower floors. Afterwards, the lifts would come back to the top to take new passengers and the same phenomenon is repeated.

GAHCA reduces the system waiting times in a considerable form. The average waiting time is reduced from 195.60 seconds to 149.98 seconds. It corresponds to the 23.32% reduction. In other line, the maximum waiting time is drastically reduced to 662.75 seconds.

The next figure 6 shows the system waiting times for the genetic and the THV algorithms. The graphics depict the waiting times by floors as well as the average value. Note how the THV results are distributed between 0 and 1000 seconds, and the genetic algorithm results are distributed between 0 and 700 seconds.

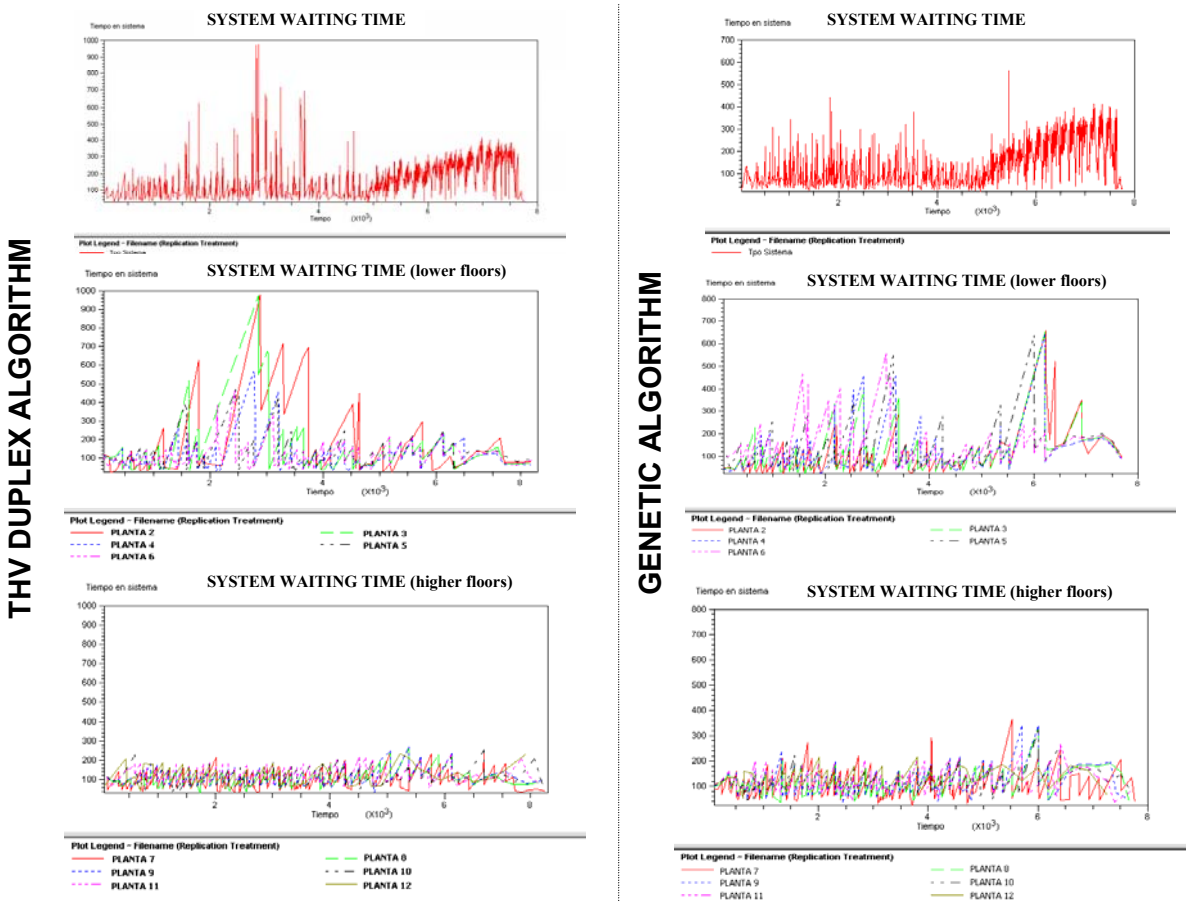
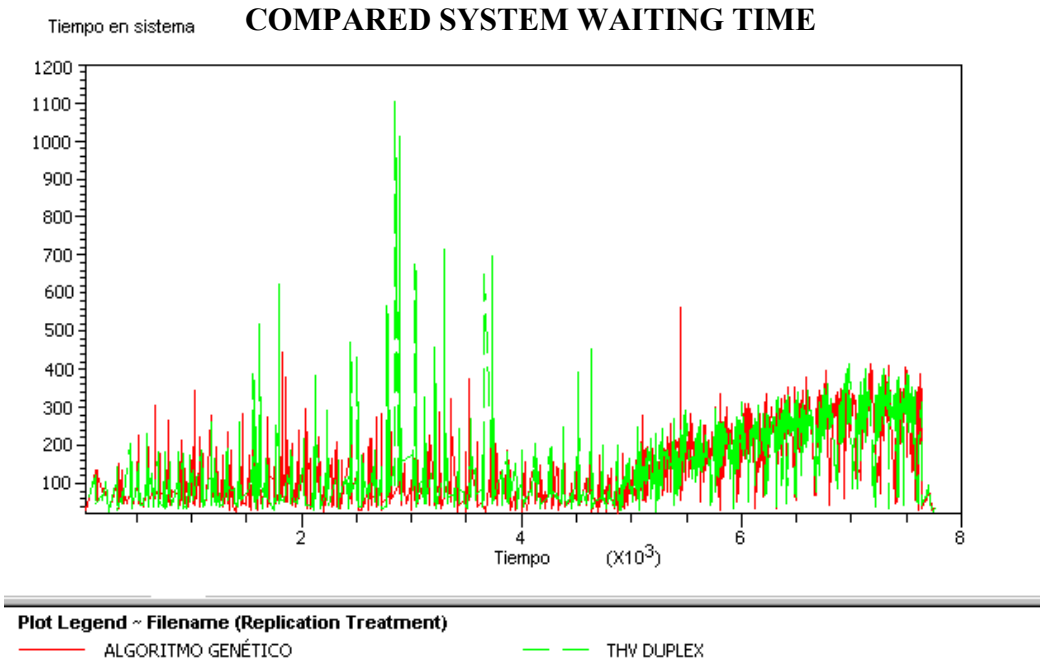


Figure 6. Comparison of system waiting times by floors

The most important peaks appear around the 14:50 (after 3000 seconds). At this moment a lot of passengers are accumulating due to the lunchpeak effect. Another peculiar effect is observed after 4800 seconds (15:20), the graphic have a monotonic increasing tendency with less significant peaks. This change is due to the arrival of passengers to the ground floor hall after lunching.

In every case both of the algorithms tend to benefit the passengers in the top of the building. The reason is that these passengers would be capable of using the stairs of the building with less probability than the passengers in lower floors (see waiting time for floor number 2 and 3). Note that the stairs effect has not been simulated.

Finally, next figure 7 depicts the comparative (in the same graphic and scale) the behaviour for the GAHCA and the THV, expressing and reinforcing all these comments. The figure shows the average system waiting times per floor. It can be viewed how the GAHCA beats the THV along the time horizon.



After 20 replications	THV Duplex	GAHCA
System Waiting Time	195.60 seconds	149.98 seconds

Figure 7. Comparative of overall average system waiting times

Hall queues

The table 2 summarises the average occupation degree of the up and down hall queues.

Table 2. Hall queues analysis

ARENA Simulation Results
PCA - License #8910593
Summary for Replication 20 of 20

Project: THV DUPLEX lunchpeak — Run execution date: 16/07/2002
Analyst: PCA — Model revision date: 16/07/2002
DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
NQ(Up-Queue Hall 1)	13.293	(Corr)	.00000	91.000	.00000
NQ(Up-Queue Hall 2)	.04073	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 3)	.03244	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 4)	.02693	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 5)	.02304	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 6)	.02547	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 7)	.03940	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 8)	.01189	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 9)	.01308	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 10)	.00622	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 11)	.00651	(Insuf)	.00000	1.0000	.00000
NQ(Down-Queue Hall 12)	.47267	(Corr)	.00000	3.0000	.00000
NQ(Down-Queue Hall 11)	.97950	(Corr)	.00000	5.0000	.00000
NQ(Down-Queue Hall 10)	1.0001	(Corr)	.00000	5.0000	.00000
NQ(Down-Queue Hall 9)	.95628	(Corr)	.00000	5.0000	.00000
NQ(Down-Queue Hall 8)	.97481	(Corr)	.00000	5.0000	.00000
NQ(Down-Queue Hall 7)	1.9891	(Corr)	.00000	14.000	.00000
NQ(Down-Queue Hall 6)	1.4776	(Corr)	.00000	15.000	.00000
NQ(Down-Queue Hall 5)	2.2106	(Corr)	.00000	35.000	.00000
NQ(Down-Queue Hall 4)	4.0493	1.3469	.00000	53.000	.00000
NQ(Down-Queue Hall 3)	5.1970	(Corr)	.00000	50.000	.00000
NQ(Down-Queue Hall 2)	7.2551	1.9828	.00000	77.000	.00000

Project: GENETIC ALGORITHM lunchpeak — Run execution date: 16/07/2002
Analyst: PCA — Model revision date: 16/07/2002
DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
NQ(Up-Queue Hall 1)	12.857	(Corr)	.00000	89.000	.00000
NQ(Up-Queue Hall 2)	.04143	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 3)	.03785	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 4)	.03237	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 5)	.02816	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 6)	.03030	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 7)	.04712	(Insuf)	.00000	2.0000	.00000
NQ(Up-Queue Hall 8)	.01179	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 9)	.01384	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 10)	.00566	(Insuf)	.00000	1.0000	.00000
NQ(Up-Queue Hall 11)	.00445	(Insuf)	.00000	1.0000	.00000
NQ(Down-Queue Hall 12)	.46608	(Corr)	.00000	4.0000	.00000
NQ(Down-Queue Hall 11)	1.0086	(Corr)	.00000	9.0000	.00000
NQ(Down-Queue Hall 10)	1.0901	(Corr)	.00000	12.000	.00000
NQ(Down-Queue Hall 9)	1.1416	(Corr)	.00000	12.000	.00000
NQ(Down-Queue Hall 8)	1.2005	(Corr)	.00000	11.000	.00000
NQ(Down-Queue Hall 7)	2.3990	(Corr)	.00000	23.000	.00000
NQ(Down-Queue Hall 6)	1.1878	(Corr)	.00000	12.000	.00000
NQ(Down-Queue Hall 5)	1.3672	(Corr)	.00000	17.000	.00000
NQ(Down-Queue Hall 4)	1.3728	(Corr)	.00000	12.000	.00000
NQ(Down-Queue Hall 3)	1.4351	(Corr)	.00000	17.000	.00000
NQ(Down-Queue Hall 2)	1.6728	(Corr)	.00000	19.000	.00000

It is remarkable how in case of THV the up queues and the down queues display different results. Note how the maximum number of persons waiting to take the elevator up in the ground floor reaches 91 (this is the maximum value after 20 replications). This

is due to the effect of the arrival after lunching. Note how even being available the two elevators in the ground floor, the system would not be able of carrying all the passengers (the maximum capacity of each cabin is 40).

Respect to the maximum values in the down queues, the size of the queues is bigger for the lower floors (maximums upper than 50 for the fourth, third and second floor). This situation is due to the THV performance. It is known as one of the main weakness of the THV algorithm its performance in downpeak traffic (when there exists a lot of people wishing to leave the building). The allocation procedure cause the cars to travel to calls too high in the building, neglecting calls lower in the building, and thus giving better service to the higher floors.

Although the GAHCA do not get significant improvements in the queues of the ground floor (due to the heavy lunchpeak traffic conditions), it reduces drastically the size of the down queues of the lower floors (the critical queues for THV). In these cases, the average values do not reach even 2 passengers waiting (excepting the Administration floor where more people works and a worst case of 23 passengers waiting appeared). All these are moderated values having in account the heavy traffic considered, the number of workers supposed, and the number and capacity of elevators assumed. All these parameters were brought to extreme values to perceive the algorithm performance.

Next figure 8 depicts the maximum size of the queues in the ground floor, as well as the down queue sizes for the THV and its comparison with the average and maximum values for GAHCA.

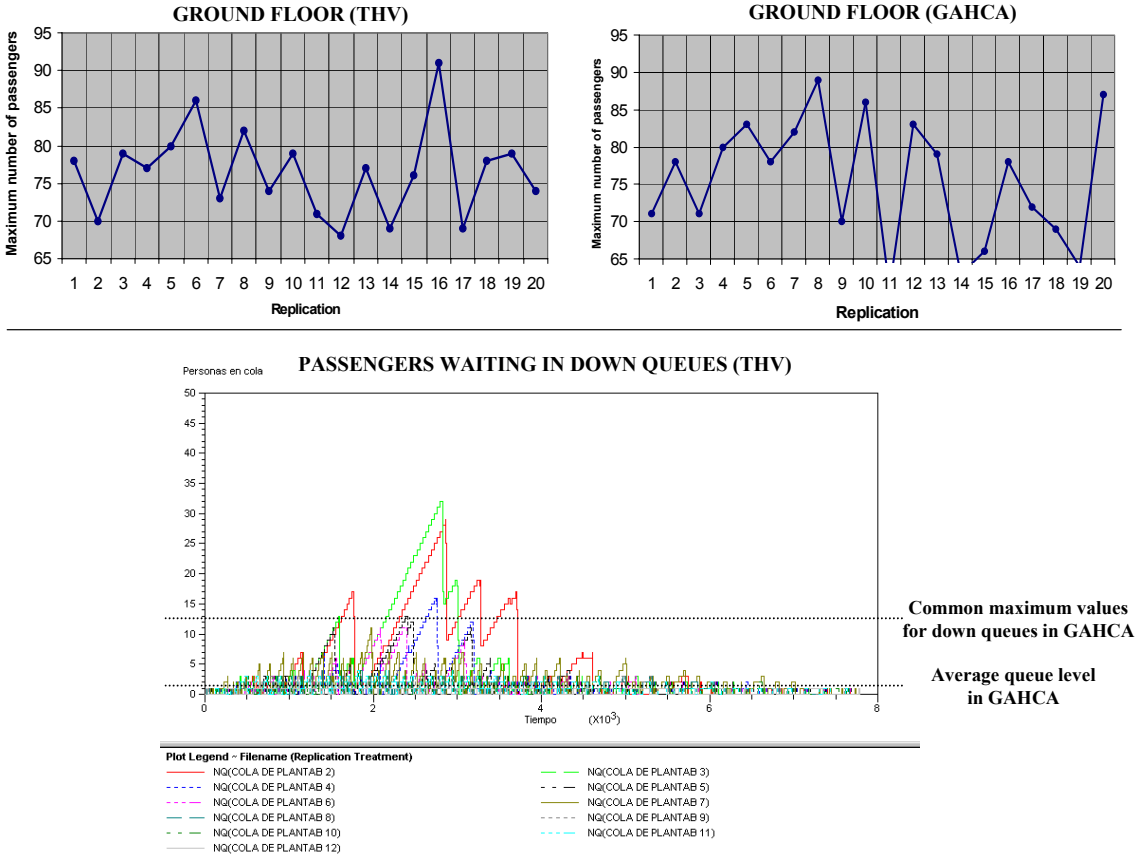


Figure 8. Queue analysis

State of occupation of the elevators

The state of occupation of the elevators can be used as a good index to infer the system energy consumption. The table 3 summarises the results of lift occupation.

Table 3. Elevator occupation analysis

ARENA Simulation Results
PCA - License #8910593
Summary for Replication 20 of 20

Project: THV DUPLEX lunchpeak — Run execution date: 16/07/2002
Analyst: PCA — Model revision date: 16/07/2002
Replication ended at time: 7800.0

FREQUENCIES

Identifier	Category	--Occurrences-- Number	AvgTime	Standard Percent	Restricted Percent
Elevate_1	Stop	231	13.479	2.00	2.00
	Up	1397	44.446	39.80	39.80
	Down	1374	66.080	58.20	58.20
Elevate_2	Stop	433	14.236	3.95	3.95
	Up	1559	42.014	41.99	41.99
	Down	1538	54.834	54.06	54.06

Project: GENETIC ALGORITHM lunchpeak — Run execution date: 16/07/2002
Analyst: PCA — Model revision date: 16/07/2002
Replication ended at time: 7800.0

FREQUENCIES

Identifier	Category	--Occurrences-- Number	AvgTime	Standard Percent	Restricted Percent
Elevate_1	Stop	252	17.155	2.77	2.77
	Up	1603	39.519	40.61	40.61
	Down	1583	55.797	56.62	56.62
Elevate_2	Stop	262	16.862	2.83	2.83
	Up	1608	39.037	40.24	40.24
	Down	1585	56.031	56.93	56.93

In THV the inactivity time percentage is 2.00% for elevator 1 and 3.95% for elevator 2. In GAHCA the inactivity time percentage is 2,77% for elevator 1 and 2.83% for elevator 2. Therefore, the results so obtained are very similar: 664 times an elevator is stopped in THV while 514 times an elevator is stopped in GAHCA. So, having in account that the major energetic consumption is produced in the start of the elevator both systems will have similar energetic consumption, perhaps a little inferior in the THV system. We have to note that extreme energetic savings could produce the effect of bad performance from the users perspective, i.e., the increase of the waiting times and the queues.

Also, in both cases can be observed other typical phenomenon: the time taken for the downward displacements is upper than the time taken for the upward displacements. This is due to the lunchpeak characteristics where the downpeak period takes a major part of the time than the uppeak period. The figure 9 depicts all these comments (we have use the state 0 for elevator stopped; the state 1 for elevator going up; and the state 2 for elevator going down).

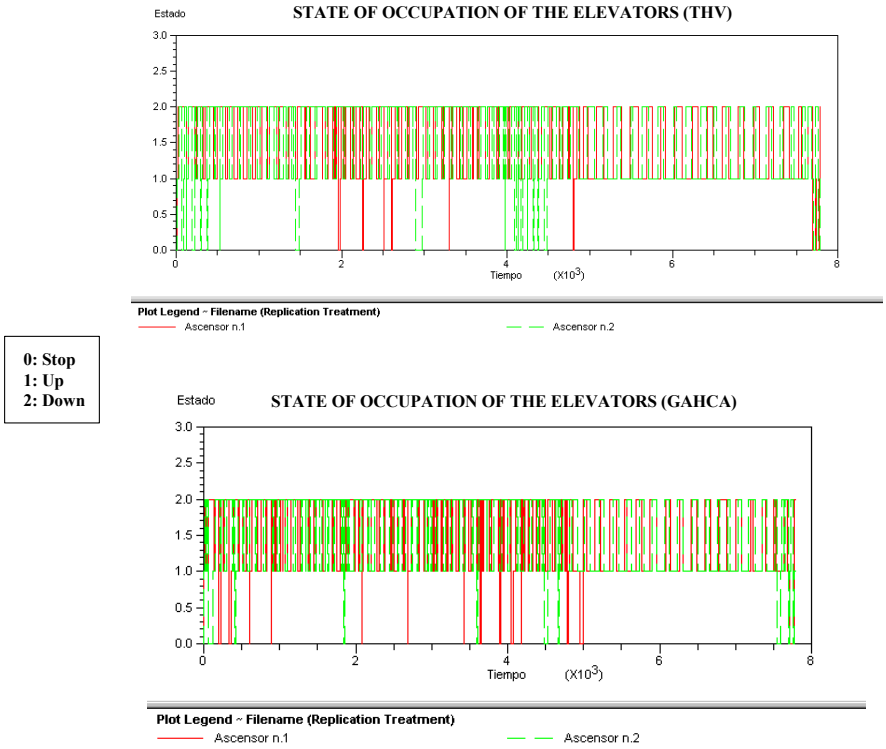


Figure 9. Analysis of elevator occupation

5. CONCLUSIONS

We have proposed a genetic algorithm (GAHCA) to control the elevator group in a professional building. The results allow us to affirm that our genetic algorithm reaches a better performance attending to the system waiting times and queue sizes than traditional controllers in industry applications as THV algorithm. The reduction of waiting times has been almost the 25% at the same time as getting a significant reduction of the hall down queues. In this situation the passengers are supposed to experiment a system time reduction from 3min15sec to 2min30sec. The analysis has been done under heavy lunchpeak traffic conditions.

The results obtained in the paper allow us to affirm that genetic algorithms, in general, and our GAHCA in particular, are valuable tools with a great potential in the control of elevator systems. However, the implementation of such type of algorithms in real controllers has to be done carefully in order to maintain bounded the response time of the algorithm. Genetic algorithms are iterative and therefore they can take very much time of execution when a long population and a great number of iterations are used. The election of these parameters has to be selected attending not so much to the algorithm accuracy but to the available time of trip of the elevator between different events, that is the time necessary to allocate a hall call (it can be the trip time between a number of floors determined, e.g. no more than two). In real cases an alternative can be stopping the algorithm previously to reach the next event, which would occur after a known time interval. Of course all these decisions are very dependant on the computation speed of the electronic microchips installed by the company.

Acknowledgements
This paper has been carried out in collaboration with MAC PUAR, S.A. (MP). MP has been supporting our research on elevator systems since 2000. Additionally, the authors acknowledge the financial support given by the Ministerio de Ciencia y Tecnologia, in its Industrial Production and Design Programme (project ref. DPI2002-01264), Spain.

Appendix 1. THV pseudocode

```
N = number of floors in the building
Read the system current state
d = Distance (call, car) = |call floor – elevator floor|
IF elevator is homing to the call floor with the same trip direction of the hall call
        Fitness Function = N+1-d
ELSE IF elevator is homing to the call floor with trip direction different from the hall call
        Fitness Function = N-d
ELSE IF elevator has just leaved the floor of the hall call
        Fitness Function = 1
ELSE (the elevator is stopped)
        Fitness Function = N-d
Car Allocation = Best Fitness Function
ARENA assignation
```

Appendix 2. GAHCA pseudocode

```
Read the system current state
Generate the population
Calculate the fitness population
Ordinate the fitness population
IF Population size = 20 THEN
        i = 0
        WHILE i < 50
                p = Rnd
                IF p < 0.85 THEN        'Crossover operator'
                        Incest=1
                        WHILE incest=1
                                Randomly selection of parents
                                Parents incest prevention
                                IF No incest THEN
                                        Crossover -> offspring
                                END IF
                        END WHILE
                ELSE                    'Mutation operator'
                        Randomly selection of parent
                        Mutation -> offspring
                END ELSE
                Individuals duplicity control
                IF No duplicity THEN
                        Evaluation of the individual fitness
                        Selection of individual for replacement
                        New individual -> offspring
                        Modification of the population fitness table
                        i = i + 1
                END IF
        END WHILE
END IF
Solution = Best fitness individual
ARENA assignation
```

References

[1] MacDonald, C. Robert and E. Abrego. Coincident call optimization in a elevator dispatching system, Westinghouse Electric Corp. U.S. Patent No. 4 782 921, 1988.

[2] Thangavelu and Kandasamy. Queue based elevator dispatching system using peak period traffic prediction, Otis Elevator Company. U.S. Patent No. 4 838 384, 1989.

[3] Thangavelu and Kandasamy. "Artificial intelligence", based learning system predicting "peak-period" times for elevator dispatching, Otis Elevator Company, U.S. Patent No. 5 241 142, 1993.

[4] Kameli, N. and Nader. Floor population detection for an elevator system, Otis Elevator Company. U.S. Patent No. 5 511 635, 1996.

[5] Kameli, N., Nader, Collins and M. James. Elevator downpeak sectoring, Otis Elevator Company. U.S. Patent No. 5 480 006, 1996.

[6] Kim, C. and O. Jeong. Group management control method for elevator system employing traffic flow estimation by fuzzy logic using variable value preferences and decisional priorities, LG Industrial Systems Co., Ltd. U.S. Patent No. 5 679 932, 1997.

[7] Bahjat, S. Zuhair and J. Bittar. Automated selection of high traffic intensity algorithms for up-peak period, Otis Elevator Company. U.S. Patent No. 5 168 133, 1992.

[8] Siikonen, M-L.. Elevator group control with artificial intelligence, Helsinki University of Technology, Systems Analysis Laboratory, Research Reports A67 (1997).

[9] Hauptmeier, D., S.O. Krumke and J. Rambau. The online dial-a-ride problem under reasonable load, Preprint SC 99-08, Konrad-Zuse-Zentrum für Informationstechnik Berlin (1999).

[10] Larrañeta, J. and Cortes, P. Optimización Dinámica en Sistemas de Tráfico Vertical. Escuela Superior de Ingenieros. Ingeniería de Organización. Seville University. Technical Report IO-01MP

[11] Barney, G.C. and S.M. dos Santos. Elevator Traffic Analysis, Design and Control (Peter Peregrinus Ltd, 2nd edition, London, 1985).

[12] Siikonen, M-L.. Planning and control models for elevators in high-rise buildings, Ph.D. Thesis, Helsinki University of Technology, 1997.

[13] Sasaki, K., S. Markon and M. Makagawa. Elevator Group Supervisory Control System Using Neural Networks, Elevator World (1996).

[14] Crites, R.H. and A.G. Barto. Improving elevator Performance Using Reinforcement Learning, Advances in Neural Information Processing Systems 8. MIT Press (1996).

[15] Kim, C., K.A. Seong and H. Lee-kwang. Design and implementation of a fuzzy elevator group control system, in: Proceedings of the IEEE Transactions on systems, man and Cybernetics (1998), vol. 28, No. 3, 277-287.

[16] Gudwin, R., F. Gomide and M.A. Netto. A Fuzzy Elevator Group Controller with Linear Context Adaptation, in: Proceedings of FUZZ-IEEE98, WCCI'98 - IEEE World Congress on Computational Intelligence, Anchorage, Alaska, USA (1998) 481-486.

[17] Fujino, A., T. Tobita, K. Segawa, K. Yoneda and A. Togawa. An Elevator Group Control System with Floor-Attribute Control Meted and System Optimization using Genetic Algorithms, IEEE Transactions on Industrial Electronics Vol. 44 no.

4 (1997) 546-552.
[18] Tobita, T. A. Fujino, K. Segawa, K. Yoneda and Y. Ichikawa. A Parameter Tuning Meted for an Elevator Group Control System using a Genetic Algorithm, Electrical Engineering in Japan, Vol. 124, No. 1 (1998) 55-64.
[19] Gudwin, R. and F. Gomide. Genetic Algorithms and Discrete Event Systems: An Application, in: Proceedings of The First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence (1994), vol II, 742-745.
[20] Alander, J.T., J. Ylinen and T. Tyni. Elevator Group Control Using Distributed Genetic Algorithm, in: Proceedings of the International Conference. Springer-Verlag, Vienna, Austria (1995), 400-403.
[21] Siikonen, M-L.. Elevator traffic simulation, Simulation Vol. 61 No. 4 (1993) 257-267.
[22] Goldberg, D.E. Sizing populations for serial and parallel genetic algorithms, in Proceedings of the Third International Conference in Genetic Algorithms, (J.D. Schaffer, ed.) (1989), 70-79.