OPTIMAL ELEVATOR GROUP CONTROL USING GENETIC ALGORITHMS

W. GHARIEB

Computer and Systems Engineering Dept., Faculty of Engineering
Ain Shams University, E-mail: wahied@hotmail.com

Abstract: This paper presents an optimal control strategy for elevator group in large buildings. An elevator group controller allocates different elevators to up and down hall calls that coming from different floors. The control strategy is a precondition to minimize mainly the average waiting time of hall calls. In this paper, two genetic algorithms are developed to treat efficiently the elevator group problem. ***The first one*** handles the set of up and down hall calls while their destination floors are unknown at this moment. ***The second one*** handles the hall calls in assuming that their destination floors are known. The simulation results of the proposed algorithms gave a superior performance to that obtained with a traditional control in the industry (duplex algorithm). *Copyright © 2005 IFAC*

Keywords: Optimal control, Optimization, Discrete-event systems, Genetic algorithms, Methodology.

1. INTRODUCTION

Elevators play an important role in our daily life. The elevator group form a class of discrete event systems whose complexity makes them difficult to model, analyze, and optimally control. In multiple car elevator systems that serve large (office) buildings, a major challenge is that of developing a dispatching control policy, i.e. a scheme for systematically deciding when and where each car should move, stop, or switch direction based on the current state and available past history. A system study of the elevator dispatching control problem begins by decomposing passenger traffic into four different situations: Uppeak traffic, Lunchtime traffic, Downpeak traffic, and Interfloor traffic (Pepyne and Cassandras, 1997). Each situation has different conditions can be analyzed separately to form the global controller. This problem has been treated using different approaches.

The first approach is mainly based on the analytical models derived from queuing theory. In (Pepyne and Cassandras, 1997), the problem of Uppeak traffic is only considered. The dynamic programming is used to find the optimal policy that minimizes the average passenger waiting time. The analysis is based on a Markov decision problem formulation with a batch service queuing model.

In (Nikovski and Brand, 2003a), Dynamics of the system are represented by a discrete-state Markov chain embedded in the continuous phase space diagram of a moving elevator car. The chain is evaluated efficiently using dynamic programming to compute measures of future system performance such as expected waiting time, properly averaged over all possible future scenarios. In (Nikovski and Brand, 2003b), the effects of future passengers arriving at the lobby and entering elevator cars can dominate all waiting times. Authors have developed a probabilistic model of how these arrivals affect the behavior of elevator cars at the lobby, and demonstrated how this model can be used to very significantly reduce the average waiting time of all passengers.

The second approach is based on the use of artificial intelligence techniques such as: fuzzy logic (FL), neural networks (NN) and genetic algorithms (GA). In (Beielstein, *et al*., 2003a), the evolution strategy is used to optimize a neural controller. Neural controller connection weights are modified, so that different weight settings and their influence on the elevator group controller performance can be tested. In (Beielstein, *et al*., 2003b), a validation methodology for a simplified elevator group system is proposed. The S-ring is constructed as a simplified model of the system using a neural network (NN) to control the elevators. In (Siikonen, 1997a), a control that optimizes passenger service in an elevator group is described. Fuzzy logic (FL) and artificial intelligence are applied in the control when allocating landing calls to the elevators. Fuzzy logic is used to recognize the traffic pattern and the traffic peaks from statistical forecasts. In (Siikonen, 1997b), an elevator control system, the Traffic Master System 9000 was developed. The TMS9000 group control system optimizes passenger waiting times instead of hall call times, and the control actions can be started in advance, which is not the case with conventional controls. In (Cortes, *et al*., 2003; Cortes, *et al*., 2002), a genetic algorithm (GA) is proposed to control elevator groups of professional buildings. The genetic algorithm is compared with traditional controller algorithms in industry applications. The genetic algorithm reaches a better performance attending to the system waiting times than traditional algorithm. However, this algorithm did not take into account the preload condition of each car (car calls) to improve the system response.

The paper is organized as follows: section 1 presents the different approaches to design an optimal supervisory controller for elevator group. Section 2 describes the elevator group problem. Section 3 presents the basic concept of genetic algorithms that used in optimization. In section 4, the proposed optimal control policy is presented. Section 5 involves simulation results and the investigation of system performance. Some concluding remarks given in section 6; end the paper.

2. ELEVATOR GROUP PROBLEM

In this section, the elevator supervisory group control problem is considered. An elevator group controller assigns elevators to service calls. There are two types of service calls: hall calls from different floors in the building and car calls from the different cars as shown in figure 1. The control system consists of two levels: Local control level for each car that can handle car calls and a supervisory level to handle the hall calls and assign different cars in order to optimize certain objective function. An optimal supervisory control strategy has a multi-objective criterion to be optimized with main goals such as:

- Minimum average waiting time
- Minimum power consumption
- Maximum car capacity, … etc.

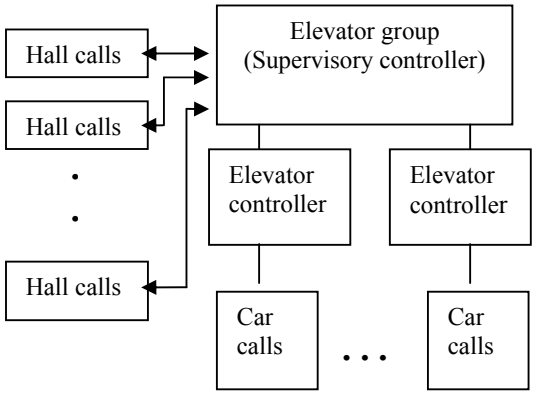

Fig. 1. Elevator group system

When dealing with elevator group systems is a usual practice to consider the following assumptions (Cortes, *et.al*., 2003):

- Each hall call is attended by only one car.
- The maximum number of passengers being transported in a car is bounded by its capacity.
- The lift car stops at a floor only if exists a hall call or car call in that floor.
- The car calls are sequentially served in accordance with the lift trip direction.
- A lift carrying passengers cannot change the trip direction until all car calls are served.

The traditional type of controllers implemented in the industry follows simple dispatch rules that make use of an IF-THEN logical commands set. Among these dispatch rules, the traditional duplex algorithm is one of the most habitual algorithms (Full Collective Control) (Otis, 1991). With no calls in the system one car rests at the main floor, the others are distributed evenly throughout the other floors. When a landing call is received, the microprocessor calculates which car is nearest to the call, traveling in the required direction. Each car responds to its car calls in logical sequence, depending upon direction of travel, and takes landing calls as assigned by microprocessor (controller). The microprocessor continuously monitors the system and re-assigns calls when necessary. This algorithm will be compared via simulation with the proposed genetic algorithms in section 4.

3. GENETIC ALGORITHMS

The basic concepts of genetic algorithms (GAs) were developed by Holland (Holland, 1975). GAs are adaptive parallel search techniques, based on the principles of natural selection and evolutionary processes, which, in control systems engineering, can be used as an optimization tool or as the basis of more general adaptive systems (Fleming and Fonseca, 1993). Genetic algorithms operate on a population of randomly drawn individuals from which improvement is sought. Individuals are coded as strings, the chromosomes constructed over some alphabet, e.g., the binary alphabet {0, 1}, so that

chromosome values are uniquely mapped onto the decision variable domain. The chromosomes in the organism correspond to the parameters to be determined in system design. All of the search process then takes place at the coding level to find out the optimum parameters. A simple genetic algorithm is described by Goldberg (Goldberg, 1989). Generally, genetic algorithms consist of three fundamental operators: reproduction, crossover, and mutation**.**

A) ***Reproduction*** is a process in which individual strings are copied according to their fitness value. The objective function $f(x_i)$ is evaluated for each chromosome x_i in the current population. This function can be any nonlinear, non-differentiable, discontinuous, positive function. The individual fitness, $F(x_i)$ is computed as the individual performance relative to that of the whole population (normalization), i.e.,

$$F(x_i) = \frac{f(x_i)}{\sum_{i=0}^{N} f(x_i)} \quad (1)$$

Where N is the population size; x_i represents the value of chromosome i. A probabilistic component is often incorporated into the selection procedures and one mechanism is known as roulette wheel selection. Each $F(x_i)$ is used as the width of a slot of a biased roulette wheel. Selection is performed by spinning the roulette N times to obtain N individuals to be integrated in the next generation.

B) ***Crossover*** is the process to generate new individuals. Individuals are paired at random with a high probability (0.6 – 0.95) that crossover will take place. The crossover point is selected at random and the rightmost segments of each individual are exchanged to produce two offspring as shown in figure 2. Sometimes two crossover points are selected at random and the middle segments are exchanged.

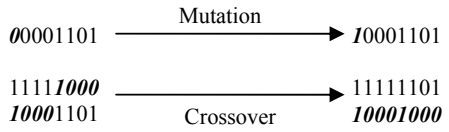

Fig. 2. Crossover and mutation

C) ***Mutation*** is simply flipping an individual bit with low probability (0.001 - 0.01). This background operator is used to ensure that the probability of searching a particular subspace of the problem space is never zero, thereby tending to inhibit the possibility of ending the search at a local, rather than a global optimum. The evolution from one generation to the next one involves mainly three steps (Jang, *et al*., 1997; John and Langari, 1999): ***evaluation***, ***selection***, and ***reproduction***. First, the current population is evaluated using the fitness evaluation function and then ranked based on their fitness values. Second, GA stochastically selects parents from the current population with a bias that better chromosomes are more likely to be selected. This is accomplished using a selection probability that is determined by the fitness value or the ranking of a chromosome. Third, the GA produces children from selected parents using two genetic operators: crossover and mutation. This cycle of evaluation, selection, and reproduction terminates when an acceptable solution is found, when a convergence criterion is met, or when a predetermined limit on the number of iterations is reached.

4. OPTIMAL SUPERVISORY CONTROL

In this section, GA will be used to develop optimal supervisory control strategies for hall calls in elevator group systems. Car calls from inside the lift can't be predicated a priori and it interrupts asynchronously the system behavior. The globally optimal control policy is not easy to be obtained due to randomly distributed service calls and dynamic changes in traffic loads. Consequently, efficient robust methods from the domain of black-box optimization are required where evolutionary computation is one of the most promising approaches (Beielstein, *et al*., 2003a).

4.1 GA1 (unknown hall calls destination)

The proposed algorithm allocates hall calls to perform the elevator group controller in N floors building. The received hall calls are stored into a list according to the moment of the request. The algorithm will handle a fixed number of hall calls at a time with maximum length of 10 calls. So the chromosome size is equal to hall calls times the binary code of cars. For examples, consider a system having 4 cars, then the individual consists of 20 binary bits to allocate 10 hall calls at a time, where the car code consists of 2 bits (00 car1, 01 car2, 10 car3, and 11 car4). Therefore, each individual of 20 bits presents a complete solution in the current generation. The fitness of this solution can be evaluated as follows:

$$\text{Fitness} = \frac{1}{T_{av}} \quad (2)$$

The average waiting time, T_{av} is computed as:

$$T_{av} = \frac{1}{m}\sum_{i=1}^{m} T_i \quad (3)$$

Where T_i estimated waiting time of hall call i
m total number of hall calls to be served

The updated service list of a car consists of the preload condition (car calls) and the adding stops from the current solution. Define the following

parameters to be able to estimate the waiting time of hall call i as following:

- HC hall call floor
- CF car current floor
- NF number of building floors
- TF inter-floor trip time

The waiting time of a hall call depends on the direction of hall call up or down, the car trip direction, and also the relative position of hall call with respect to the current car floor. In figure 3, **Case (I):** If HC is up and less than CF, the car trip consists of three segments: from CF to NF, from NF to 1st floor, and from 1st floor to HC.

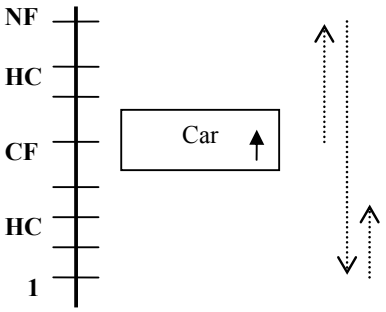

Fig.3. Elevator is stopped or going up

Case (II): If HC is down, the car trip consists of two segments: from CF to NF and from NF to HC. **Case (III):** If HC is up and greater than or equal CF, the car trip consists of only one segment: from CF to HC. Therefore, the waiting time T_i of the hall call is given by:

$$T_i = \begin{cases} [HC - CF]\,TF & ; \uparrow HC \geq CF \\ [2\,NF - CF + HC - 2]\,TF & ; \uparrow HC < CF \\ [2\,NF - CF - HC]\,TF & ; \downarrow HC \end{cases} \quad (4)$$

In figure 4, **Case (I):** If HC is down and greater than CF, The car trip of a going down car consists of three segments: from CF to 1st floor, from 1st floor to NF, and from NF to HC. **Case (II):** If HC is up, the car trip consists of two segments: from CF to 1st floor and from 1st floor to HC. **Case (III):** If HC is down and less than or equal CF, the car trip consists of only one segment: from CF to HC.
Therefore, the waiting time T_i of the hall call is given by:

$$T_i = \begin{cases} [CF - HC]\,TF & ; \downarrow HC \leq CF \\ [2\,NF + CF - HC - 2]\,TF & ; \downarrow HC > CF \\ [CF + HC - 2]\,TF & ; \uparrow HC \end{cases} \quad (5)$$

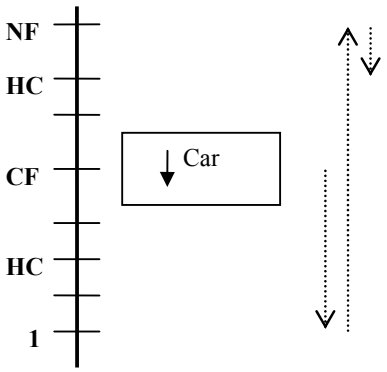

Fig.4. Elevator is going down

4.2 GA2 (known hall calls destination)

This algorithm is different in the following assumption that makes the system behavior deterministic. In contrast to traditional elevators, where passengers only press a button to request up or down service and choose the exact destination from inside the elevator car, a destination call system lets the passenger choose the desired destination at a terminal before entering the elevator car. This provides more exact information to the group controller, and allows higher efficiency by grouping of passengers into elevators according to their destinations. Based on the above assumption that all calls are destination calls from different floors (deterministic system), define the following:

- DU maximum up destination floor (hall call)
- DD minimum down destination floor (hall call)
- CU maximum up destination floor (car call)
- CD minimum down destination floor (car call)
- MU maximum of (DU, CU)
- MD minimum of (DD,CD)

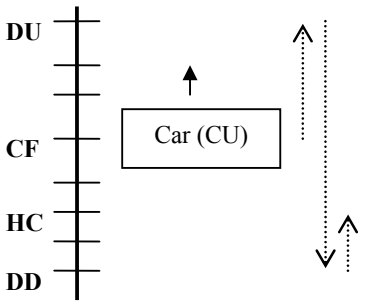

Fig.5. Elevator is stopped or going up (known destination)

In figure 5, **Case (I):** If HC is up and less than CF, the car trip consists of three segments: from CF to MU, from MU to DD, and from DD to HC. **Case(II):** If HC is down, the car trip consists of two segments: from CF to MU and from MU to HC. **Case (III):** If HC is up and greater than or equal CF, the car trip consists of only one segment: from CF to HC. Therefore, the estimated waiting time T_i of the hall call is given by:

$$T_i = \begin{cases} [HC - CF]\,TF & ; \uparrow HC \geq CF \\ [2\,MU - CF + HC - 2\,DD]\,TF & ; \uparrow HC < CF \\ [MU - CF + |HC - MU|]\,TF & ; \downarrow HC \end{cases} \quad (6)$$

In the above equation, HC (down) could be less than MU; so the absolute function is used to avoid a negative result. In figure 6, **Case (I):** If HC is down and greater than CF, the car trip consists of three segments: from CF to MD, from MD to DU, and from DU to HC.

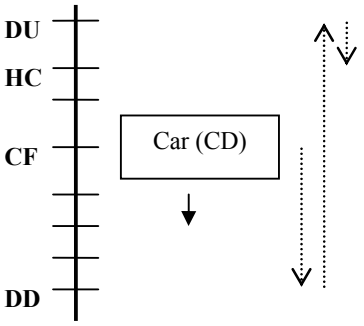

Fig.6. Elevator is going down (known destination)

Case (II): If HC is up, the car trip consists of two segments: from CF to MD and from MD to HC. **Case(III):** If HC is down and less than or equal CF, the car trip consists of only one segment: from CF to HC. The estimated waiting time T_i of the hall call is given by:

$$T_i = \begin{cases} [CF - HC]\,TF & ; \downarrow HC \leq CF \\ [2\,DU + CF - HC - 2\,MD]\,TF & ; \downarrow HC > CF \\ [CF + HC - 2MD]\,TF & ; \uparrow HC \end{cases} \quad (7)$$

It is expected that this algorithm will give superior performance where adequate information about the passenger destination are involved. The above algorithm is summarized as:

1. Read *the system current state*
2. Generate *randomly the initial population of 6 individuals*
3. Compute t*he fitness of each solution (Equation 2)*
4. Rank *the population according to the fitness values*
5. Replace *the solution that has lower fitness value with the one that has its maximum*
6. Apply *crossover process between parents with the given probability*
7. Apply *mutation process to flip randomly one binary bit in the individual*
8. Generate *the new population*
9. Go to *step 3, if the convergence condition is not met*
10. Select *the best solution (Best fitness function)*
11. Select *next 10 hall calls from the hall calls list*
12. If *no calls distribute the cars throughout the building floors****, else*** *Go to step 1*

5. SIMULATION RESULTS

The simulation work is carried out on a PC using MATLAB. Information is treated as list processing in the programming environment using variable length array. The following parameters are chosen for simulation:

- 20 floor building is served by a group of 4 cars
- Inter-floor distance = 4 m
- Door opening time = 2 sec
- Door closing time = 3 sec
- Passenger transfer rate = 2 sec
- Car speed 2 m/sec

Hall calls are:
H_{15} (9), H_{13} (6), H_{12} (20), H_{11} (2), H_9 (16), and H_7 (1)
That means; hall call from floor 15 is a down call to the 9th floor. The situation is similar for other hall calls.

The pre-load condition for each car is given by:
Car1 is in up direction to floor 7, CF=5
Car2 is in down direction to floor 8, CF=17
Car3 is in up direction to floors (18, 20), CF=3
Car4 is in down direction to floors (6, 1), CF=19

The proposed algorithms in section (4) are applied via simulation for the above initial conditions and compared with the duplex algorithm. The probability of crossover is chosen 70% and mutation rate is chosen 2%. The generation consists of 6 individuals. The chromosome length is equal two times the number of hall calls, where the car code consists of 2 binary bits. The following table presents the assigned hall calls to each car for each algorithm.

Table 1 Assigned hall calls

Algorithm	Car1	Car2	Car3	Car4
Duplex	H_9, H_{12}	H_{15},H_{13}, H_{11},H_7	Non	Non
GA1	H_9, H_{12}	H_{15},H_{13}, H_{11},H_7	Non	Non
GA2	H_7	H_{15}, H_{13}, H_{11}	H_{12},H_9	Non

From the above table, it is clear that the first genetic algorithm gave the same results as the duplex. That means, the duplex algorithm is optimal in the sense of search optimization to minimize the average waiting time as the hall calls destinations are unknown. The second genetic algorithm gave another result where the hall calls destinations are known to improve the system performance.

Table 2 Cars trip time

Algorithm	Average trip time (sec)	Car1 trip time (sec)	Car2 trip time (sec)	Car3 trip time (sec)	Car4 trip time (sec)
Duplex	64.5	65	95	48	50
GA1	64.5	65	95	48	50
GA2	59.25	32	86	69	50

The GA2 gave a better performance where the average trip time is reduced by 8.12% and consequently the consumed energy is reduced.

Table 3 Car stops

Algorithm	Total (stops)	Car1 stops	Car2 stops	Car3 stops	Car4 stops
Duplex	18	5	9	2	2
GA1	18	5	9	2	2
GA2	17	2	8	5	2

The number of stops is reduced using GA2 and consequently the consumed energy is reduced.

Table 4 Waiting time analysis

Hall calls	*Waiting time (sec)* ***Duplex***	*Waiting Time (sec)* ***GA1***	*Waiting Time (sec)* ***GA2***
H_7	55	55	11
H_9	22	22	19
H_{11}	33	33	33
H_{12}	35	35	36
H_{13}	22	22	22
H_{15}	11	11	11
Average time (sec.)	29.67	29.67	22

Table 4 compares the waiting time between the different algorithms. The obtained results affirmed the potential of the proposed algorithm GA2 where the average waiting time is reduced by 27%.

6. CONCLUSION

Two genetic algorithms are proposed to search the optimal control strategy for elevator group. The analysis has been done under normal traffic conditions. The simulation results are compared to the duplex algorithm. The first algorithm GA1 has proved that the duplex algorithm is optimal and gave the same results. Therefore, duplex algorithm is practical and so simple if the hall calls destinations are unknown a priori. The second algorithm GA2 gave a superior performance where the average waiting time is reduced by 27% and the average trip time is reduced by 8.12%. Therefore, GA2 algorithm is more effective and economic when the hall calls destinations are known. During simulation, it is observed that the developed GA takes more time as iterative algorithm rather than duplex algorithm. However, the real time decision making algorithms are very dependent on hardware environment and it requires a more powerful microprocessor chips in architecture and real time performance.

REFERENCES

Beielstein T., C-P. Ewald, and S. Markon (2003a). *Optimal Elevator Group Control by Evolution Strategies*, Dortmund University, technical report, ECCO 2003, LNCS 2724, pp. 1963–1974, Germany.

Beielstein T., S. Markon, and M. Preuss (2003b). *Algorithm Based Validation of a simplified Elevator Group Controller Model*, 5th Metaheuristics Int. Conf., Kyoto-Japan.

Cortes P., J. Larraneta, and L. Onieva (2003). *A Genetic Algorithm for Controlling Elevator Group Systems*, 7th Int. Work-Conference on Artificial and Natural Neural Networks IWANN, Vol.2, pp.313-320, SPAIN.

Cortes P., J. Larraneta, and L. Onieva (2002). *Genetic Algorithm for Controllers in Elevator Groups: Analysis and Simulation During Lunchpeak Traffic*, Seville University, Research Report, DPI2002-01264, SPAIN.

Fleming P. J. and C. M. Fonseca (1993). *Genetic Algorithms in Control Systems Engineering*, 12th World Congress IFAC, Vol.2/20, pp.383-390, Sydney-Australia.

Goldberg D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company.

Holland J. H. (1975). *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI, The University of Michigan Press.

Jang J-S., C-T Sun, and E. Mizutani, (1997). *Neuro-Fuzzy and Soft Computing*, Ch(7), pp.177-178, Prentice-Hall Inc.

John Y. and R. Langari (1999). *Fuzzy Logic: Intelligence, Control, and Information*, Prentice-Hall Inc., Ch(17), pp.469-470.

Nikovski D. and M. Brand (2003a). *Decision-Theoretic Group Elevator Scheduling*, 13th Int. Conf. on Automated Planning and Scheduling ICAPS'03, June 9-13 Trento, Italy.

Nikovski D. and M. Brand (2003b). *Marginalizing Out Future Passengers in Group Elevator Control*, 9th Conf. on Uncertainty in Artificial Intelligence, UAI-2003, Acapulco-Mexico, August 8-10.

Otis (1991). *Passenger Lift Planning Guide*, OTIS Elevator Company, pp.15-16.

Pepyne D. L. and C. G. Cassandras (1997). *Optimal Dispatching Control for Elevator Systems During Uppeak Traffic*, IEEE Trans. On Control Systems Technology, Vol. 5, No. 6, pp.629-643.

Siikonen M-L. (1997a). *Planning and Control Models for Elevators in High-Rise Buildings*, Helsinki University of Technology, Systems Analysis Laboratory, Research Report#A67.

Siikonen M-L. (1997b). *Elevator Group Control with AI*, Helsinki University of Technology, Systems Analysis Laboratory, Research Report#A68.