An ACO Model for a Non-stationary Formulation of the Single Elevator Problem

S. Molina, G. Leguizamón
Universidad Nacional de San Luis,
Ejército de los Andes 950, (5700) San Luis, Argentina
{smolina,legui}@unsl.edu.ar
and
Enrique Alba
Universidad de Málaga
Complejo Tecnológico - Campus de Teatinos, 29071
Málaga, España
eat@lcc.uma.es

Abstract

The Ant Colony Optimization (ACO) metaheuristic is a bio-inspired approach for hard combinatorial optimization problems for stationary and non-stationary environments. In the ACO metaheuristic, a colony of artificial ants cooperate for finding high quality solutions in a reasonable time. An interesting example of a non-stationary combinatorial optimization problem is the Multiple Elevators Problem (MEP) which consists in finding a sequence of movements for each elevator to perform in a building so that to minimize, for instance, the users waiting average time. Events like the arrival of one new user to the elevator queue or the fault of one elevator dynamically produce changes of state in this problem. A subclass of MEP is the the so called Single Elevator Problem (SEP). In this work, we propose the design of an ACO model for the SEP that can be implemented as an Ant Colony System (ACS). *Keywords:* Ant Colony Optimization, Single Elevator Problem, Non-stationary Problems, Ant Colony System design.

1 Introduction

The Ant Colony Optimization metaheuristic (Dorigo et al. [7]) is a bio-inspired approach for hard combinatorial optimization problems for stationary and non-stationary environments. ACO algorithms, that is, algorithms that are designed according to the principles of the ACO metaheuristic are essentially approximation algorithms which are capable of reaching good quality solution (some time the optimal one) at a very low computational cost. The design of ACO algorithms is in general inspired by the behavior of some ant colonies species which are able to find the shortest path from the nest to the food source. Some of these ant colonies make use of an indirect communication approach, called *stigmergy* consisting in laying a pheromone trail on the ground. More specifically, ant algorithms involve a colony of artifical ants (or agents) working cooperatively and depositing artificial pheromone trail in order to share the search experience with the rest of the colony. The artificial ants are independent processes the stochastically and iteratively construct solutions for the problem at hand regarding: (i) heuristic information about the problem, if available, and (ii) the pheromone trail, which changes dynamically according to the search experience of the whole colony.

The Multiple Elevators Problem (MEP) is a non-stationary combinatorial optimization problem which consists in finding a sequences of movements for a set of elevators in a particular building in order to minimize the Users Average Waiting Time. It is non-stationary since the search space changes as the algorithm is processing toward the optimum. Typical events in this kind of environment, e.g., arrival of a new user or an elevator broken produce changes that modify the problem instance as can be found in many non-stationary problems. The study of MEP is particularly interesting when considering some similar problems of the real world, e.g., industrial processes involving sensors to detect changes in the environment. These changes force a continuous adaptation of the used algorithms for solving non-stationary problems.

In this work we present an ACO model for the Single Elevator Problem (SEP). The ACO model proposed is represented by an ACS, a particular instance of the class of ACO algorithms. A solution for SEP is a sequence of visits that the elevator must follow in order to minimize the Average Waiting Time of the users waiting for service at any floor of the building.

The formulation for SEP is based on the work of Torres [8] which applies an evolutionary approach and considers a static configuration when designing the simulation model of the elevator.

2 The Multiple Elevators Problem

The general formulation of the MEP can be stated as follows: a system with multiple elevators consists of m elevators in a building of n floors where the users, i.e., people that need to travel from one floor to any other of the building, call for the elevators at different times. An elevator call can be generated at any floor of the building as well as from the elevator itself, e.g., go to a particular floor, open the door, stop the elevator, etc. Therefore, each elevator should execute an efficient sequence of movements (of length k) that minimizes, for instance, the users average waiting time, the users average traveling time, or the number of users transported during a given elapsed time.

Clearly, this is a dynamic problem, since after a period of time different changes in the environment could be observed: number of user calls, number of working elevators, changes on the priorities for visiting certain floors, etc. Therefore, some changes should be applied in the sequence of visits to each floor performed by each elevator.

Previous works on MEP have focused, for instance, on the construction of the controller itself, different parameters for an elevator system, and re-optimization. For example, the construction of controllers itself, Ho et al. [5] propose a combination of Petri Nets with Neural Networks to learn the best possible scheduling policy, Kojima et al. [12] apply ADN-Computation to minimize the waiting times. More recently, some evolutionary approaches have been proposed, e.g., Markon et al. [2] developed a Genetic Network Programming to evolve a controler for a multiple elevator system. Other approaches apply fuzzy controllers, expert systems, neural networks and several combinations of them [1, 4, 5, 6]. With respect to the techniques towards to the optimization of certain parameters for a controller, it is assumed that a control unit takes into account a number of parameter values to make the more appropriate decision about which elevator should be assigned when the elevators are called. Accordingly, the objective in this context is determine the best parameter values in order to optimize a number of criteria, e.g., minimization of waiting times, reduction of the crowding factor, minimization of the riding time, etc. In this direction, Fujino et al. [11, 3] show the application of genetic algorithms to find the best parameter setting used in the controller for a multiple elevator system.

In the re-optimization approach, Friese et al. [10] shows a set of re-optimization algorithms for a constrained elevator system where two models, one stationary and other non-stationary, are tested under different assignation and scheduling policies.

2.1 The Single Elevator Problem

The Single Elevator Problem (SEP) is a subclass of the multiple elevators problem. In this case, we have a building with n floors and only one elevator ($m = 1$), where the objective is to minimize the users average waiting time. The elevator in the system must follow a sequence of moves. On each visited floor, the elevator should process as many as possible requirements while minimizing the users waiting time.

In this work we adopt the model for a single elevator problem proposed in Torres [8]: (i) the sequence of visits involves a sequence a integer number representing one of the possible floors of the building, (ii) the elevator stop for service in each floor in which exists a user that made a call to the same elevator destination floor, (iii) a static configuration is considered for the calls to the elevator. In addition, only one temporal list of calls exists in order to produce the respective sequence of visits to the building floors. The last component of the model is (iv), the objective function to evaluate the sequence of visits proposed.

a. Problem Formulation

In the SEP, the elevator must visit the different building floors by following a sequence of visits $s =< p_1, p_2, \cdots, p_k >$ of length k which can be considered as a list of integer numbers representing one of the n possible floors where two consecutive numbers must be different (avoid to go to the same current floor in the next visit). The first number of the sequence, p_1, correspond to the initial floor, i.e., the floor where the elevator will start to execute the sequence of visits given by a particular solution. During the trip from on floor to the next one as indicated in the sequence, the elevator will stop in all intermediate floors to service those users going to the destination floor as the elevator capacity is under the maximum allowed (C_{max}).

For example, Figure 1 shows a possible sequence of visits of length $k = 8$. The elevator starts its ride from the second floor, then execute the first move towards to the third floor. Later it continues to the 5^{th} and 6^{th} floors until completing the whole trip by making $k-1 = 7$ moves to finally arrive to the 10^{th} floor.

1	2	3	4	5	6	7	8
2	3	5	6	4	3	6	10

Figure 1: A sequence of visits of length $k = 8$, i.e., $k - 1 = 7$ moves.

To find the sequence of visits which minimizes the users average waiting time, the following information about the problem is used: (i) $\mu_i(s)$, the accumulated users waiting time at floor i; (ii) $\mu_{ij}(s)$, the average waiting time of user j at floor j, and (iii) γ_i, the number of users making a call on floor i.

Then, the problem formulation is equivalent to find the sequence which minimizes the following objective function:

$$f(s) = \frac{1}{n} \sum_{i=1}^{n} \mu_i(s) \qquad (1)$$

where:

$$\mu_i(s) = \frac{1}{\gamma_i} \sum_{j=1}^{\gamma_i} \mu_{ij}(s) \qquad (2)$$

b. Instances of the SEP as a combinatorial problem

As a combinatorial optimization problem, the SEP can be defined as:

$$\pi = (S, f, \Omega) \qquad (3)$$

where S is the set of all candidates solutions, i.e., possible sequences of visits of length k; f is the objective function which assigns a value $f(s)$ to each solution $s \in S$, and finally, Ω, the set of the problem constrains. For the SEP, the only constraint consists in not allowing the same floor number in contiguous positions of the sequence. The maximum capacity of the elevator (C_{max}) is not part of the problem constraints since the sequence remains valid when C_{max} is reached during the simulation of the sequence.

The candidate solutions $\tilde{S} \subseteq S$ satisfying the problem constraints Ω are called *feasible solutions*. Our aim is to find a global optimal solution $s^* \in \tilde{S}$.

2.2 An ACO Model for the SEP

In an ACO algorithm the artificial ants build solutions incrementally by adding the respective problem components one by one. Thus, the SEP problem can be solved by an ACO algorithm, according to the algorithm proposal (Dorigo et al. [7]) if it can be conveniently represented in the way that the ants can execute a constructive procedure on a construction graph to obtain a feasible solution to the problem at hand.

The SEP formulated as $\pi = (S, f, \Omega)$ is consequently transformed to a problem that can be characterized by the following items:

- A set of components $P = \{p_1, p_2, ..., p_n\}$, i.e., the building floors, where n is the number of problem components.
- The possible problem states are defined in terms of finite sequences $x = < a, p_j, \ldots, p_h, \ldots, p_q >$ on elements from P. The set of all possible states is denoted as χ. The maximum length of a particular sequence is bounded by a positive constant $k < \infty$, where k is the length of the sequence representing a solution to the problem;
- The set of candidate solution is $S \subset \chi$.
- The set of feasible states $\widetilde{\chi}$, with $\widetilde{\chi} \subseteq \chi$, the set of sequences x satisfying the Ω constraints. For the SEP, these are the solutions of length k without contiguous positions having the same floor number.
- A non empty set of optimal solutions $S^* \subseteq S$, i.e, the set of sequences with the minimum objective value.
- A cost $g(s)$ for each candidate solution $s \in S$. For the SEP, $g(s) \equiv f(s), \forall s \in \widetilde{S}$.

According to the above formulation for the SEP, the artificial ants should build a solution by walking the a construction graph G defined in the following.

a. Definition of the Construction Graph

The graph $G = (P, L)$ represents the construction graph used by the artificial ant to build solutions to the SEP, where P is the set of nodes, i.e., the building floors (as defined in Section 2.2) and L is the set of edges representing the existing connections between the nodes. A direct connection between two nodes represents the possibility of the elevator to move from one floors towards to another one. From the perspective of artificial ants' behavior, this means that a particular ant can walk from one node to another one in order to incrementally build a solution. However, a solution for the SEP is not a permutation of integers as in the Traveling Salesperson Problem (TSP). Instead, a solution can be represented as an integer vector since one ant can visit the same floor at different times (i.e., during the elevator trip). To cope with this situation, we propose a partially connected construction graph with $n \times (k-1)$ nodes labeled by (*floor number, move number*) which indicates that the elevator will visit the floor *floor number* at the step *move number*, and a set of edges representing the possible moves between two different floors.

Figure 2, shows the proposed partially connected graph where:

- $a \in \{1 \ldots n\}$ is the initial floor which can be reached by 0 moves.
- $P_k \in \{1, \ldots, n\} - \{a\}$ is any floor different from floor a,

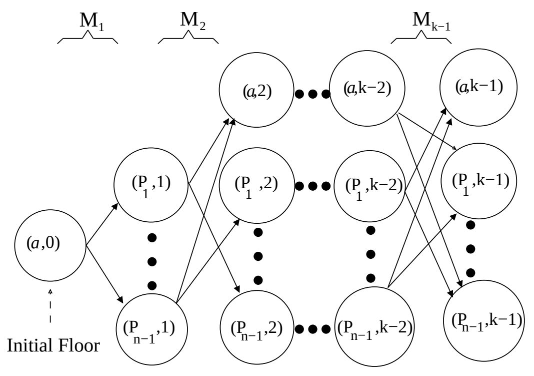

Figure 2: Partially connected graph.

- each "column" of nodes corresponds to the floors that can be visited by making the same number of moves, i.e., M_1, M_2,...,M_{k-1}. It can be observed that there is no direct connection between two nodes where *"floor number"* is the same.

If the sequence of floors ontained is of length k, then the number of moves is $k-1$. For example (see Figure 2), a cannot be chosen at move M_1 since $(a,1)$ is not reachable from $(a,0)$. Consequently only $n-1$ nodes can be chosen as the first move. Similarly, if $P=\{1,2,3,4,5\}$, $k=4$, $n=5$, $a=3$, $P_1=1$, $P_2=2$, $P_3=4$, $P_4=5$, then the possible sequences of length k can be seen as the path connecting the respective nodes that match with *floor number* (Figure 3).

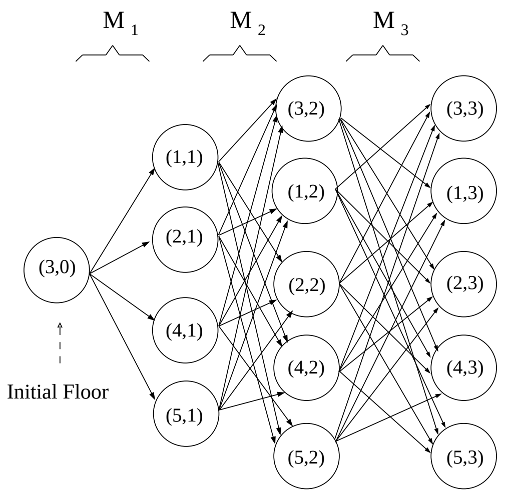

Figure 3: Partially connected graph for $P=\{1,2,3,4,5\}$, $k=4$, $a=3$.

Before presenting the specific ACO algorithm for SEP, it is important to remark that some components of the generic ACO model correspond to the Ant Colony System, which is indeed the ant algorithm to be considered here for implementation.

b. Pheromone trail representation (τ structure)

The pheromone matrix τ, as described by Dorigo et al. [7], is a data structure maintaining a set of real values that represent the accumulated amount of pheromone trail that the ants deposit during the walking. According the construction graph described above, the pheromone structure τ for the SEP is represented by a set of $k-1$ substructures $\tau_{M_1}\ldots\tau_{M_{k-1}}$, each of one corresponding respectively to each possible elevator move when following a particular sequence. In the case of the first move, the dimension of τ_{M_1} is $n-1$ because are only considered the moves from the initial floor a to any other floor. For the remaining moves, the dimension of each pheromone structure is τ_{M_j} is $n\times n$.

c. Pheromone update

The Ant Colony System (ACS) is an improved version of the Ant System (AS). The ACS aims at exploiting in a different way the information about the search experience in order to efficiently guide the exploration of the search space. There exist two mechanism to do that. The first one uses an elitist strategy to update the pheromone trail, while the second one, adopts an alternative selection method applied in the process of a solution construction. With respect to the pheromone update, two different rules are applied: *local* and *global* update rules (Eq. (4) and (5)).

Local update rule: when an ant h selects floor j at the move x^{th} and the elevator is at floor i, the pheromone levels at $\tau_{M_x}[i,j]$ are modified by adding a constant value τ_0 (a parameter of the algorithm).

Global update rule: after each ant had built a solution, the best one is selected (i.e., an ant h^* which represents the solution with the minimum objective value). The global update on each τ_{M_x} is carried out by taking into account $\Delta\tau_{M_x}[i,j]^{h^*}$ value as follows:

$$\Delta\tau_{M_x}[i,j]^{h^*} = \begin{cases} \frac{TME}{TT} & \text{if ant } h^* \text{ uses edge } ((i,x-1),(j,x)) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Then, the pheromone matrix τ_{M_x} is modified according to:

$$\tau_{M_x}[i,j] = \rho\times\tau_{M_x}[i,j]+\Delta\tau_{M_x}[i,j]^{h^*} \quad (5)$$

where,

- $\rho\in[0,1]$ is the parameter that controls the reduction of the pheromone trail (*evaporation rate*),
- $\Delta\tau_{M_x}[i,j]^{h^*}$ the amount of change according to the solution given by ant h^*,

- *TME* is the users average waiting time according to the solution given by ant h^*.
- *TT* is the total service time of the elevator.

d. Floor selection rule.

The floor to be visited by ant h in the next move is obtained by applying a *pseudo-random-proportional rule*. If ant h is situated at floor i, then the probability of choosing floor $j \in \widetilde{\chi}$ as the next move is:

with $rnd < q_0$

$$j = \text{arg max}_{j \in \widetilde{\chi}} \{\tau_{M_x}[i,j][\eta_{ij}]^\beta\} \quad (6)$$

where rnd is a random number from interval $[0,1]$. Otherwise, the next component is chosen according to the following equation:

$$p^h_{ijx} = \frac{\tau_{M_x}[i,j][\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}^h_i} \tau_{M_x}[i,l][\eta_{il}]^\beta} \quad (7)$$

where $\mathcal{N}^h_i = P - \{i\}$ represents the set of candidates floors, η_{ij} is the heuristic information and β is the parameter determining the influence of the heuristic information. In this model we propose the following heuristic information with respect to floor j: (a) the distance between floor i and j, (b) the queue length of users, (c) the average waiting time, and (d) the number of users currently in the elevator going to floor j. Therefore, we define:

$$\eta_{ij} = \alpha_1 \left(\frac{1}{d_{ij}}\right) + \alpha_2 Q_j + \alpha_3 AQ_j + \alpha_4 R_j \quad (8)$$

where

(a) d_{ij} is the distance from floor i to j.

(b) $Q_j = \frac{\text{\#users waiting at floor } j}{\text{Total \#users waiting in the whole building}}$

(c) $AQ_j = \frac{\text{Average waiting time at floor } j}{\text{Total average waiting time}}$

(d) $R_j = \frac{\text{\#calls at floor} j}{\text{Total \#calls in the whole building}}$

According to Eq. (8), the influence of each term in the heuristic value is determined by the weights $\alpha_1 \ldots \alpha_4$. For example, when $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$ all the heuristic values have the same relative importance, i.e., the more likely floor is that which is located within a short distance, with the large number of user waiting for the elevator, waiting for a longer time, and having the larger number of users leaving at that floor.

e. An ACO algorithm of the SEP

In this section we present the algorithm, called *ACS-elevator* (Algorithm 1), which is an ACS based on the model proposed above. The ACS, as mentioned before, increase the exploitation of the information collected by the colony in order to efficiently guide the exploration in the search space. To do that, the mechanisms involve the use of an elitist strategy for updating the pheromone trail and the application on the *pseudo-random proportional rule* described in Eqs. (6) and (7).

When $q_0 = 1$, the ACS is highly greedy. In this case, the more promising floor are those the have the maximum value obtained as a combination of high levels of pheromone trial and large heuristic values. On the other hand, when $q_0 = 0$, the ACS adopts the behavior of the classical Ant System. Therefore, the value for parameter q_0 determines the tradeoff between a greedy and probabilistic components selection. In addition, the ACS applies two rules for updating the pheromone trail on the respective graph connections: *local* and *global* updating rules as described in item *c* of this section. Algorithm *ACS-elevator* works as follows: at each iteration, an ant h generates a sequence of visits called *seq*. Each component of *seq* is obtained through *select_floor()* function, except the first component in the sequence (*seq[1]*) which corresponds to the initial floor. After selecting the next floor, must be simulated the elevator trip to the selected floor and considering all the leaving and incoming users to the elevator. This action is represented by calling to *move_elevator()*

```
proc $ACS-elevator$
  begin
    $initialize()$;
    for (t=1 ... T) do
      for (h=1 ... m) do
        $seq[1] = initial\_floor$;
        for (i=2 ... k) do
          $seq[i] = select\_floor()$;
          $move\_elevator()$;
          $Update\_Times()$;
          $Local\_Update()$;
        od
        if ($f(seq) < f(Opt\_seq)$)
          $Opt\_seq = seq$;
        fi
      od
      $Global\_Update()$;
    od
  end
end
```

Algorithm 1: ACS for the SEP.

which could modify at the involved floors the respective queue states, number of served users, and waiting times. Consequently, it is necessary to apply a special function called *Update_Times()* to handle that. At the same time, the best sequence of the whole search process is maintained in *Opt_Seq*. Other components of the ACS are: *initialize()*, handles the input parameters and data structures initialization, and functions *Local_Update()* and *Global_Update()* which are in charge of the pheromone trial updating as describe before.

3 Experimental study

The experimental study presented here aims at finding an acceptable combination of parameter values for the *ACS-elevator* in order to obtain a sequence of visits of length k which minimizes the users average waiting time μ. To do that, we considered a static configuration of the SEP, i.e., a fixed randomly generated sequence of elevator calls are used during the whole algorithm execution. Let $L = \langle (t^1, p_c^1, p_d^1), \ldots, (t^l, p_c^l, p_d^l) \rangle$ be the sequence of calls of length l where: t^i is the time at which the call occurs satisfying $\forall i = 1 \ldots l-1, t^i < t^{i+1}$ (p_c^i is the calling floor and p_d^i is the destination floor). We consider two different instances of the SEP in this work. The time, in seconds, between calls is given by a uniform distribution from interval $[10, 60]$ (Instance 1) and $[10, 300]$ (Instance 2). For both instances we considered a sequence L of calls of length $l = 1000$.

The parameters considered in the experimental study are: a, initial position of the elevator; C_{max}, maximum capacity of the elevator, n, number of floors in the building; k, length of the sequence of visits (or solution); T, maximum number of the algorithm iteration, ta trip elevator time between two consecutive floors; and the other *ACS-elevator* parameters, i.e., q_0, β, τ_0, ρ, α_1, α_2, α_3, and α_4 (Eqs. (4),(5), and (6)).

For each parameter setting, given below, 30 runs were performed by using different random seeds. Thus the values displayed here correspond to average waiting times $f^{(i)}(s^*) = \frac{1}{30}\sum_{j=1}^{30} f_j(s^*)$, for $i = 1, 2$ (the two instances considered).

We performed two experimental studies with the following common parameter setting: $T = 100$, $m = 10$, $a = 0$, $C_{max} = 8$, $n = 10$, $k = 1000$, $ta = 2$. For the first experimental study (ExpStd$_1$) we took into account the parameter values reported in Molina et al. [9], i.e., $q_0 = 0.8$, $\beta = 0.5$, $\tau_0 = 0.8$, $\rho = 0.5$, $\alpha_1 = 0.05$, $\alpha_2 = 0.7$, $\alpha_3 = 0.5$, $\alpha_4 = 0.3$. However, some parameters were varied to observe the impact on the solutions quality. Those parameters are: q_0, β, and τ_0 (see Table 1). The results for ExpStd$_1$are showed in Table 2.

It can be observed from ExpStd$_1$ that the best performance was achieved in experiment E_{12} where all

ExpStd$_1$	q_0	β	τ_0
E_{11}	0.8	0.2	0.8
E_{12}	**0.8**	**0.8**	**0.8**
E_{13}	0.2	0.5	0.8
E_{14}	0.7	0.5	0.8
E_{15}	0.8	0.5	0.05
E_{16}	0.8	0.5	0.5

Table 1: Parameters setting for the first experimental study (ExpStd$_1$). E_{1j} stands for experiment j of ExpStd$_1$.

parameters considered are set to a large value. That is, *ACS-elevator* involves a highly greedy components selection, a large importance to the heuristic information, and adding an important amount of pheromone trail when applying the local update rule. In addition, a one-way statistical ANOVA test (at a confidence level of 95%) indicated that all mean values differed statistically. More precisely, experiment E_{12} achieved the best performance giving a mean value that differed statistically from the remaining data experiments according to the Tukey's method.

Experiments	$f^{(1)}$	$f^{(2)}$
E_{11}	147.93	601.92
E_{12}	**118.81**	**471.16**
E_{13}	179.43	729.88
E_{14}	133.95	537.34
E_{15}	125.78	501.35
E_{16}	127.09	508.14

Table 2: ExpStd$_1$: average waiting times $f^{(1)}$ and $f^{(2)}$ for the respective two considered instances.

In the second experimental study (ExpStd$_2$) we performed 3 additional experiments ($E_{2j}, j = 1, 2, 3$) with the same parameter setting as in E_{12} except the setting of $\tau_0 = 0.05, 0.2,$ and 0.5 (see Table 3) to see the influence of this parameter. The respective results for ExpStd$_2$ are showed in Table 4. Although we can observe a lower value for the objective function when comparing E_{21} against E_{12}, they did not differ statistically according the the ANOVA test. In fact, the values for the second experimental study, i.e., E_{2j}, for $j = 1, 2, 3$, and the values from E_{12} (the best behavior from experiment one) were statistically insignificant. Clearly, these results showed that q_0 and β are among the most important parameters regarding the performance of our algorithm *ACS-elevator*.

ExpStd$_2$	q_0	β	τ_0
E_{21}	**0.8**	**0.8**	**0.05**
E_{22}	0.8	0.8	0.2
E_{23}	0.8	0.8	0.5

Table 3: ExpStd$_2$ is based on E_{12} with three different values for τ_0.

Experiments	$f^{(1)}$	$f^{(2)}$
E_{21}	**117.66**	**467.22**
E_{22}	118.00	468.35
E_{23}	118.65	470.96

Table 4: ExpStd$_2$: average waiting times $f^{(1)}$ and $f^{(2)}$ for the respective two considered instances.

4 Conclusions

In this work we presented an ACO model and a possible ACO algorithm, the *ACS-elevator*, for solving the Single Elevator Problem. The ACO model involves a partially connected construction graph used by the artificial ants to construct the solutions for the SEP. For the experimental study we considered two instances of SEP (a static version) in order to study the influence of some parameters of the proposed ACO algorithm which can be considered an interesting starting point to go further in the research of a more complex version of SEP and the possibility of the application of ACO algorithms for the general version of the SEP, i.e., the Multiple Elevators Problem.

ACKNOWLEDGMENTS

The first and second author acknowledge supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency for Promotion of Science and Technology).

The third author is partially supported by the Spanish Ministry of Education and European FEDER under contract TIN2005-08818-C04-01 (the OPLINK project,http://oplink.lcc.uma.es).

References

[1] Zhu Dewen, Jiang Li, Zhou Yuwen, Shan Guanghui, and He Kai. Modern elevator group supervisory control systems and neural networks technique. In *1997 IEEE International Conference on Intelligent Processing Systems, ICIPS '97.*, pages 528–532, 28-31 Oct. 1997.

[2] T. Eguchi, K. Hirasawa, Jinglu Hu, and S. Markon. Elevator group supervisory control systems using genetic network programming. In *Congress on Evolutionary Computation, 2004. CEC2004*, volume 2, pages 1661–1667, 19-23, June 2004.

[3] A. Fujino, T. Tobita, K. Segawa, K. Yoneda, and A. Togawa. An elevator group control system with floor-attribute control method and system optimization using genetic algorithms. In *IEEE Transactions on Industrial Electronics*, volume 44, pages 546–552, Aug. 1997.

[4] R. Gudwin, F. Gomide, and M. Andrade Netto. A fuzzy elevator group controller with linear context adaptation. In *IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence*, pages 481–486, 4-9 May 1998.

[5] Ming Ho and B. Robertson. Elevator group supervisory control using fuzzy logic. volume 2, pages 825–828, 25-28 Sept.

[6] N. Imasaki, S. Kubo, S. Nakai, T. Yoshitsugu, Jun-Ichi Kiji, and T. Endo. Elevator group control system tuned by a fuzzy neural network applied method. In *Proceedings of 1995 IEEE International Conference on Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium*, volume 4, pages 1735–1740, 20-24 March 1995.

[7] Dorigo Marco and Stützle Thomas. *Ant Colony Optimization.* Mit Press, 2004.

[8] Rafael Torres Márquez. Algoritmos evolutivos distribuidos en entornos dinámicos. Master's thesis, Departamento Lenguajes y Ciencias de la Computación - Universidad de Málaga, 2000.

[9] Silvia M. Molina, M. Guillermo Leguizamón, and Enrique Alba. Un Modelo ACO para una Versión No Estacionaria del Problema del Ascensor Único. In *XII Congreso Argentino de Ciencias de la Computación (CACiC)*, San Luis, Argentina, 2006. Universidad Nacional de San Luis.

[10] J. Rambaud and P. Friese. On line-optimization of a multi-elevator transport system with reoptimization algorithms based on set-partitioning models. Technical report, ZIB Report 05-03, 2003.

[11] T. Tobita, A. Fujino, K. Segawa, K. Yoneda, and Y. Ichikawa. A parameter tuning method using genetic algorithms for an elevator group control system. In *Proceedings of the 1996 IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, pages 823–828, 5-10 Aug. 1996.

[12] J. Watada, S. Kojima, S. Ueda, and O. Ono. DNA computing approach to optimal decision problems. In *International Joint Conference on Neural Networks - IEEE Conference on Fuzzy Systems*, pages 25–29, Budapest, Hungary, July 2004.