Available online at www.sciencedirect.com
ScienceDirect
Engineering Applications of Artificial Intelligence 21 (2008) 343–352
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

Rescheduling and optimization of logistic processes using GA and ACO

C.A. Silva[a,1], J.M.C. Sousa[a,*,1], T.A. Runkler[b]

[a]Technical University of Lisbon, Instituto Superior Tećnico, Department of Mechanical Engineering, CSI-IDMEC, 1049-001 Lisbon, Portugal
[b]Siemens AG, Corporate Technology, Information and Communications, Learning Systems Department, 81730 Munich, Germany

Received 12 February 2007; accepted 21 August 2007
Available online 13 November 2007

Abstract

This paper presents a comparative study of genetic algorithms (GA) and ant colony optimization (ACO) applied the online re-optimization of a logistic scheduling problem. This study starts with a literature review of the GA and ACO performance for different benchmark problems. Then, the algorithms are compared on two simulation scenarios: a static and a dynamic environment, where orders are canceled during the scheduling process. In a static optimization environment, both methods perform equally well, but the GA are faster. However, in a dynamic optimization environment, the GA cannot cope with the disturbances unless they re-optimize the whole problem again. On the contrary, the ant colonies are able to find new optimization solutions without re-optimizing the problem, through the inspection of the pheromone matrix. Thus, it can be concluded that the extra time required by the ACO during the optimization process provides information that can be useful to deal with disturbances.

Keywords: Rescheduling; Logistic system; Genetic algorithms; Ant colony optimization; Optimization

1. Introduction

In the last decade, leading companies all over the world, across all business areas, adopted the *supply chain* (Barbuceanu and Fox, 1996) organization methodology to face the increasing competitiveness of the markets. One of the most challenging problems in supply chain management is the optimization of the logistic subprocess (Swaminathan et al., 1998). This is a very complex scheduling problem that deals with the planning, handling and control of the storage of goods between the manufacturing point and the consumption point. However, a question that arises throughout the development of every new scheduling application is "What optimization method should be used?". Is there a method that guarantees a higher chance of finding the global optimum? Which is the easiest method to program and tune?

Over the last decade, *meta-heuristics* have replaced the analytical methods, exhaustive search methods or local search heuristics in the optimization of scheduling problems (Jain and Meeran, 1999). These type of methods, which includes *genetic algorithms* (GA) (Holland, 1975), *simulated annealing* (Kirkpatrick et al., 1983), *tabu search* (Glover and Laguna, 1997) or *ant colony optimization* (ACO) (Dorigo et al., 1996), uses an algorithm to guide simple heuristics in the search of global optima. Nowadays, meta-heuristics are considered to be very powerful scheduling techniques (Jain and Meeran, 1999) and therefore suitable to be applied to the optimization of logistic processes (Silva et al., 2005).

GA are the most studied and applied meta-heuristic in the optimization field (Michalewicz, 1999; Michalewicz and Fogel, 2002), because GA are very easy to implement in all sort of problems, and usually guarantee good solutions, whatever the type of solution space. Although is not difficult to find for specific problems methods that outperform GA, it is almost impossible to find any other method

*Corresponding author.
E-mail addresses: csilva@dem.ist.utl.pt (C.A. Silva), jmsousa@ist.utl.pt (J.M.C. Sousa).
[1]The work is supported by the project POCI/EME/59191/2004, co-sponsored by FEDER, Programa Operacional Ciência e Inovação 2010, FCT, Portugal.

0952-1976/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2007.08.006

that performs as well as GA in so many different types of optimization problems.

ACO is a relatively new optimization technique that has been gaining followers in the past decade, since it is especially suited for problems with dynamic behavior (Dorigo and Stützle, 2004). Furthermore, the intrinsic agent nature makes it also suitable for parallel and multi-agent applications, like supply chain management (Parunak, 1997). However, the application of ACO is restricted to optimization problems that can be described by graphs (Dorigo and Stützle, 2004).

For these reasons, GA and ACO are both very appealing optimization methods to design scheduling applications such as the logistic scheduling problem, because they are simple to implement and efficient in finding optimal solutions. But which of the algorithms should be used? At the end, the optimization method ends up to be the technique that one knows better. Or, if there is the time and the expertise knowledge, it is possible to test more than one method and choose the one that performs better. But the question remains: are there any elements that might tell us before hand which algorithm is better for a specific application?

To answer this question, it would help to have a systematic comparison between the two methods for different problems. Although the literature in optimization is very rich in terms of algorithm implementations, only few authors present comparison results with other algorithms. It is possible, however, to compare algorithms presented in different works when the test instances are the same. Nevertheless, the results often lack accuracy indicators based on statistical analysis; the analyzes do not consider the same measurements or number of trials; different local heuristics are used with each method; finally, the computational effort, when measured, is presented on a pure time basis, which is not easily comparable from one computer to another.

This paper describes a systematic comparison between GA and ACO to assist the decision process about what algorithm should be used to reschedule a logistic system. The comparison includes a literature survey on several benchmark optimization problems, in order to make valid assessments for optimization problems in general, and a study on the performance of each algorithm in a logistic scheduling problem. This paper describes further the importance of the information generated during the optimization process and how this information can be used in dynamic environments. Nowadays, the optimization of real systems is no longer a static problem, where all the parameters remain constant during the optimization process (Sarmiento and Nagi, 1999). In the logistic system case, the changes on the optimization environment may be caused by cancellation of orders from the clients. Therefore, the optimization method of a logistic system has to cope with these issues.

The paper is organized as follows. Section 2 presents a literature survey on optimization results for different benchmark problems with GA and ACO. The logistic scheduling problem is introduced in Section 3, which also describes the GA and ACO implementations. The comparison between the methods for static logistic problem and for dynamic logistic problems is in Section 5. Section 6 concludes the paper and indicates the future research steps.

2. Literature survey

GA have been intensively applied to different types of optimization problems in the last 15 years and it is easy to find implementation results for all sort of benchmark problems. In terms of ACO, because it is a relatively new method, it has been applied to less benchmark problems.

Table 1 summarizes the comparison results for several instances of well known and studied benchmark optimization problems: the *traveling salesman problem* (TSP), the *quadratic assignment problem* (QAP), the *vehicle routing problem* (VRP) and the *job shop problem* (JSP).

The performance results are usually reported in terms of relative error e to the *best known solution* (BKS) of the best

Table 1
Comparison between GA and ACO performance for different instances of different benchmark problems

Instance	GA			ACO		
	min(e) (%)	$\mu(e)$ (%)	t (s)	min(e) (%)	$\mu(e)$ (%)	t (s)
Traveling salesman problem						
lin318	0	0	15	0	0	94.2
pcb442	0	0	33	0	0.26	308.9
att532	0	0	97	0	0.08	387.3
rat783	0	0	376	0	0.10	965.2
pcb1173	0	0.001	1177	0	0.11	3219.5
Asymmetric traveling salesman problem						
ry48p	0	0.20	72	0	0	2.3
ft70	0	0	111	0	0	37.2
kro124p	0	0	35	0	0	7.3
ftv170	0	0.26	100	0	0	56.2
Quadratic assignment problem						
tai20a	0	0.268		0	0.191	
tai80a	0	0.796		0	0.836	
nug30	0	0.007		0	0.013	
sko42	0	0.003		0	0.032	
bur26a	0	0.043		0	0.006	
tai20b	0	0		0	0	
tai80b	0	0.829		0	0.591	
Vehicle routing problem with time windows						
R1	3.6	12.0	118	1.1	12.4	210
RC1	4.5	11.5	95	1.9	11.7	210
R2	7.6	3.0	105	2.4	2.7	210
RC2	5.2	3.3	44	0.7	3.3	210
Job shop problem						
FT10	0		292	0	0.9	93.49
FT20	0		204	0	0.3	88.28
LA29	3.3		1350	0.3	0.9	1084.98
LA38	1.9		1859	2.5	3.3	928.02
LA40	1.5		2185	0.5	1.0	1031.1

trial min(e) and the mean relative error of different trials $\mu(e)$. When possible, Table 1 also indicates the time required to find the optimization result t, as an indicator of the computational effort of each algorithm. This indicator is not accurate: the methods are implemented on different machines, using different programming languages and therefore require different computational power. Some algorithms have a fixed number of iterations, others have a stop criteria. Further, many algorithms have less operations per iteration, but need more iterations to find the optimum. Therefore, to use the computational time as a fairly good indicator of computational effort, it is necessary that the time measurements for two different algorithms consider the same number of total iterations, and the experiences are done using the same machine and the same type of programming.

2.1. Traveling salesman problem

To the best of our knowledge, the latest comparison between GA and ACO for the symmetric TSP problem is the one presented in Tsai et al. (2003), a work that proposes new genetic operators and compares them with several different heuristics. The ACO results used for the comparison are taken from Stützle and Dorigo (1999b). The results show that both methods are very good, with the biggest average error being lower than 0.3% for the ACO and a null average error in all instances except one for the GA. It can also be observed that the computational effort for the ACO is one order of magnitude higher than for the GA, as expected, since the intrinsic number of operations of GA is $\mathcal{O}(n^2)$ and ACO is $\mathcal{O}(n^3)$, where n is the number of cities in the problem.

For asymmetric problems, it is necessary to consider the fact that the travel cost ω between city i and city j depends on the traveling direction, i.e. $\omega_{ij} \neq \omega_{ji}$, and the problem complexity increases. The first noticeable difference is the fact that there are much less GA implementations for asymmetric TSP than for the symmetric case. This has to do with the fact that the GA implementation for the asymmetric case is not straightforward for many chromosome encodings. It is difficult to find a suitable representation of a GA solution that: (1) represents solutions in graphs where the weights in the arcs depend on the travel sequence or when the weights change during the optimization problem; (2) allows at the same time an easy implementation of the genetic operations. The presented results concern genetic local search (Merz and Freisleben, 1997), a special type of GA which makes equal use of crossover and mutation operators. Recently, some GA results were presented in Choi et al. (2003), concerning asymmetric problems only. However, the results in this paper are not very clear in terms of average results, and do not show any special improvement when compared to the implementation in Merz and Freisleben (1997). They are not much faster either, and therefore they are not considered here. On the contrary, for the ACO algorithm, the codification is exactly the same when solving symmetric or asymmetric problems. This property arises from the graph searching nature of the algorithm, and therefore, the results consider always both type of instances. They show that the ACO always finds the best solutions for this type of problems, in a short period of time, while the GA fails occasionally to find the BKS, although in general they are also very good. The computational effort shows that for ACO, the time order of magnitude for symmetric or asymmetric problems is the same, while for GA, the asymmetric problems involve much more effort than the symmetric instances of the same size.

The overall comparison results show a clear advantage of ACO to solve asymmetric problems. In the symmetric instances, where the problem can be represented by a simple permutation, the GA performs better and is faster than the ACO.

2.2. Quadratic assignment problem

For the QAP, Stützle and Dorigo (1999a) present a direct comparison between the ACO and the GA implementation of Fleurent and Ferland (1994). There is a more recent GA implementation in QAP, presented in Lim et al. (2000), with promising results. However, the used test instances do not allow a direct comparison and therefore those results are not presented.

The QAP problem can be easily depicted in a graph, and the problem solution can also be represented as a simple permutation. In this sense, it is very easy to implement both GA or ACO methods to optimize this problem. The results presented in Table 1 show that also here, both GA and ACO present similar performances, with the biggest average error from the BKS being smaller than 0.9% for the GA. The error of the best know solution of the best trial is not presented. The computational effort is also not explicitly compared, but it is mentioned in Stützle and Dorigo (1999a) that the ACO algorithm is 25% slower than the GA approach.

In conclusion, for the QAP problems, both GA and ACO perform very well, but GA are faster to find the optimum, which confirms the conclusions drawn for the TSP.

2.3. Vehicle routing problem

For the VRP, there is a systematic comparison work by Bräysy (1999, 2001) between GA and other different methodologies for the problem with time windows, such as the ACO implementation proposed in Gambardella et al. (1999). Table 1 presents the error in terms of travelled distance, as well as the total number of used vehicles m.

The results show that in general the ACO approach is better to find the minimum cost distance, while the GA approach deals better with the customer clustering part of the problem, since it uses slightly less vehicles. The computational effort seems to be smaller for the GA approach. Notice however that ACO uses a fixed

computational time, which means that the algorithm probably finds the optimum in less time for most of the instances.

2.4. Job shop problem

For the *Job Shop Scheduling* benchmark problems, there has been a lot of GA implementations, although they are not the best performing algorithm, which has been until very recently considered to be the taboo search technique of Nowicki and Smutnicki (1996). Nevertheless, the GA present very interesting performances in JSP, when implemented in some hybrid form, together with other optimization methods, such as the method proposed by Gonçalves et al. (2005).

There had been no interesting results of ACO implementations for JSP until very recently, when Blum and Sampels (2004) reported for the first time competitive results of ACO. They compared their implementation with the tabu search technique (Nowicki and Smutnicki, 1996) and were able to improve the best known solutions for 15 of the 28 tested instances for the group scheduling problem. For the problems normally used to study scheduling algorithms, including the FT instances (Giffler and Thompson, 1960), the ACO algorithm proved also to be an efficient method and better performing than the hybrid GA approach, as Table 1 shows. For all instances except the *LA38*, the ACO best solution is equal or outperforms the GA solution. Moreover, for instances *LA29* and *LA29*, the ACO mean result is better than the GA best result. Nothing is said about the mean optimization results of the GA approach and therefore nothing can be concluded about the robustness of the GA algorithm. The ACO algorithm is very robust, since the mean results are close to the best result. The computational effort of both approaches has the same order of magnitude.

2.5. Preliminary comparison conclusions

The comparison of GA and ACO based on the literature review can be summarized as follows:

- GA and ACO algorithms present similar performances for different types of optimization problems;
- GA are in general faster than ACO, due to the nature of the algorithms. However, for complex codifications of GA, the computational effort can increase dramatically.
- ACO seem to perform better in path finding problems in disjunctive graphs, while the GA seem to perform better in pure discrete problems that aim the selection of the best combination of items from a broader set.

One issue that remained unanswered was whether the extra time required by the ACO to find a solution can be useful for some applications. The computational effort of the ACO algorithm is $\mathcal{O}(N)^3$, while the GA is $\mathcal{O}(N)^2$ (where N is the number of operations). The difference can be explained from the graph theory point of view, since each individual of the ACO population constructs the solution path node per node, while the GA provide a entire path in one single iteration for each individual. This constructive nature of the ants algorithm is recorded in a pheromone matrix that at the end of the optimization process provides information regarding the whole optimization process. Therefore, if the optimization search space suddenly changes, it might be possible to trace back good optimization solutions for the new search space. This advantage of the ACO algorithm, as a solution constructor algorithm, has already been explored in the optimization of communication networks (Di Caro and Dorigo, 1998), a highly dynamic problem.

3. The logistic scheduling problem

The objective of this paper is to provide sustainable arguments to decide which optimization algorithm should be used to optimize a real-world logistic process at Fujitsu-Siemens Computers (Silva et al., 2005). In this way, this paper considers the simulation of a simplified but realistic model of a logistic system. At each day, the logistic system has an *order list* O of n orders waiting to be delivered. An order $o_j \in O$ with $j = 1, \ldots, n$ is a set of m different types of items, called the components c_i with $i = 1, \ldots, m$, in certain quantities q_{ij}. Therefore, an order can be defined as an m-tuple $o_j = (q_{1j}, \ldots, q_{mj})$. When a new order o_j arrives, it receives two labels: the arrival date or *release date* r_j and the desired delivery date or *due date* d_j, which is the date when the client wishes to receive the order. The components c_i are purchased to external suppliers and collected at the cross-docking center after some time, called the suppliers service, which is the time that the suppliers take to deliver the components. At this point, the decision process starts. Based on the components stock at the cross-docking center and the order list, the logistic system has to assign the components to the orders. Finally, the orders are delivered at the completion date C_j.

Fig. 1 presents a schematic representation of the logistic process. In this example, the logistic system collects $n = 7$ orders, composed of different combinations of 4 types of components. These components are purchased to the suppliers, which are able to deliver only part of the requested quantity of components. The optimization algorithm has to decide which of the orders waiting on the order list will receive the components, based on the desired delivery date d_j and trying to maximize the number of delivered orders. One optimal solution is to deliver orders o_6 and o_7, whose completion date C_j is equal to the desired delivery date d_j.

Assuming that the system does not deliver orders if they are ready before the due date, the difference between the completion date and the due date is called the *tardiness* $T_j = C_j - d_j$. The objective is to match both dates, i.e. to have for all orders $T_j = 0$. However, two disturbances may influence the system: the fact that suppliers service may not

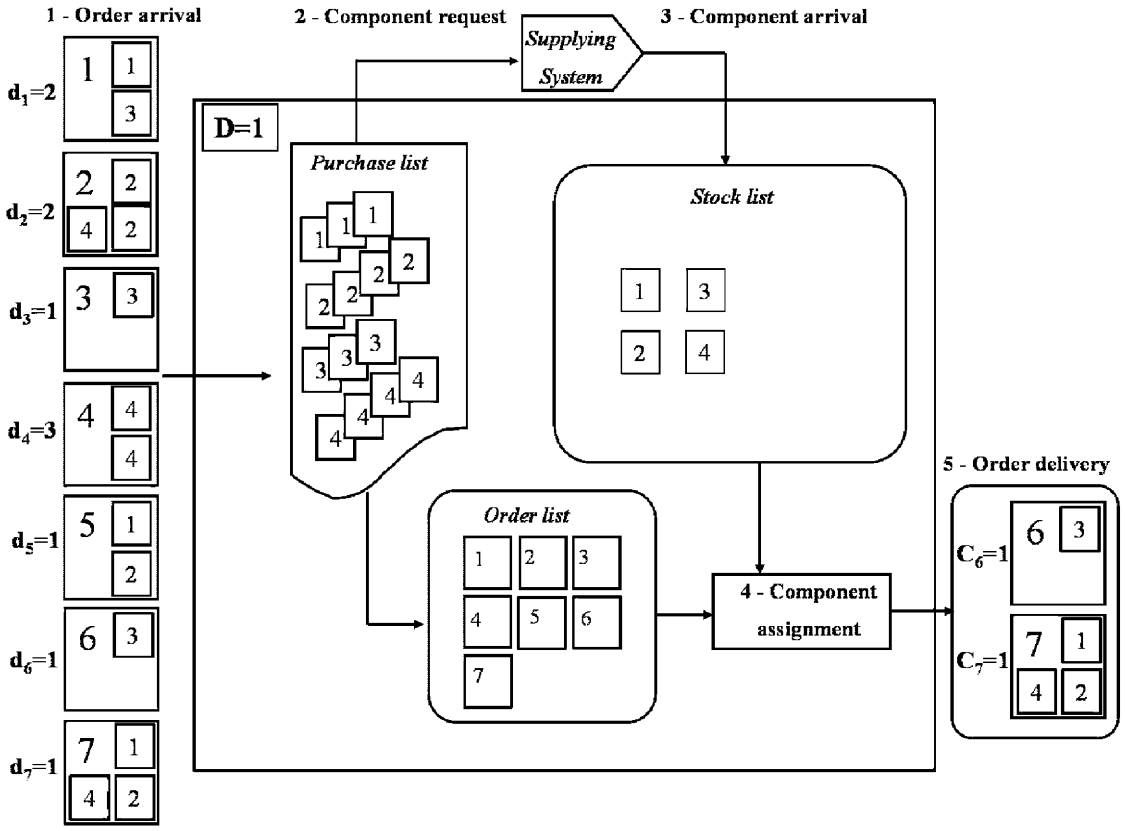

Fig. 1. Logistic system representation, with 7 incoming orders.

be respected and the fact that some clients ask for desired delivery dates not compatible with the supplier services.

In order to define the cost function that best describes the objective of the logistic system, consider the following definitions. Given the set O of orders in the system waiting to be delivered, the subset of orders that are going to be *delivered* is defined as $O_D \subseteq O$. The complementary subset of orders that are *not delivered* and remain in the system is defined as $O_{ND} \subseteq O$, such that $O_D \cup O_{ND} = O$. Consider further that the subset of orders that are delivered at the correct date is defined as $O_D^0 \subseteq O_D$ and the subset of orders that are not delivered and are already delayed is defined as $O_{ND}^d \subseteq O_{ND}$. The optimization objective is to minimize the cost function given by

$$f_L = \frac{\sum_{j \in O} T_j + |O_{ND}^d|}{|O_D^0| + \varepsilon}, \quad (1)$$

where $\sum_{j \in O} T_j$ accounts for the minimization of the tardiness of the total set of orders in the system O; $|O_{ND}^d|$ is the cardinality of the subset O_{ND}^d and refers to the minimization of the number of orders that are not delivered and are already delayed; and finally $|O_D^0| + \varepsilon$ is the cardinality of the subset O_D^0 and accounts for the maximization of the number of orders delivered at the correct date. The ε is a small constant that avoids the infinity value when no orders are delivered at the correct date. This decision step is done once per day, but different solutions for the same daily problem originate different next-day scheduling problems. The supply chain management is a dynamic succession of daily different problems, that are treated independently, even though they are not. Therefore, to evaluate the performance of the supply chain, larger periods of time, such as weeks or months, should be considered.

4. The logistic scheduling methods

This section presents the implementation of GA and ACO to the logistic scheduling problem introduced in the previous section.

4.1. GA implementation

In the GA implementation the chromosome is a binary vector representing the list of orders waiting to be delivered, with size n, with a 0 value if the order is not delivered, and the value 1 if the order is delivered. The population is initialized as random binary strings. The selection of the individuals is done according to the roulette wheel method (Michalewicz, 1999). The eliminated population is replaced by the new offsprings. The new offsprings are generated using the traditional *one-point* crossover, where two parents originate two different offsprings. Finally, the mutation is applied to a small subset of offsprings after the crossover step. The GA implementation

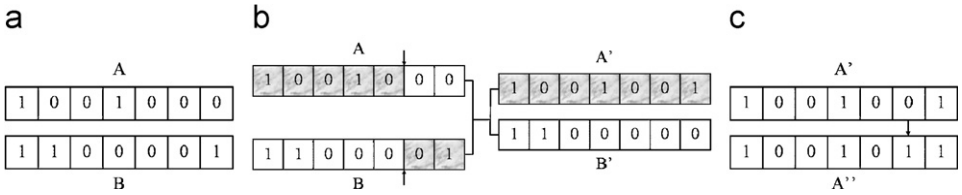

Fig. 2. GA implementation to solve the logistic process. (a) Encoding of two different solutions. (b) One-point crossover between solutions A and B, resulting in two new members A′ and B′. (c) Mutation suffered by offspring A′.

in this paper is *elitist*, since the best individuals within the population are maintained. Fig. 2 represents the operations between two solutions for the example described in Fig. 1 with $n = 7$ orders. The final solution A″ is to deliver the set of orders $\{1, 4, 6, 7\}$.

The fitness function of a solution is given by the objective function of the scheduling problem f, as proposed in (1). Due to the constraints present in our problem, not all the solutions are *feasible*. In the present implementation, if an infeasible solution is created after the genetic operations are applied, the infeasible solution is repaired into a feasible one before the algorithm proceeds to the next iteration. The transformation of an infeasible solution into a feasible one consists of checking if there are enough components in the stock to deliver this order or not. If yes the gene remains 1, the stock is updated and the algorithm proceeds to the next gene. If there are not enough components in the stock, the gene is changed from 1 to 0. The solution depends on the sequence by which the genes are individually checked. Since we do not want orders to be delivered with delay, we use the *first desired first served* heuristic to define the sequence by which the genes are checked, introducing in this way a local heuristic. In the example solution presented in Fig. 2, the feasible solution would be to deliver the set of orders $\{6, 7\}$ (see Fig. 1 to verify the desired deliver dates d_j).

The algorithm runs for $\mathcal{O}(g \times N) \approx \mathcal{O}(N_{\max}^2)$ time, where $N_{\max}$ is the maximum number of iterations allowed and g is the size of the population. The parameters for the GA used in this implementation are: the selection rate is 0.5 and the crossover rate is 1. The mutation rate is 0.1. The population size is $g = 100$.

4.2. ACO implementation

The logistic problem is modeled by a graph, where the nodes represent the orders waiting to be delivered. The role of the ants is to find the minimum cost path connecting the orders that should be delivered. The objective function to be minimized by each ant k, is the one defined in (1), and is denoted by f^k.

One important aspect of the problem is the fact that the number of visited nodes may not be the same from one ant to another (e.g., in TSP the number of nodes to visit is fixed and equal to the number of cities to visit Dorigo et al., 1996). We consider that each ant is traveling with a bag with the available stocks and is distributing the stocks

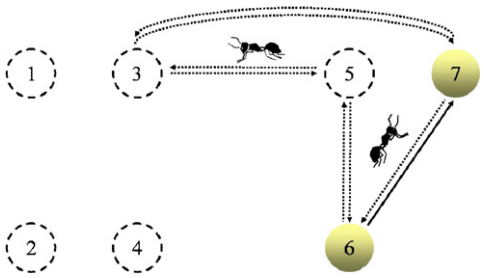

Fig. 3. Graph representing the scheduling problem with 7 orders solved by the ACO. The pheromone trails have different intensities: strong (-), medium (..).

between the orders that it is visiting. It only visits orders which is possible to deliver: e.g., if an order needs 2 components c_i, and the ant only has one ($q_i = 1$), it will not visit that order. In this way, the ACO only builds feasible solutions. When the stock bag is empty or the remaining components are not enough to deliver any missing order, the search for this ant is finished. Therefore, the algorithm will not have a constant number of iterations, since the number of orders visited by each ant is different. Since the path is not closed, the initial starting point for each ant assumes an important role. Fig. 3 represents schematically the optimization graph. The ants have found a good schedule, delivering orders (6, 7) and alternative schedules, such as (6, 5) or (3, 7). Notice that the notation change from set notation {,} in the GA, to vectorial notation (,) because ACO solution also indicates the sequence in which the orders should be scheduled.

In this way, for an ant k in the graph, the probability of choosing the next order to deliver is given by

$$p_{ij}^k(t) = \begin{cases} \dfrac{\tau_{ij}^\alpha \times \eta_{ij}^\beta}{\sum_{j \notin \Gamma}^m \tau_{ij}^\alpha \times \eta_{ij}^\beta} & \text{if } j \notin \Gamma, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where τ_{ij} is the pheromone concentration in the path (i,j), η_{ij} is a *heuristic function* and Γ is a *tabu list*. The pheromone trails τ_{ij} are restricted to the interval $[\tau_{\min}, \tau_{\max}]$, with $\tau_{\min} = 0$ and $\tau_{\max} = 1$. Initially, all the pheromones track are initiated with the value $\tau_{\max}/2$.

The heuristic function η in this case is the tardiness of the order: if an order has already a delayed the ant will feel a stronger attraction to visit it. This function depends only on the order and therefore it concerns the nodes only,

therefore $\eta_{ij} = \eta_j$. Notice that this is again the *first desired first served* heuristic, also used in the GA. However, we define the heuristic function here as an exponential function in the interval $[0, 1]$ where the value 0 is for the order that has the minimum tardiness $T_{\min}$ and 1 is for the most delayed order $T_{\max}$. The objective is that the orders already delayed attract ants much more than the orders not yet delayed:

$$\eta = \frac{e^{(T_j - T_{\min})/(T_{\max} - T_{\min})} - 1}{e - 1}. \quad (3)$$

As described in Silva et al. (2006), the values are restrained to this interval to make a more comprehensive weighting of the weights α and β.

The tabu list Γ is the list of orders already delivered by the ant and also the orders which are not possible to visit due to lack of stocks. The parameters α and β measure the relative importance of trail pheromone (experience) and local heuristic (knowledge), respectively.

Let the *tour* be the route made by one ant until it empties the stock bag and an *iteration* be the set of the tours performed by all the g ants. The update of the pheromone concentration in the trails is done at the end of each iteration and is given by

$$\tau_{ij}(t+1) = \tau_{ij}(t) \times (1 - \rho) + \sum_{k=1}^{g} \Delta\tau_{ij}^{k}, \quad (4)$$

where $\rho \in [0, 1]$ expresses the pheromone evaporation phenomenon, and $\sum_{k=1}^{g} \Delta\tau_{ij}^{k}$ are pheromones deposited in the trails (i,j) followed by all the g ants after a complete tour, which are defined as

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{1}{f_k} & \text{if arc } (i,j) \text{ was used by the } k \text{ ant,} \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

where f_k is the value of an evaluation function for each k ant in a minimization problem. However, it improves the algorithm convergence if only the best ant of each iteration is allowed to update the trail. In this way, the pheromone matrix is only biased by the solution found by the best ant at each iteration. Notice that the time interval taken by the g ants to complete all tours is at most $n \times g$. In every Nth of the $N_{\max}$ maximum number of tours, a new *ant colony* is released. The algorithm runs for $\mathcal{O}(n \times g \times N_{\max}) \approx \mathcal{O}(N^3)$ time.

5. Comparison for the logistic problem

The simulation results consider a logistic problem with an average number of n arriving orders at each day, following a Poisson distribution (Ross, 1996). Each order is a set of 1 up to m different types of components, and has in average $\mu(m)$ different types of components. Each type of component within an order can have a maximum quantity of $q_{\max}$. In this way, a logistic instance problem is defined by the 4-tuple $(n, m, \mu(m), q_{\max})$. This study considers three different instances, describing different sizes an complexity degrees: instances $(5, 5, 3, 10)$, $(10, 10, 2, 30)$ and $(20, 10, 5, 20)$.

The simulation results describe two different scenarios. In the first one, the static scenario, the orders accepted in the logistic system are never canceled or postponed. This situation is advantageous in terms of the logistic system optimization, since it describes a static solution search space. In real-world systems this type of environment is achieved by imposing cancelation fees.

There are however, situations where the logistic system may accept the cancelation or postponement of certain orders, e.g., when the clients are very important. This dynamic environment, the second proposed scenario, increases the complexity of the optimization problem, because the solution search space changes and the optimal solution of a certain search space is likely to be sub-optimal in a different space. Even if the cancelation fee covers the lost of unused stock components, it does not cover the costs of non optimal solutions.

5.1. Static scenario

We present the optimization results considering an optimization time interval of 30 days. Although both algorithms start with the same initial conditions, each algorithm finds a solution for the one day problem that is slightly different from the one found by the other method. In this way, the next day optimization problem will have different initial conditions and the results are no longer directly comparable. However, it is important to observe whether the methods drive the logistic system to the same global and stable behavior on a long term optimization. The results are presented in Table 2 in terms of cost function f_L, number of orders delivered at the correct date $|O_D^0|$ and the number of orders delivered after the desired date $|O_0^T|$.

They represent the statistical average μ and standard deviation σ of 50 trials.

The results show that in general, the ACO is able to deliver more orders at the correct date and less orders with delay. On the other hand, the orders' delays are smaller for the GA, according to the average tardiness $\mu(T_j)$, as well as

Table 2
Statistical logistic system scheduling results at day $D = 30$

Instance	Method	$\mu(f_L)$		$\lvert O_D^0 \rvert$		$\lvert O_0^T \rvert$	
		μ	σ	μ	σ	μ	σ
(5, 5, 13, 10)	GA	57.09	20.90	89.36	4.24	86.16	3.91
	ACO	59.74	20.79	102.44	3.60	74.08	3.64
(10, 10, 2, 30)	GA	41.55	1.82	182.50	3.47	111.36	3.30
	ACO	41.67	1.65	188.64	2.94	105.64	3.01
(20, 10, 5, 20)	GA	35.81	0.71	369.75	21.10	254.63	27.13
	ACO	36.05	1.27	371.25	12.78	252.63	11.13

the stock size. From the cost function value perspective, the GA performance is slightly better than the ACO one. However, from the practical point of view, both solutions can be considered similar.

5.2. Dynamic optimization problems

In order to deal with the dynamic environment problem, there are different possibilities. The first approach is to optimize the new problem. However, depending on when the cancelation is made, there may be no time left to optimize the new problem once again. Another approach is to consider the final solution and deliver all the orders except the ones that were canceled. Nevertheless, this solution is most probably non-optimal. Using GA or ACO can introduce a third different approach. Since both methods consider a population of different solutions at the same time, there is a chance that one of the non-optimal solution within the population might be a better solution in the new logistic environment and in this way, this would become the new scheduling solution. But this is not very likely to occur, since most of the solutions population in both methods has majorally converged to the best found solution. The use of the ACO and the time that the algorithm spends to generate the pheromone matrix introduces a new approach to solve a dynamic optimization problem.

Consider the logistic example introduced in Section 3 and the GA and ACO solutions for that example in Section 4. The final solution given by the GA is the one that delivers the set of orders $\{6,7\}$. Consider that the second best solution is given by $\{3,7\}$. With this information, it can be assumed that the block $\{7\}$ is essential to achieve a good solution, and that delivering order 3 or 6 does not influence too much the final solution. The best solution given by the ACO is $(6,7)$ (and for this analysis we recall Fig. 3). This solution is different from the GA solution $\{6,7\}$, because it is a vectorial solution. The ACO algorithm provides the information that to achieve a good solution it is important to deliver first order 6, followed by order 7. This vectorial information can be very useful in environments with fast dynamics, where orders can be canceled at any time.

Consider now that suddenly order 6 is canceled and there is no time to run the scheduling algorithm one more time. In this case, the GA ends up with another optimal solution $\{3,7\}$, just by the inspection of the population of solutions. Consider however that the canceled order was order 7. In this case, the GA ends up with the sub-optimal solution $\{6\}$, because the second best solution (to deliver orders $\{3,7\}$) cannot be followed either, since it also includes the canceled order $\{7\}$. This example is depicted in Fig. 4. Observe that in this figure, the dark boxes represent the non-feasible part of the GA solutions due to stock constraints (orders $\{1,4\}$) or due to cancellation (order $\{7\}$).

On the other hand, with the ACO algorithm and the vectorial information, we can quickly define a new possible solution, just by inspecting the intensity of the pheromone

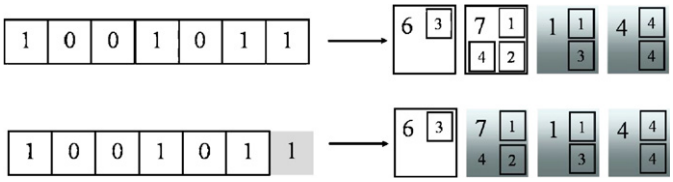

Fig. 4. GA solution for a dynamic logistic scheduling problem.

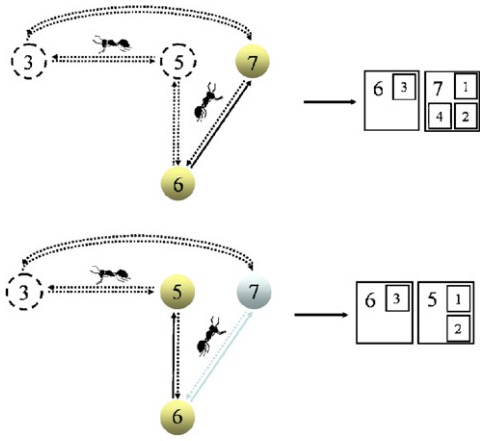

Fig. 5. ACO solution for a dynamic logistic scheduling problem.

trails left by the ants. If order 6 is canceled, it can be observed on Fig. 5, that delivering order $(7,3)$ is also a good solution. If order 7 is canceled, it can be seen that the arc connection $(6,5)$ is also strong, which means that solution is also good. Therefore, it is very easy to compute a new solution in a different optimization space through the analysis of the pheromone matrix.

The following simulation scenario was designed to prove this advantage of the ACO relatively to the GA. It is considered that after the scheduling process is finished, one order is randomly canceled. The GA adopts the best solution within the solution population that does not consider the canceled order. The ACO reconstructs a new solution based on the pheromone matrix: it traces the solution back to the node describing the order before the canceled one and then founds the rest of the solution following the arcs with the highest pheromone intensity. This heuristic procedure is very fast.

Observe that the simulation considers that the canceled order is the same. The results are presented in Table 3, where $\mu(f_L)$ represents the average logistic scheduling solution when the environment is static, $\mu(f_L^*)$ represents the average logistic scheduling result in a dynamic environment and $\mu(|f_L = f_L^*|)$ represents the average number of days where the ACO algorithm was able to find better logistic solutions for the static and dynamic environments. As it can be seen, the logistic system is less affected by the canceled orders when ACO is used. Since all experiences consider that only one order is canceled, the difference between GA and ACO becomes less significant from instance $(5,5,13,10)$ to instance

Table 3
Comparison of static and dynamic solutions at $D = 30$

Instance	Method	$\mu(f_L)$		$\mu(f_L^*)$		$\mu(\lvert f_L = f_L^* \rvert)$
		μ	σ	μ	σ	
(5, 5, 13, 10)	GA	57.09	20.90	62.015	18.65	–
	ACO	59.74	20.79	60.015	17.45	15.31
(10, 10, 2, 30)	GA	41.55	1.82	46.12	2.87	–
	ACO	41.67	1.65	44.88	1.34	23.76
(20, 10, 5, 20)	GA	35.81	0.71	37.18	1.73	–
	ACO	36.05	1.27	36.18	1.58	29.47

(20, 10, 5, 20), because the number of orders in the system increases. For many days, the ACO algorithm is able to find a solution of the dynamic problem that is equivalent to the static problem. Evidently, this becomes more easy for the larger instances.

In conclusion, the results show that in dynamic environments the use of ACO and its pheromone matrix allows to find good solutions without the need of computing a new solution.

6. Conclusions and future work

This paper presents a comparative study on GA and ACO, in order to decide which of the algorithms should be used on the optimization and rescheduling of a logistic system. The study is based on two different analysis: a literature survey, where the performances of GA and ACO are compared for different well-known benchmark problems, and on a detailed comparison for a logistic optimization problem.

Both analysis led to the same general conclusions: GA and ACO have good and similar performances for different instances of different optimization problems, but the GA have in general a lower computational burden. However, the extra-time required by the ACO is used to record information about the optimization procedure that can be used to reschedule the logistic system in dynamic environments.

Therefore, it can be concluded that for static optimization problems, GA and ACO can be both applied, although GA have lower computational burden. For dynamic optimization problems, ACO performs better than GA due to the use of information stored in the pheromone matrix during the optimization process.

In the future, the rescheduling of logistic systems using ACO will consider the complete supply-chain. A distributed optimization approach for supply-chain can be used, as in Silva et al. (2006). Further, as logistic processes deal often with many contradictory objectives and constraints, classical cost functions as (1) may not describe properly the optimization problem. In these cases, fuzzy optimization can be considered, as introduced recently in Silva et al. (2007).

References

Barbuceanu, M., Fox, M.S., 1996. Capturing and modeling coordination knowledge for multi-agent systems. International Journal of Cooperative Information Systems 5 (2–3), 275–314.

Blum, C., Sampels, M., 2004. An ant colony optimization algorithm for shop scheduling problems. Journal of Mathematical Modelling and Algorithms 3 (3), 285–308.

Bräysy, O., 1999. A new algorithm for the vehicle routing problem with time windows based on the hybridization of a genetic algorithm and route construction heuristics. In: Proceedings of the University of Vaasa, Research papers. p. 227.

Bräysy, O., 2001. Efficient local search algorithms for the vehicle routing problem with time windows. In: Proceedings of MIC'2001—4th Metaheuristics International Conference.

Di Caro, G., Dorigo, M., 1998. Antnet: distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9, 317–365.

Choi, I.-C., Kim, S.-I., Kim, H.-S., 2003. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. Computers & Operations Research 30, 773–786.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge, MA.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. The ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 26 (1), 29–41.

Fleurent, C., Ferland, J.A., 1994. Genetic hybrids for the quadratic assignment problem. In: Quadratic Assignment and Related Problems, vol. 16. American Mathematical Society, Providence, RI, pp. 173–187.

Gambardella, L.M., Taillard, E., Agazzi, G., 1999. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: New Ideas in Optimization. McGraw-Hill, New York, pp. 63–76.

Giffler, B., Thompson, G.L., 1960. Algorithms for solving production scheduling problems. Operations Research 8, 487–503.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Dordrecht.

Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C., 2005. A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research 167, 77–95.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press.

Jain, A.S., Meeran, S., 1999. Deterministic job-shop scheduling: past, present and future. European Journal of Operational Research 113, 390–434.

Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.

Lim, M.H., Yuan, Y., Omatu, S., 2000. Efficient genetic algorithm using simple genes exchange local search policy for the quadratic assignment problem. Computational Optimization and Applications 15, 249–268.

Merz, P., Freisleben, B., 1997. Genetic local search for the TSP: New results. In: IEEECEP: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence.

Michalewicz, Z., 1999. Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin.

Michalewicz, Z., Fogel, D.B., 2002. How to Solve It: Modern Heuristics, third ed. Springer, Berlin.

Nowicki, E., Smutnicki, C., 1996. A fast taboo search algorithm for the job-shop problem. Management Science 42 (6), 797–813.

Parunak, H.V.D., 1997. Go to the ant: engineering principles from natural multi-agent systems. Annals of Operations Research 75, 69–101.

Ross, S.M., 1996. Stochastic Processes, second ed. Wiley, New York.

Sarmiento, A.M., Nagi, R., 1999. Recent directions in integrated analysis of manufacturing-distribution systems. IIE Transactions on Scheduling and Logistics 31, 1061–1074.

Silva, C.A., Sousa, J.M.C., Runkler, T., Palm, R., 2005. Soft computing optimization methods applied to logistic processes. International Journal of Approximated Reasoning 40, 280–301.

Silva, C.A., Sousa, J.M.C., Runkler, T., Sá da Costa, J.M.G., 2006. Distributed optimization of logistic systems and its suppliers using ant colony optimization. International Journal of Science Systems 37 (8), 503–512.

Silva, C.A., Sousa, J.M.C., Runkler, T.A., 2007. Optimization of logistic systems using fuzzy weighted aggregation. Fuzzy Sets and Systems 158 (17), 1947–1960.

Stützle, T., Dorigo, M., 1999a. ACO algorithms for the quadratic assignment problem. In: New Ideas in Optimization. McGraw-Hill, New York.

Stützle, T., Dorigo, M., 1999b. ACO algorithms for the traveling salesman problem. In: Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications. Wiley, New York, pp. 163–183.

Swaminathan, J.M., Smith, S.F., Sadeh, N.M., 1998. Modeling supply chain dynamics: a multiagent approach. Decision Sciences Journal 29 (3), 607–632.

Tsai, H.-K., Yang, J.-M., Tsai, Y.-F., Kao, C.-Y., 2003. Some issues of designing genetic algorithms for traveling salesman problems. Soft Computing—A Fusion of Foundations, Methodologies and Applications. Springer, Berlin Paper 1433-7479 (Online).