A particle swarm optimization algorithm for optimal car-call allocation in elevator group control systems

Berna Bolat
Yildiz Technical University, Faculty of Mechanical Engineering,
Mechanical Engineering Department, Yildiz, TR-34349, Istanbul, Turkey.
Email: balpan@yildiz.edu.tr

Oğuz Altun
Yildiz Technical University, Faculty of Computer Engineering,
Computer Engineering Department, Yildiz, TR-34349, Istanbul, Turkey.
Email: oguz@ce.yildiz.edu.tr

Pablo Cortés
University of Seville, Escuela Técnica Superior de Ingeniería, Ingeniería Organización
Camino de los Descubrimientos s/n, Sevilla 41092, Spain.
Email: pca@us.es

Abstract.- High-rise buildings require the installation of complex elevator group control systems (EGCS). In vertical transportation, when a passenger makes a hall call by pressing a landing call button installed at the floor and located near the cars of the elevator group, the EGCS must allocate one of the cars of the group to the hall call. We develop a Particle Swarm Optimization (PSO) algorithm to deal with this car-call allocation problem. The PSO algorithm is compared to other soft computing techniques such as genetic algorithm and tabu search approaches that have been proved as efficient algorithms for this problem. The proposed PSO algorithm was tested in high-rise buildings from 10 to 24 floors, and several car configurations from 2 to 6 cars. Results from trials show that the proposed PSO algorithm results in better average journey times and computational times compared to genetic and tabu search approaches.

Keywords: Elevator; lift; particle swarm optimization; elevator group control system; vertical transportation

1. Introduction

High-rise buildings require the installation of large elevator groups. Therefore, the vertical transportation industry grows together with the more and more increase of such high-rise buildings. Such situations require the management of multiple elevators in a coordinated way in order to efficiently transport passengers throughout the building. This management is done by the Elevator Group Control System (EGCS).

The EGCS performance depends on the traffic pattern in the building. It represents the mobility of a building population in its necessities of vertical transportation. Each building addresses a specific shape of its own traffic pattern. Typically, in a professional building the traffic pattern will present a larger than average number of up landing calls at the start of the day. These are due to the building's workers arriving to start work. This phase is called uppeak traffic. On the contrary, late in day there is the opposite phenomenon, and a larger than average number of

down landing calls takes place. It corresponds to the building's population wanting to go home after a working day. This traffic pattern is called downpeak. In the middle of the day there are two joint phenomena, because the appearance of up and down peaks. It depicts a situation of people wanting to leave the building for lunch and people coming back after lunch. This period is called lunchpeak traffic. Finally the rest of the day does not show any special tendency from any specific floor or from any specific stream. Generally, less traffic is registered too. It is called interfloor traffic.

It has been proven (Barney *et al.* 1985; or Cibse Guide D (2000) amongst others) that uppeak traffic is the most stressed traffic that is produced in a high-rise building. Thus, according to that we undertake the analysis of our algorithm under such conditions.

In general terms, the main decision the controller has to take is to determine which car from the EGCS (one specific lift from the elevator group) must be assigned to a call. That is, once a passenger wants to travel from a floor to another different floor in a building, and the passenger makes a hall call of an elevator by pressing a landing call button installed at the floor and located near the cars of the elevator group, the EGCS must identify the elevator in the group that is most suitable to serve the passenger. Thus, the problem to be solved is to select an elevator for each hall call that is issued. This problem is called as the car-call allocation problem.

The optimization of such problem is mainly based on the minimization of the average journey time (AJT), which is the time in seconds that a passenger spends travelling to a destination floor measured from the instant of call registration to the instant passenger steps onto the destination floor. AJT time consists of the average travel time (ATT), which consists of the car travel time from the origin floor to the destination plus the average waiting time (AWT), which consists of the time in seconds that a passenger waits for service measured from the instant a passenger registers a call to the instant the passenger enters an elevator cars. See Barney *et al*. (1985) and the Cibse Guide D on Transportation systems in buildings (2000), which probably constitute some of the most recognized handbooks in vertical transportation. These parameters are also discussed in the simulation suite that is proposed in Cortés et al. (2006). Recently, some authors (Tyni *et al.* 2006 or Hasan *et al.* 2012) are considering the energy consumption of the system as objective function for low traffic situations and basing its analysis on the use of the energy generation by the counterweight. It is called the energy problem of the vertical transportation system.

In this paper, we focused on the car-call allocation problem is NP-Hard independently of the criterion. Thus, most of the approaches are focused on soft computing techniques. In fact, vertical transportation has become a major field of application for soft computing approaches such as fuzzy logic, neural networks, genetic algorithms, etc. (see section 2). All of them are techniques capable of providing better solutions than traditional controllers implementing dispatch expert rules that make use of simple IF-ELSE logical command sets.

The rest of the paper follows with the presentation is organized as follows: section 2 describes related work undertaken to solve the car-call allocation problem in elevator group control systems; the PSO algorithm that we implemented is described in section 3; the computer simulations attending to the average journey time and computational time for PSO algorithm and its comparison with genetic algorithm and tabu search approaches are dealt in section 3; and finally main conclusions are discussed in section 4.

2. Related work

As it has been previously addressed in section 1, the implementation of efficient control systems in elevator groups is absolutely required for tall buildings. Such type of systems are called Elevator Group Control Systems (EGCS), and although this is a young field of research (accordingly with the recent evolution of the electronic) it can be found relevant references in the scientific literature.

Most of modern EGCSs implement complex algorithms based on different soft computing techniques. Genetic algorithms appeared as one of the first attempts to tackle with such problem, and since then have been widely used providing good and valuable results. Examples of that are Cortés *et al.* (2003) and Cortés *et al.* (2004), where the authors implemented a binary genetic encoding for feasible solutions that has been widely followed by several authors since then. The algorithm was compared to traditional industry collective controllers (that are not based on soft computing approaches) providing significant improvements. The comparison was undertaken using simulation ARENA software. The main problem of these approaches was the required computation time required for the microchips of the controllers. After that, genetic approaches continued being applied. It was the case of Tyni *et al.* (2006) that developed a bi-objective genetic algorithm that optimized the waiting times and the energy consumption for the KONE Corporation. Later, Hirasawa *et al.* (2008) applied another genetic approach to a specific type of elevators that are called double-deck where two cages are connected in a shaft have been developed for the rising demand of more efficient transport of passengers in high-rise buildings. Here, the authors develop a graph-based evolutionary method named genetic network programming that introduce various node functions that can be easily executed by an efficient rule-based group supervisory control that is optimized in an evolutionary way. More recently, Bolat *et al.* (2010) developed a genetic algorithm based on the previous work of Cortés *et al.* (2004) providing a novel way to compute the average waiting time of passengers that allowed a computationally effective evaluation, and overcoming that limitation.

Tabu search has attracted less attention than genetic algorithms, although recently two algorithms based on deterministic and probabilistic approaches have been presented to deal with the problem. Bolat *et al.*, (2011) have provided a deterministic and probabilistic approach for the tabu search algorithm that allows outperform the results provided by the equivalent genetic implementation.

Li *et al.* (2007) has tried immune systems based on a two-level control structure. One structure is the locally optimal assignment of a hall call performed by a conventional collective algorithm; the other is the globally optimal assignment of all hall calls, which is executed periodically by artificial immune algorithm. This represents one of the very few approaches based on immune systems to deal with vertical transportation problems.

Control methodologies have been applied to EGCS too. It is the case of neural networks that were very enthusiastically tried in the origins of the discipline (as it was the case of Imrak *et al.*, 2001). More recently, the same author (Imrak, 2008) has presented an evolution of his first approach and has compared it with industry conventional approaches. The paper shows how the EGCS can predict the next stopping floors to stop by considering what has been learnt by processing the changes in passenger service demand pattern. Echavarria *et al.* (2009) have developed a feed-forward neural network based control algorithm has been developed that can approximate elevator call patterns by learning to associate time of day with specific call

locations. A brief comparison of the method allows anticipating suitability when comparing to fuzzy approaches, although future research is required to guarantee such claim. More recently, Dursun (2010) has implemented a neural network to control the EGCS for cases of external buttons. It corresponds to the case of vertical transportation zoning approaches. The approach provided advantages to conventional static zoning approaches.

Although fuzzy logic was also considered since a long time ago (see Kim *et al.*, 1998 for one of the first attempts to deal with fuzzy controllers), nowadays, the extensive use of fuzzy logic is being implemented in an enthusiastic way. It is the case of Jamaludin *et al.* (2010) that have presented a fuzzy controller that instead of depending heavily on the predicted passenger traffic pattern for adaptation, the fuzzy logic group controller adjusts itself to suit the system's environment through a self-tuning scheme. Results are simulated and compared to conventional approaches showing a significant improvement. Later, Cortes *et al.* (2011) have developed a fuzzy controller to forecast the traffic patterns in the vertical transportation system. The controller allows to identify whether systems is under an uppeak, downpeak, lunchpeak or interfloor pattern. Recently, Chen *et al*. (2012) have presented a fuzzy logic approach to control the EGCS self-tuned by a genetic algorithm to maximize service quality and managing a wide set of variables such as the number of elevators, traffic flow, direction, congestion, priority of floor, and preference of passengers, amongst others.

Here we present a particle swarm optimization (PSO) algorithm based on a hall call allocation strategy to define the solution encoding and computationally effective car-call allocation quality estimation. It constitutes a novel application of PSO to a new relevant industry sector. Approaches based on PSO are scarce in the literature. Li (2010) PSO algorithm was an exception. Li presented a PSO algorithm that is used to optimize typical zoning elevator problems where the floor sector that is going to be served by each elevator car has to be stated. The approach shows good results when comparing to other techniques.

Zoning (or sectoring) approach for vertical transportation is a technique used to serve skyscrapers during the uppeak traffic. It divides the building into different sectors and assign one (or a group of) car to each sector. When the car leaves the passengers in the floors assigned to its zone, the car go down to collect more passengers travelling to the floors of the zone. This technique is only applied for uppeak conditions because it worsens the EGCS performance during other patterns (downpeak, lunchpeak or interfloor). It also requires the installation of external button box with all the floor destination number in the hall of the car. Thus, it relates to other vertical transportation philosophy of management. This is called the zoning problem at difference from the dispatching problem (or car-call allocation problem). When using zoning approaches, one of the main parameters to be optimised is the round trip time (RTT), which measures the time required to take a passenger from the ground floor, go up to the highest floor and come back to the ground floor. Hence, RTT formulas can only be applied to up traffic where passengers enter to the lobby and are destined to upper floors.

Therefore, due to these reasons, we cannot compare our algorithm that is configured for a traditional button box with only up and down buttons and that do not consider zoning approach for dispatching cars, but a global EGCS where all the cars serve all the floors of the building. However, the good results provided by PSO to the zoning problem, lead us to try with the approach for the car-call allocation problem providing outstanding results as it is shown in the result section. To have benchmarks capable of assessing our proposal, we compare it against identical objective function implementations and the same building configuration for genetic

algorithms and tabu search approaches. In this line, results are provided for high-rise buildings from 10 to 24 floors, and several car configurations from 2 to 6 cars.

3. The particle swarm optimization algorithm for EGCS

In real buildings, passengers arrive randomly to different floors, even at the same time, wishing to be transported from a floor to other one. In addition, buildings show specific movement of passengers that can determine the flow pattern in the building. Four main patterns are traditionally catalogued: (i) uppeak traffic when a larger than average number of up landing calls are produced (typically because the building's workers arriving to start work); downpeak traffic when a larger than average number of down landing calls takes place (because building's population wishes to go back home after the working day); (iii) lunchpeak traffic that takes place in the middle of the day, and it is due to the appearance of up and down peaks; and finally (iv) interfloor traffic corresponding to the rest of the day. The latter phenomenon can be characterised for a low demand (usually around 4% of the population) in both directions.

The elevator group control systems determine which car of the group should serve a hall call. If the hall call is allocated to the most appropriate car, the passengers' travel and waiting times are reduced. We develop here a PSO algorithm that outperforms other soft computing implementations such as genetic algorithm or tabu search. Next we develop the solution encoding for the characterization of proposed solutions that is described in next subsection 3.1; the way to assess the car-call allocation, i.e., the evaluation of the candidate solutions (what is called as fitness); and the detailed description of the algorithm and its flowchart.

3.1. Car-call allocation solution encoding

Hall calls are encoded using a 2×(Number of floors – 1) elements array. The first half of the array corresponds to upwards landing calls, and the second half corresponds to downwards landing calls. A specific car from the elevator group must be allocated to each requested hall call, and a solution of the car-call allocation problem consists of the allocation of a specific car of the elevator group to all the hall calls being requested in the building. Therefore, the dimension of the solution becomes equal to number of requests in the hall call array ('ones' in Figure 1.a), as no elevator needs to be assigned to floors without requests. So, the dimension of the solution encoding can change according to the different hall call request configurations.

Figure 1 (a) and (b) provides an example of car-call allocation solution encoding for a building with 10 floors and 3 elevators. Figure 1 (a) relates to the hall call requested at the floors. It is represented by an array of 18 elements, 9 for upwards landing calls, and 9 for downwards landing calls. A number equal to '1' means that there is a call, and a '0' means that there are not calls at that floor. Figure 1 (b) relates to the solution encoding of the car-call allocation problem. In the example, Figure 1 (a) shows that the required solution encoding will have a dimension equal to 13; such dimension corresponds to the requested hall calls. A solution consists of the allocation of a car of the elevator group (composed of three cars) to a hall call. The solution encoding in the example shows the number of the car that is assigned. Hence, each element can get a value between 1 and 3 since we have 3 elevators.

Upwards Landing Calls									Downwards Landing Calls								
F1	F2	F3	F4	F5	F6	F7	F8	F9	F2	F3	F4	F5	F6	F7	F8	F9	F10
1	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0

(a)

F1	F3	F4	F6	F7	F8	F2	F3	F4	F5	F7	F8	F9
3	1	2	3	3	1	2	1	2	3	2	1	3

(b)

Figure 1. (a) An example of hall call encoding for a building of 10 floors. (b) Corresponding solution encoding for an elevator group with 3 cars.

3.2. Car-call allocation fitness evaluation

To evaluate the quality of a car-call allocation by the EGCS is, we evaluate the fitness of such allocation. As any possible allocation must be evaluated, computationally effective is of enormous relevance. Therefore, the method to evaluate the fitness function is a very important issue. In addition, the fitness has to provide an adequate value of the performance of the car-call allocation. One of the most time-implemented dispatchers in the elevator industry is the THV one (see Cortés *et al.,* 2003). THV algorithm was implemented at UMIST (University of Manchester Institute of Science and Technology), and assigns the hall call to the nearest lift in the adequate trip direction (sometimes appears referred as nearest call algorithm). It is easy to be implemented but does not provide high quality estimations of the proposed allocation. Another common implementation is due to the estimated time of arrival (*even at the same time, wanting*) algorithm, which undertakes an estimation of the required time since the landing call is issued until the car arrives (Barney *et al.*, 1985). ETA includes different levels of priority: (*i*) long waiting calls; (*ii*) high activity floors; (*iii*) priority levels; and (*iv*) remaining calls. Those cars attending calls with priority level one to three do not stop at landing calls and only serve car calls requiring a special AJT calculation. ETA algorithm provides a suitable behavior during uppeak traffic, a medium level service for interfloor, and a bad level of service during lunchpeak and specially downpeak. These approaches were tested in Cortes *et al*. 2004, Bolat *et al*. 2010 and it was appreciated that the Bolat *et al*. (2011) fitness evaluation proposal outperformed the other previously described approaches both in quality of solutions and computational speed. So then, we opted to follow the methodology described in Bolat *et al*. (2011) as an easy-to-implement and fast-to-compute technique, providing the better index of performance of the system in a suitable time of response. It is described next.

Given the following parameters:

- Ψ_1: ground floor level
- Ψ_2: highest down hall call level
- Ψ_3: number of down hall calls between Ψ_1 and Ψ_2.
- Ψ_4: highest up hall call level
- Ψ_5: number of up hall calls between Ψ_1 and Ψ_4.
- Ψ_6: lowest down hall call level
- t: door opening and closing time
- t_p: passenger transfer time
- *Hct*: Highest car trip time

- *Lct*: lowest car trip time

The solution fitness, *f*, is calculated depending on the type of passengers' movements. So, a fitness value is firstly calculated for each car, *i*, in the group by considering four different cases that are shown in figure 2. Note that each formula relates in which floor the passenger is taken and into which floor the passenger is transferred. Hence, as a general definition for Ψ, it shows the floors where passengers are getting on and off.

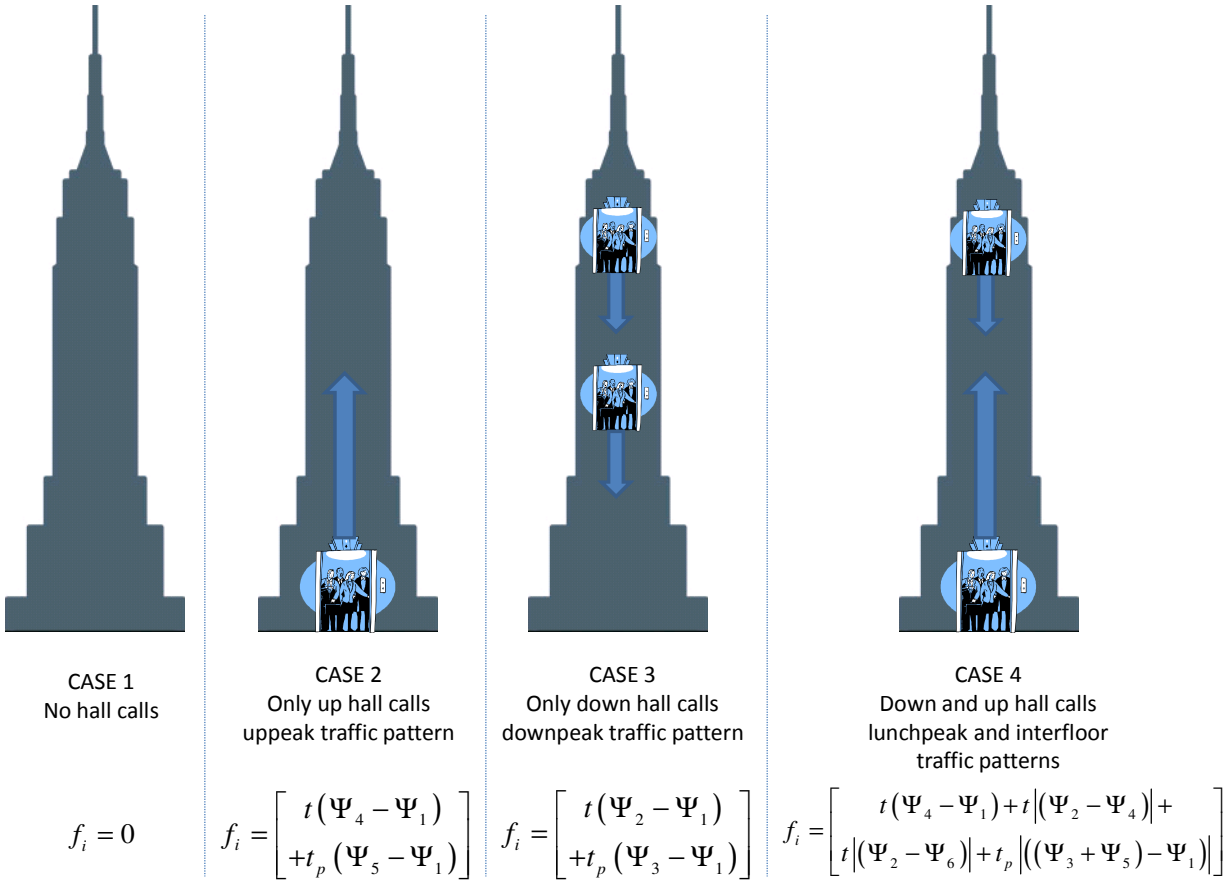

Figure 1. Fitness estimation depending on the traffic pattern

Finally the group fitness is calculated, see equation (1), and the final fitness is evaluated for the proposed allocation, see equation (2). That is, the total fitness of the system is calculated as a two-part function where the first part collects the concern related to passengers waiting in the halls, and the second part concern related to an unbalanced performance in the cars of the group. That is, the second part tries to level the use of each car. Let's consider the following example: car no.1 is working during 100 seconds and car no.2 during 11 seconds, so both cars would be working during 111 seconds but in a much unlevelled manner. On the other hand if car no.1 works during 57 seconds and car no.2 during 54 seconds, the two cars work during 111 seconds but in a much more levelled manner. This action will have repercussion on the life cycle of the cars. Although k_1 and k_2 are design parameters, and variations and a discretional criterion can be admitted, we select them equal to 1.5 and 2 accordingly with Bolat *et al.* (2011).

$$f_{group} = \frac{\sum_{i=1}^{n} f_i}{n}, \text{being } n \text{ the number of cars in the group} \quad (1)$$

$$f = k_1 \cdot f_{group} + k_2 \cdot (Hct - Lct) \quad (2)$$

3.3. Particle swarm optimization algorithm

Particle swarm optimization, presented in Kennedy *et al*. (1995), is a population based stochastic optimization technique inspired by social behaviour of bird flocking or fish schooling.

PSO is a population based technique. The system is initialized with a population of particles that correspond to initial solutions of the problem. These particles move in the search space searching for optimal fitness value.

Following Coelho (2010), PSO algorithm defines particle as a potential solution represented by an *s*-dimensional vector, where *s* is the number of optimization variables. The swarm concept relates to an apparently disorganized population of moving particles that tends to cluster together while each particle seems to be moving in a random direction. The particle best position is calculated for a particle moving through the search space, it compares the fitness value at the current position to the best fitness value it has ever attained at any time up to current time. Then, the global best relates to the best position among all individual best position; and the velocity of the particle flight represents the velocity of the particle in the physical analogy, taking into that all the particles velocities and positions are updated after each iteration.

We implemented a PSO algorithm that is described by the flowchart in Figure 3. The steps of the flowchart are numbered for easy referring. Steps 1 to 4 initialize variables that change in the main loop in a suitable way before entering into the main loop (step 5). In step 1, each particle is "thrown" to some random place in the search space: each particle is given a random position. Then, each particle gets a random velocity (step 2). Since a particle has a position and a given velocity, its next position can be calculated in the first iteration of the main loop following the rules and calculations detailed in steps 8 and 9 later. In step 3, the fitness value of each particle's initial position is calculated. The best of these initial positions is assigned to the variable "global best" (step 4). Steps 5 to 14 constitute the main loop of the algorithm. In step 5, it is decided whether another iteration of the loop is undertaken or not by checking the number of iterations. Once the maximum number or iterations is reached, the procedure gets out of the loop and proceeds to step 15. Step 6 makes sure that all the particles are processed. In step 7, we get the next unprocessed particle, and proceed with it.

Let us call this particle the "current" particle. In step 8, a new velocity is set for the current particle using next equation (3):

$$v_{j+1} = w_1 v_j + w_2 r_1 \Delta l + w_3 r_2 \Delta g \,, \quad (3)$$

where v_{j+1} is the new velocity vector, v_j is the current velocity vector, w_1, w_2, and w_3 are constant scalars (weights), r_1 and r_2 are vectors whose elements are random values between

0 and 1, Δl is the difference vector between the best and current position of the particle, and Δg is the difference vector between the global best position and the particles current position. This equation explains that the new velocity of the particle is calculated using three values: the current velocity, the distance of the particle to its best position, and its distance to the global best position. Weights w_1, w_2, and w_3 allow us to set relative importance of the three terms in determining new velocity. After several tests, we concluded that no part of equation (3) should be prioritized to another in order to get the best performance of the algorithm. So then, values were set to 0.34, 0.33, and 0.33 to make them roughly equal, and to make them sum up to 1.

In step 9, the current position together with the new velocity determines the new position (each iteration was assumed as a unit time). In some occasions, the new calculated position could not be valid, e.g. they are out of the bounds of the search space. This fact is checked in step 10, and if the position is not valid a random valid position is assigned to the particle. And the new velocity value is re-updated according to this new position (step 11). Steps 12 to 14 are for bookkeeping and preparation for the next iteration. Step 12 calculates the fitness value of the new position. If necessary, the particle's best position and global best position are also updated (steps 13 and 14). When the maximum number of iterations is reached, the procedure gets out of the main loop and moves to next step 15. The global best retains the result of the optimization.

After testing several values, we realized that a maximum number of iterations equal to 30 were enough to guarantee convergence without penalising computational times. In a similar line, a number of 30 particles guaranteed a suitable mapping of the alternative solutions. Increasing the iterations and the number of particles did not report significant improvements meanwhile the required time to run the algorithm increased in values out of real applicability in the elevator industry.

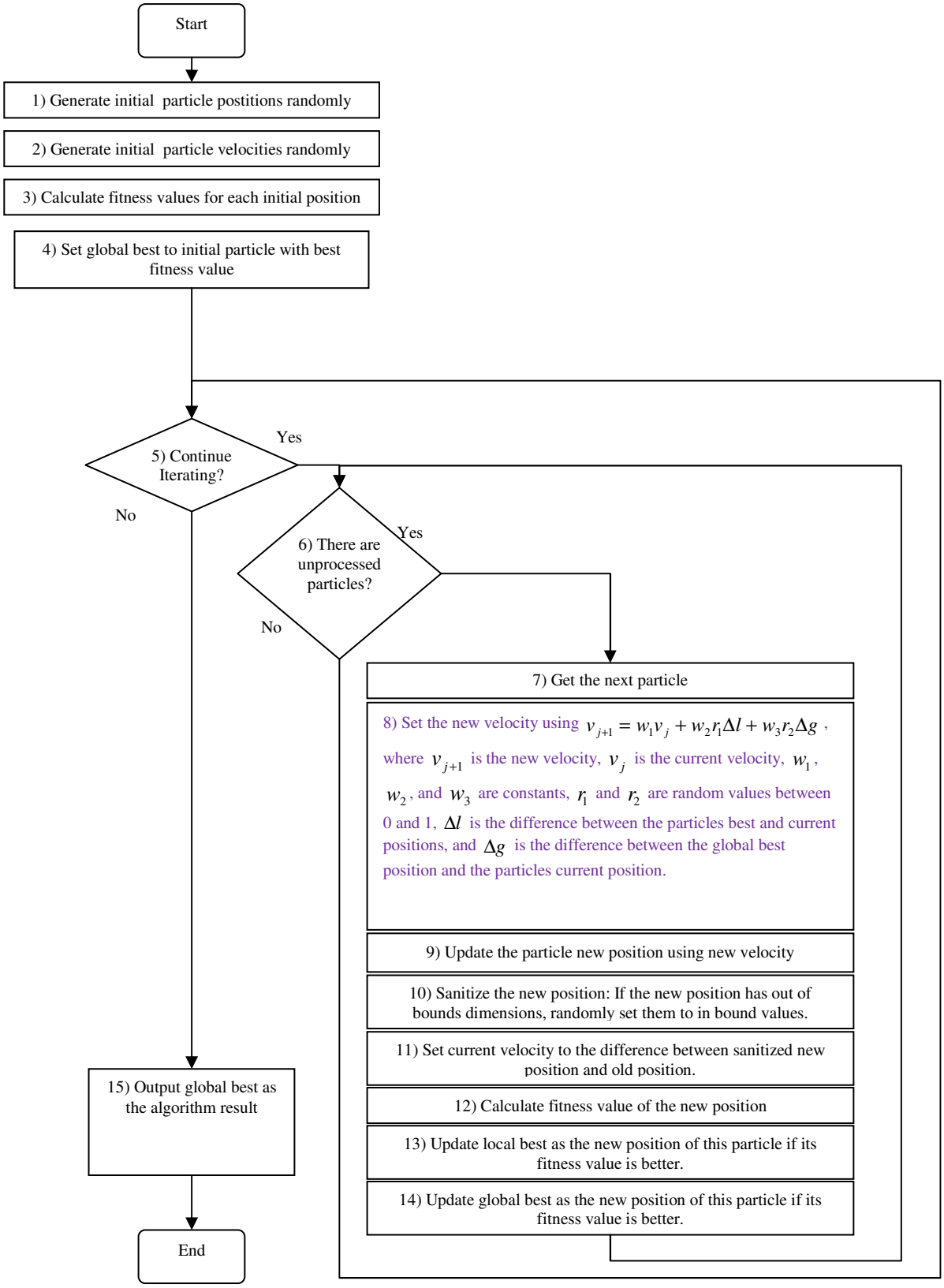

Figure 2. Particle swarm optimization algorithm flowchart

4. Computer Simulations

4.1. Simulation scenario

High-rise buildings from 10 to 24 floors were selected for our case study. We also considered different car configurations for the EGCS: 2, 3, 4 and 6 cars. The configurations were selected following the general rules provided in Barney *et al*. (1985) and the last edition of the CIBSE Guide edited by the Chartered Institution of Building Services Engineers (CIBSE, 2010). They represent a height that make them qualified as tall buildings. Table 1 and 2 give the main specifications of the building and elevators respectively. Total time of a car stopping is given by $t_s = t_{open}+t_{close}+t_{p,}$ where t_{open} is the time for opening door, t_{close} is the time for closing door, and t_p is the time for passenger transfer.

Table 1. Specifications of the buildings

Items for building	Value
Number of floors	from 10 to 24
Number of cars	2,3,4,6
Floor distance (m)	3.3

Table 2. Specifications of the elevator system

Items for elevator system	Value
Car capacity (people)	8
Speed (m/s)	3
Time for opening door (s)	3
Time for closing door (s)	3
Time for passenger transfer (s)	3

We carried experiments on a machine with 2.33GHz (Intel Core2 Quad) CPU and 3GB RAM running Microsoft Windows 7. For our application we use only one core.

4.2. Simulation results

In this section, we provide test results for different car group configurations as specified in Table 2. We have compared our PSO algorithm with the genetic algorithm implementation provided in Cortes et al. (2004) and subsequently modified in Bolat *et al.* (2010), and with the tabu search presented in Bolat *et al.* (2011). Average journey time analysis is provided in Tables 3 and 4 for each approach for buildings that have 10 to 24 floors and 2 to 6 cars.

We run the algorithms 6 times. We selected this number after testing the algorithms many times, once we checked that more runs did not provide different values, so then six could be selected as an adequate number of replications to make the final analysis and comparison study. Average AJT provides the arithmetic mean, best AJT the best value that was obtained, and SD represents the standard deviation for the experiments. As can be viewed, the relation between the mean and the standard deviation presents a ratio around 5%, which adds robustness to the analysis in terms of the mean.

Table 3 and 4, allow us to state that PSO outperforms in general terms TS and GA. In terms of both average value and best-obtained value PSO test results are better than the others for all configurations (two, three, four and six cars) and for practically all the floors. We have to note the exception that can be appreciated in the 2 cars / 13 floors and 2 cars /20 floors, where the genetic approach provides better results of AJT. It is an exception to the general rule that shows how PSO outperforms genetic algorithm, although it has to be noted that for these cases the standard deviation was better for the PSO algorithm.

Another relevant aspect is the obtained improvement regarding the PSO AJT with respect to the other approaches. The advantage of the PSO increases when the problem increases its difficulty. It is frequent to appreciate the better results of the PSO for complex cases implying a large configuration (6 cars) and tall buildings (more than 20 floors), as can be viewed in table 4. In fact, when dealing with simple configurations most of methods can provide acceptable solutions (even in case of basic dispatching algorithms). However, when the complexity is increased, only advanced methods can provide acceptable solutions. It is under these stressed situations when the quality of a method can be better appreciated. And it is under these circumstances that PSO provided better values than GA or TS.

The variability of the solutions provided by the three approaches is characterized by the standard deviation. In general terms, we have to state that standard deviations have similar values for all the algorithms due to the combinatorial character of the soft computing approaches. Indeed, the approaches include a random component that makes them vary from the mean necessarily. Anyhow, PSO showed a more robust behaviour than TS and most of the times than GA in terms of standard deviation too. In fact, the best result provided by the different approaches was many times closer to the best results of all the simulations in the PSO approach than in the other approaches. That is, the other methods (GA and TS) varied in a longer interval (provided by the best value, best AJT, and the standard deviation, SD AJT) for the undertaken simulations than PSO providing many times worse results.

Table 3. AJT comparison for 2 and 3 cars' configuration (in seconds)

Num. of floors	2cars									3cars								
	Average AJT			SD AJT			Best AJT			Average AJT			SD AJT			Best AJT		
	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS
10	50.5	51	51.5	1.2247	4.6476	3.5071	48	45	45	33	39	42	3.2863	7.0993	7.0993	30	33	33
11	56.5	60	60	3.5071	3.7947	7.5895	51	57	45	43.5	49	47	4.1352	4.899	5.8992	36	42	39
12	64	66.5	64.5	5.2536	4.8062	6.2209	57	60	57	48	56	60	6.8411	8.1976	1.8974	42	48	57
13	73.5	72	77	4.1352	7.5895	4.0988	66	63	72	51	59.5	63	7.823	9.5656	3.7947	45	48	60
14	77	81.5	83.5	9.2304	3.5071	4.8062	63	78	75	61	71.5	71	4.5166	9.3755	5.8992	54	54	60
15	82	85.5	86.5	7.2664	3.1464	4.4159	69	81	81	69.5	73	74.5	4.4159	5.8992	8.7807	63	66	60
16	87.5	91.5	92	3.5071	7.0356	5.2536	84	81	84	71	82	85.5	4.0988	6.4807	8.4321	63	75	72
17	100.5	106.5	102	5.6125	3.1464	7.0993	93	102	93	75	84.5	90.5	8.8994	6.9498	6.9498	66	72	81
18	104.5	109	111.5	3.9875	1.5492	1.2247	99	108	111	82	93.5	94	6.1968	6.9498	6.1968	75	87	87
19	111	111.5	113.5	5.02	4.4159	4.8062	102	105	108	87.5	101.5	98.5	4.4159	4.8062	6.9498	81	96	87
20	117.5	114	121	8.3606	11.0635	2.4495	108	102	117	98.5	105.5	106.5	4.8062	9.9348	4.5497	90	93	99
21	119	125.5	129	7.2664	7.9183	3.7947	111	111	123	104	115.5	115.5	5.2536	4.1352	4.5497	96	108	108
22	128.5	132	136	8.9833	9.4868	6.4807	117	117	126	108.5	118	113.5	7.6877	10.1587	6.1237	99	99	102
23	132.5	136.5	137.5	7.9183	8.6429	5.8224	123	123	129	110.5	124	126.5	11.131	4.899	8.7807	93	117	117
24	144	146.5	148	9.4868	8.7807	2.4495	126	132	144	122.5	136	133.5	4.4159	5.5857	9.2466	117	129	117

Table 4. AJT comparison for 4 and 6 cars' configuration (in seconds)

Num. of Floors	4 cars									6 cars								
	Average AJT			SD AJT			Best AJT			Average AJT			SD AJT			Best AJT		
	PSO	GA	TA	PSO	GA	TA	PSO	GA	TA	PSO	GA	TA	PSO	GA	TA	PSO	GA	TA
10	28	35.5	36	1.5492	3.9875	6	27	30	30	24	27	29.5	1.8974	2.6833	3.5071	21	24	24
11	34	38.5	44.5	1.5492	2.9496	7.4498	33	36	33	30.5	33	36.5	1.2247	3.7947	2.2583	30	30	33
12	40	51.5	51	2.4495	3.5071	5.6921	36	48	45	34.5	40	39.5	3.1464	2.4495	6.4109	30	36	33
13	45.5	58	58.5	2.2583	7.2664	9.6281	42	48	48	40.5	46	49	2.515	5.2536	2.4495	39	39	45
14	49.5	57.5	64.5	5.9245	6.9498	4.5497	45	51	57	43.5	50.5	55.5	2.515	6.4109	10.8582	42	42	42
15	55.5	70.5	73	6.775	2.51	9.0333	48	69	60	44	55.5	62.5	2.4495	6.2209	7.2042	42	48	51
16	61.5	73	79	9.4393	5.2536	8.8318	51	66	69	52	71.5	71	2.4495	3.9875	4.5166	48	69	63
17	71	77.5	88	9.0333	7.4498	11.1714	60	69	72	57	67.5	76	1.8974	8.4321	5.8992	54	60	69
18	76.5	85	85.5	7.0356	6.4807	1.6432	72	75	84	58.5	72	81	2.515	3.7947	6.8411	54	66	75
19	84	95.5	96	6	7.4498	6.8411	75	87	87	62	74.5	86	1.5492	4.4159	9.798	60	69	75
20	83.5	96.5	96	4.4159	8.3606	9.4868	78	81	78	69	84	84	6	9.6747	10.0399	60	69	72
21	92.5	101.5	110.5	6.1237	6.9498	6.1237	84	90	105	77	88	104	7.2664	5.5857	5.8992	69	78	96
22	98.5	112.5	114.5	11.7601	8.6429	11.2916	84	102	96	81	99	115.5	3.2863	12	12.6925	78	81	96
23	98.5	116	117	9.5656	2.4495	11.0635	84	114	99	93.5	102	113	11.131	9.2952	18.942	78	90	75
24	112	127	130	6.1968	6.7528	4.0988	102	120	123	86	117.5	121	9.2304	9.7519	7.975	72	99	105

Figure 4 shows an expected increase in the AJT when the number of floors grows and the number of cars decreases. The reduction of AJT with respect to the increase in the number of cars follows a quasi-linear tendency, because more cars lead to a better service in univocal. However, the variation with the number of floors is less predictable (due to the random arrivals of passengers to the floors) although with a clear increasing tendency.

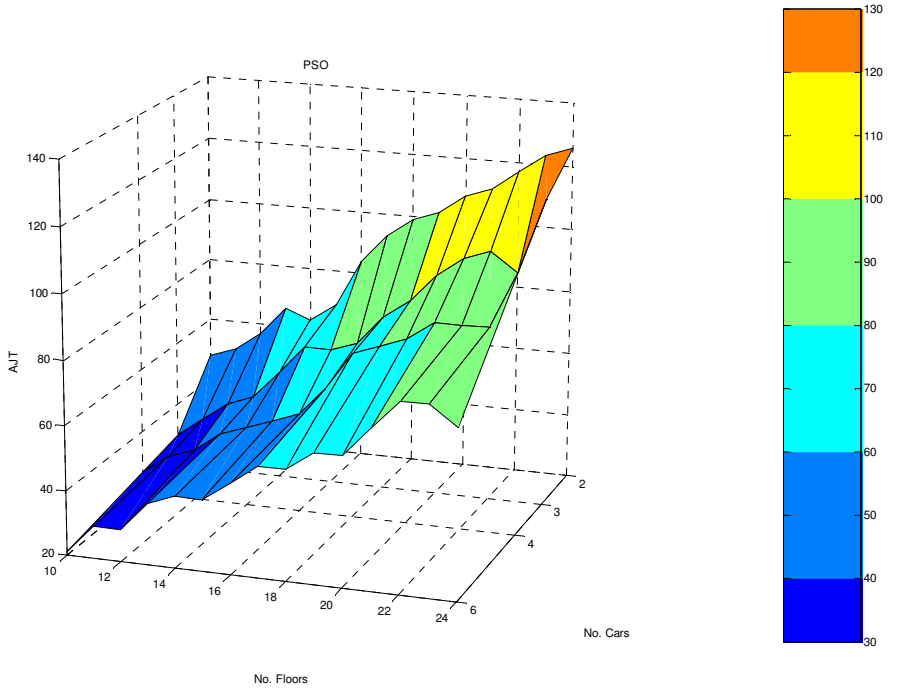

Figure 3. Best AJT results variation with respect to number of floors and car configuration in the EGCS

Regarding the computational time required for the execution of the algorithms, Tables 5 and 6 show the obtained values for all the approaches with respect to the number of floors of the building and the cars' configuration of the EGCS.

TS produced the quickest results for low cars' configurations (2 and 3 cars) and PSO quicker results than GA as a general rule. However, for larger configurations of cars (4 and 6 cars) PSO was the fastest approach to solve the problem. Thus, for complex systems PSO showed faster computation than the other methodologies. This adds robustness to the PSO approach that we developed, and reveals it as a suitable dispatcher when dealing with highly complex building.

Table 5. Computational time comparison for 2 and 3 cars' configuration (in seconds)

Num. of floors	2 cars						3 cars					
	Average Comp. Time			SD Comp. Time			Average Comp. Time			SD Comp. Time		
	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS
10	0.09165	0.1067	0.04362	0.0012315	0.0025211	0.0039119	0.098535	0.13425	0.06699	0.00050272	0.0015377	0.0040434
11	0.091384	0.11263	0.057458	0.0001369	0.0030153	0.041875	0.09964	0.13925	0.069127	0.00099552	0.002483	0.0013116
12	0.091934	0.11334	0.039024	0.0002953	0.0015726	0.0012413	0.10013	0.14112	0.069938	0.00056182	0.0030143	0.0033965
13	0.09281	0.11613	0.059107	0.00051594	0.0021316	0.046707	0.10046	0.14647	0.069873	0.00046022	0.0018993	0.0037979
14	0.092371	0.11868	0.039047	0.00026921	0.0018817	0.001489	0.10107	0.15127	0.070838	0.00024622	0.0024427	0.0035364
15	0.092698	0.122	0.040319	0.00022865	0.0023994	0.0025782	0.10164	0.15506	0.071975	0.00014657	0.0020194	0.003563
16	0.092886	0.12566	0.059817	0.0002612	0.0017492	0.044805	0.10232	0.16006	0.068473	0.00029236	0.0022192	0.0024951
17	0.093138	0.12648	0.038633	0.00011625	0.0031815	0.0020637	0.10243	0.1622	0.069635	0.00024742	0.0023478	0.0044573
18	0.093261	0.13094	0.040271	0.00024043	0.002626	0.0030197	0.10234	0.16584	0.06855	0.00018732	0.001619	0.0032402
19	0.093508	0.13102	0.038971	9.2921e-005	0.0027765	0.0014896	0.10304	0.1656	0.070114	0.00011068	0.0055216	0.0023473
20	0.093923	0.13442	0.039405	0.00025867	0.0012189	0.001644	0.10359	0.17238	0.072942	0.00024677	0.0017868	0.0031705
21	0.09428	0.1386	0.040831	0.00021424	0.0024249	0.002334	0.10394	0.17725	0.070167	0.00019275	0.0020878	0.0027
22	0.094017	0.1407	0.039659	0.00015839	0.0012717	0.0030056	0.10405	0.18126	0.071361	0.00043271	0.0040127	0.0037535
23	0.094821	0.14478	0.039874	0.0005905	0.0022684	0.0028439	0.10491	0.1833	0.070392	0.00071336	0.0027484	0.0036012
24	0.094677	0.14463	0.042606	0.00027286	0.0028016	0.0029168	0.1044	0.18901	0.072497	0.0001107	0.003163	0.002709

Table 6. Computational time comparison for 4 and 6 cars' configuration (in seconds)

Num. of floors	4 cars						6 Cars					
	Average Comp. Time			SD Comp. Time			Average Comp. Time			SD Comp. Time		
	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS	PSO	GA	TS
10	0.10475	0.15825	0.1202	0.00018015	0.0022947	0.0096018	0.11459	0.19727	0.31517	0.00017138	0.0014222	0.011704
11	0.10592	0.16323	0.12042	0.00029818	0.004744	0.005759	0.11604	0.20437	0.29796	0.0001243	0.0014699	0.04199
12	0.10673	0.16771	0.12648	0.00022179	0.0027201	0.0066631	0.11742	0.21022	0.3311	0.00041909	0.001958	0.010741
13	0.10769	0.17033	0.12256	0.00013718	0.0015645	0.0062888	0.11915	0.21677	0.31696	0.00025296	0.0025879	0.019271
14	0.10781	0.17671	0.12713	0.0002104	0.0030834	0.0086551	0.11971	0.22168	0.33321	0.00019081	0.001862	0.017419
15	0.10937	0.18225	0.12577	0.0010358	0.0020738	0.0045146	0.12072	0.2288	0.32129	0.00027924	0.0019329	0.026135
16	0.11029	0.18643	0.12672	0.0012647	0.0020034	0.0059848	0.12209	0.23424	0.33816	0.00025407	0.0029885	0.013306
17	0.11015	0.1894	0.12568	0.00020715	0.0022632	0.0044516	0.12292	0.23993	0.35357	0.00037531	0.0030672	0.016571
18	0.1104	0.19598	0.12769	0.00026058	0.0030638	0.0074558	0.12324	0.24508	0.34916	0.00068712	0.0011765	0.026119
19	0.11119	0.19823	0.13465	0.00017876	0.0034357	0.0077959	0.12438	0.25236	0.3559	0.00032028	0.0020994	0.019683
20	0.11194	0.20452	0.13632	0.00011599	0.0017819	0.0063289	0.1254	0.25784	0.37479	0.00035846	0.0023494	0.023016
21	0.11253	0.20654	0.12922	0.00033222	0.0018198	0.0081645	0.12666	0.26353	0.35121	0.00042655	0.0023575	0.0086219
22	0.11279	0.21573	0.12824	0.00026371	0.00338	0.0029092	0.12708	0.26993	0.37179	0.00038169	0.0036077	0.026862
23	0.11323	0.2188	0.13716	0.00021016	0.0038409	0.0063958	0.128	0.27306	0.37377	0.00022468	0.0041186	0.010537
24	0.11358	0.22144	0.14021	0.00011879	0.0035743	0.006558	0.12875	0.28103	0.3799	0.00033332	0.0029461	0.020579

Finally, Figure 5 represents the average car working time for each car configuration with respect to the number of floors of the building. That is, the time as average that a car of each configuration (2, 3, 4 or 6 cars) is working. It can be observed how the value of the average car working time is significantly reduced for larger configurations. This aspect will affect the life cycle of the cars, so it is important and relevant. Management teams of EGCSs should consider the required investment in the EGCS depending on the number of cars with respect to the operation costs and the total life cycle cost when doing real monetary calculations.

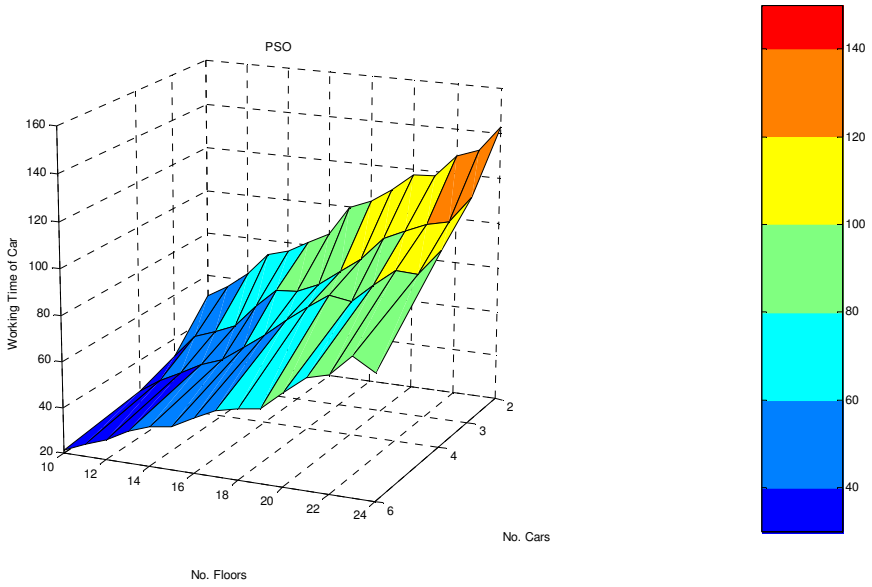

Figure 4. Average car working time with respect to number of floors and car configuration in the EGCS

5. Conclusion

We have presented a novel application of PSO algorithm to optimize the car-call allocation strategy of the controller in Elevator Group Control Systems. PSO algorithms had been tried in vertical transportation problems to deal with sectoring problems, which is a technique used to serve skyscrapers during the uppeak traffic dividing the building into different sectors and assigning one (or a group of) car to each sector. Our PSO implementation to solve the car-call allocation problem, also known as the dispatching problem, constitutes a novelty in the elevator scientific literature.

Our implementation is based on a hall call allocation strategy to define the solution encoding and includes computationally effective car-call allocation quality estimation. Results were provided for high-rise buildings from 10 to 24 floors, and several car configurations from 2 to 6 cars. The algorithm was successfully applied to all the case studies outperforming other soft computing techniques such as genetic algorithms and tabu search. Results were better attending to the average journey time (that includes the waiting plus travel times) as well as attending to the algorithm computational times. Even more the robustness of the method (measured as the standard deviation) was also proven. Hence, it was shown that PSO got better results in a faster, cheaper way compared with the other methods. Another reason that made PSO more attractive with respect to other techniques is its capability to be adjusted with very few parameters, which adds additional robustness.

However, results obtained with a CPU can be limited with respect the real industry operation. In practice, real implementation should be installed in specific microchips, which would require lighter implementations. Bounding this fact, the real implementation of the PSO algorithm in the industry appears to be possible. For example, in real cases an alternative can be calculating the fitness not every time but in a selective manner, or stopping the algorithm after a lower number of iterations. Of course all these decisions are very dependent on the computation speed of the electronic microchips installed by the company in the controller, and could affect the quality of the implemented algorithm.

Currently, our further research focuses on global controllers capable of identifying traffic pattern taking part in the building and launching the corresponding specialised algorithm. This global controller intends to incorporate the consideration of energy optimization for low demand patterns such as interfloor traffic pattern that can produce a significant reduction of energy consumption without worsening very much the quality of service indexes.

Acknowledgements

The Spanish author acknowledges the financial support given by the Counselling of Innovation, Science and Business of Andalusia, through its Excellence Projects Programme (project ref. P07-TEP-02832), and the Spanish Research Agency dependent on the Department of Science and Innovation, through its DPI Programme (project ref. DPI2010-15352).

References

[1] G. Barney, S. Dos Santos (1985) *Elevator traffic analysis design and control*, Peter Peregrinus Ltd, Londres.

[2] B. Bolat. P. Cortés, E. Yalcin, M. Alisverisci (2010) Optimal car dispatching for elevator groups using genetic algorithms, *International Journal of Intelligent Automation and Soft Computing*, 16(1), pp. 89-99.

[3] B. Bolat, P. Cortés (2011) Genetic and tabu search approaches for optimizing the hall call-car allocation problem in elevator group system, *Applied Soft Computing*, 11(2), pp.1792-1800.

[4] Chartered Institution of Building Services Engineers. CIBSE Guide D (2010) *Transportation systems in buildings*, London.

[5] T-C. Chen, Y-Y. Hsu, Y-J. Huang (2012). Optimizing the Intelligent Elevator Group Control System by Using Genetic Algorithm. *Advanced Science Letters* 9(1), pp. 957-962

[6] L.d.S. Coelho (2010) Gaussian quantum-behaved particle swarm optimization approachesfor constrained engineering design problems, *Expert Systems with Applications*, 37(2), pp.1676-1683.

[7] P. Cortes, J. Larrañeta, L. Onieva, (2003) A genetic algorithm for controlling elevator group systems. In: *Artificial Neural Nets Problem Solving Methods*, Part II Vol. 2687, pp, 313-320.

[8] P. Cortés, J. Larrañeta, L. Onieva (2004) Genetic algorithm for controllers in elevator groups: analysis and simulation during luncpeak traffic, *Applied Soft Computing*, 4(2), pp.159-174.

[9] P. Cortés, J. Muñuzuri, L.Onieva (2006) Design and analysis of a tool for planning and simulating dynamic vertical transport, *Simulation: Transactions of the Society for Modelling and Simulation International* 82(4), pp. 255-274.

[10] P. Cortés, J. Fernández, J. Guadix and J. Muñuzuri (2012) Fuzzy logic based controller for peak traffic detection in elevator systems, *Journal on Computational and Theoretical Nanoscience* 9 (2), pp. 310-318.

[11] M. Dursun (2010). Estimation of passenger waiting time in elevator systems with artificial neural network. *Intelligent Automation and Soft Computing* 16(1), pp. 101-110

[12] J. Echavarria and C. M. Frenz (2009). Improving Elevator Call Time Responsiveness via an Artificial Neural Network Control Mechanism. *Systems, Applications and Technology Conference*, 2009. LISAT '09. IEEE Long Island, pp. 1-3.

[13] M.Z. Hasan, R. Fink, M.R. Suyambu, M.K. Baskaran (2012). Assessment and improvement of intelligent controllers for elevator energy efficiency. *IEEE International Conference on Electro/Information Technology*, EIT 2012; Indianapolis, (Code 91401).

[14] K. Hirasawa, T. Eguchi, J. Zhou, L. Yu, J. Hu, S. Markon (2008) A double-deck elevator group supervisory control system using genetic network programming, *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev*., 38(4), pp. 535–550.

[15] C.E. Imrak (2008) Artificial neural networks application in duplex/triplex elevator group control system. *Journal of Mechanical Engineering* 54 (2), pp. 103-114.

[16] C.E. İmrak and G.C. Barney (2001) The Application of neural networks to lift traffic control, *Elevator World*, 49(5), p. 82.

[17] J. Jamaludin, N.A. Rahim, W.P. Hew (2010) An elevator group control system with a self-tuning fuzzy logic group controller, *IEEE Transactions on Industrial Electronics* 57 (12), pp. 4188-4198

[18] J. Kennedy, R.C. Eberhart (1995) Particle swarm optimization. In: *Proc. IEEE Int'l. Conf. on Neural Networks*, Vol. IV, 1942–1948. Piscataway, NJ: IEEE Service Center.

[19] C. B. Kim, K. A. Seong, H. L. Kwang, and J. O. Kim (1998) Design and implementation of a fuzzy elevator group control system, *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, 28(3), pp. 277–287.

[20] Z. Li, (2010) PSO-Based real time scheduling for elevator group supervisory control system, *Intelligent Automation and Soft Computing*, 16(1), pp.111-121.

[21] Z. Li,, H-Z.Tan, Y-N. Zhang, Z-Y. Mao (2007). Dynamic optimization of elevator group control based on artificial immune algorithm for inter-floor peak traffic during lunch-time, *Control Theory and Applications* 24(2), pp. 177-182.

[22] T. Tyni, J. Ylinen (2006) Evolutionary bi-objective optimisation in the elevator car routing problem. *European Journal of Operational Research* 169, pp. 960–977