A LOGISTIC PROCESS SCHEDULING PROBLEM: GENETIC ALGORITHMS OR ANT COLONY OPTIMIZATION?

C.A. Silva [*,1] **, J.M. Sousa** [*,3] **, T. Runkler** [**,2] **and J.M.G. Sá da Costa** [*,3]

[*] *Instituto Superior Técnico, Dept. of Mechanical Eng., GCAR/IDMEC, 1049-001 Lisbon, Portugal*
[**] *Siemens AG, CT-IC, Neural Computation Dep., 81730 Munich, Germany*

Abstract: This paper compares the optimization of a logistic scheduling problem using two different optimization techniques; the genetic algorithms and the ant colony optimization. The comparison is preceded by a literature review that summarizes the available comparison results for different benchmark problems and tries to generalize the differences between the techniques. The simulation results for the logistic problem confirm the conclusions of the literature survey: both methods perform equally well, but in general the genetic algorithms are faster. However, the ant colonies give more information about the solution, which is advantage in some applications. *Copyright©2005 IFAC.*

Keywords: Genetic algorithms, Ant colony optimization.

1. INTRODUCTION

In order to improve competitiveness and profitability, most of the companies today are organized as *supply-chains*: a world-wide network of external partners (suppliers, warehouses and distribution centers) through which raw materials are acquired, transformed into products and delivered to costumers (Barbuceanu and Fox, 1996). Logistics are the subprocess of the supply chain that collects the goods purchased to external suppliers and ships them to the different costumers on time, minimizing the delivery time and the storage costs. Therefore, the control of a logistic process consists of optimizing a *scheduling problem*.

Nowadays, meta-heuristics like *genetic algorithms* (GA) (Holland, 1975) or more recently *ant colony optimization* (Dorigo *et al.*, 1996) (ACO) are considered to be very powerful scheduling techniques (Jain and Meeran, 1999). These techniques are very appealing to design an optimization method for a new scheduling application, because they are simple to implement and have proved to be efficient in finding optimal solutions for different applications. But which method should we choose? Is there a method that guarantees a higher chance of finding the global optimum?

To answer these questions, it is important to have comparison results of different methods for the same applications. However, there are not many works on direct comparison. Moreover, when different papers focus different algorithms for the same problem, they often lack statistical analysis and do not consider the

[1] This work is partially supported by the Portuguese Foundation for Science and Technology (FCT) under Grant no. SFRH/BD/6366/2001.
[2] This work is partially supported by the German Ministry of Education and Research (BMBF) under Contract no. 13N7906 (project Nivelli).
[3] This work is partially supported by "Programa de Financiamento Plurianual de Unidades de I&D (POCTI), do Quadro Comunitário de Apoio III", FCT, Ministério do Ensino Superior, da Ciência e Tecnologia, Portugal.

same measurements or number of trials. At the end, one ends up using either the most common technique, the GA, or the method one knows better. If there is time and expertise knowledge, it is even better to test more than one method and choose the best performing technique. But the questions remain: are there any elements that might tell us before hand which algorithm is better to our application?

In this paper, we try to answer this question for the GA and ACO algorithms on the logistic scheduling problem, and we try to generalize the conclusions for other type of optimization problems.To do so, Sec. 2 presents a literature survey on comparisons between the methods and preliminary comparison conclusions. Sec. 3 describes the logistic scheduling problem and the GA and ACO implementations to solve this problem. The comparison results for the logistic problem are present and discussed in Sec. 4. Section 5 concludes the paper and indicates the future research steps.

2. LITERATURE REVIEW

In order to compare the GA and the ACO algorithms, a literature survey is presented on some of the most representative combinatorial optimization problems. The GA has been in the last decade and half the most studied meta-heuristic and it is easy to find applications in all sort of bench-mark problems. As for the ACO, because it is a relatively new method, it has been applied in less bench-mark problems.

Table 1 represents the summary of the comparison results for the *Traveling Salesman Problem*, symmetric (TSP) or asymmetric (ATSP), the *Quadratic Assignment Problem* (QAP), and the *Vehicle Routing Problem with Time Windows* (VRPWT) as the most studied bench-mark combinatorial optimization problems. For the *job shop scheduling problem* (JSSP) there are no ACO implementations found for the most relevant benchmark problems, therefore we present results for a special type of single machine scheduling problem (SMP).

This table indicates the algorithms performance in terms of percentage of mean relative error for several trials. The closer the mean value to the zero, the more accurate the algorithm is. It also indicates, when possible, the time required to find the optimization result. However, this is not a very accurate indicator of the computational effort of each algorithm, since it depends on the machine and the programming.

2.1 Traveling Salesman Problem

The latest comparison between GA and ACO for the symmetric TSP problem is the one presented in (Tsai *et al.*, 2003). The results show that both methods are very good, with an average error smaller than 1%

Table 1. Error (%) to the best known solution

Instance	GA			ACO		
	Best	Mean	Time	Best	Mean	Time
TSP - GA ((Tsai *et al.*, 2003)) and ACO((Stützle and Dorigo, 1999*a*))						
lin318		0	15		0	94.2
pcb442		0	33		0.26	308.9
att532		0	97		0.08	387.3
rat783		0	376		0.10	965.2
pcb1173		0.001	1177		0.105	3219.5
ATSP - GA [(Merz and Freisleben, 1997)] and ACO [(Stützle and Dorigo, 1999*a*)]						
ry48p	0	0.2	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
QAP - GA [(Fleurent and Ferlan, 1994)] and ACO [(Stützle and Dorigo, 1999*b*)]						
tai20a		0.268			0.191	
tai80a		0.796			0.836	
nug30		0.007			0.013	
sko42		0.003			0.032	
bur26a		0.043			0.006	
tai20b		0.0			0.0	
tai80b		0.829			0.591	
VRPTW GA [(Bäysy, 2001), (Bräysy, 1999)*] and ACO [(Gambardella *et al.*, 1999)]						
R1 (V)		12.0	118		12.4	210
(D)		1240			1211	
C1* (V)		10			10	
(D)		829			828	
RC1 (V)		11.5	95		11.7	210
(D)		1418			1382	
R2 (V)		3	105		2.7	210
(D)		1017			968	
C2* (V)		3			3	
(D)		592			590	
RC2 (V)		3.3	44		3.3	210
(D)		1200			1149	
SMP - GA [(Rubin and Ragatz, 1995)] and ACO [(Gagn *et al.*, 2002)]						
Prob408	0	0	-	0	0.011	13.2
Prob508	0.0139	0.0139	-	0.0139	0.124	102.9
Prob608	0.0015	0.0046	-	0	0.034	880.35
Prob708	0.0654	0.107	-	0.0077	0.0221	2336.1

for the ACO and a null average error in all instances except one for the GA. It is also possible to observe that the computational effort for the ACO is one order of magnitude higher than for the GA.

For asymmetric instances, there are much less GA implementations than for the symmetric case, mainly because for many genome encoding, the implementation is not straightforward - here the solution cannot be represented by a simple permutation, as in the TSP. The presented results concern genetic local search (Merz and Freisleben, 1997). There are more recent results (Choi *et al.*, 2003), however, the results are not very clear and do not show any special improvement, therefore they are not considered. For the ACO algorithm, the codification is exactly the same for symmetric or asymmetric instances, due to the graph searching nature of the algorithm. ACO always finds the best solutions, while the GA fail occasionally to find them. The computational effort shows that for ACO, the time's 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 for the same type of instances.

These results show an advantage of ACO to solve asymmetric problems, which indicates that it is more difficult to implement GA on optimization problems described by graphs where the weights in the arcs can assume different values, i.e where the solution contains vectorial information and it is not just some set of values as in a permutation.

2.2 *Quadratic assignment problem*

For the Quadratic Assignment Problem, there is a direct comparison presented in (Stützle and Dorigo, 1999*b*), where the ACO algorithm for the QAP is compared with the GA implementation of Fleurent and Ferland introduced in (Fleurent and Ferlan, 1994). There is a 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 we omit those results.

The QAP problem can be easily depicted in a graph, and the problem's solution can be represented by a simple permutation. In this sense, it is very easy to implement both GA or ACO methods to optimize this problem. The results present in Table 1 show that also here, both GA and ACO present similar performances, again with an average error from the best known solution smaller than 1%. The computational effort is not explicitly compared, but it is mentioned in (Stützle and Dorigo, 1999*b*) that the ACO algorithm is 25% slower than the GA approach. This difference can be easily explained by the difference in maturity of the implementation/code of the algorithms.

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

2.3 *Vehicle Routing Problem*

For the Vehicle Routing problem, it exists a systematic comparison work by Bräysy (Bräysy, 1999; Bäysy, 2001) between different methodologies for the problem with time windows. Table 1 presents the results in terms of number of vehicles (V) and travel distance (D). 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 customers clustering part of the problem, since it uses less vehicles. The computational effort seems to be smaller for the GA approach, however, notice that the ACO use a fixed computational time, which means that they probably find the optimum in less time for most of the instances. Again in this application, the ACO seem to be more fitted to solve problems represented by graphs, while the GA are better to solve problems that concern the evaluation of a set of items.

2.4 *Single Machine Problem*

We havent found any ACO implementations for the most famous job-shop problems. The only comparison work found on this topic is the work by Gagné *et al* presented in (Gagn *et al.*, 2002). The results presented in Table 1 concern this work. They show that also for this problem both GA and ACO perform equally well and there are no significative differences. The lack of results in the manufacturing field clearly indicates that the ACO algorithm does not perform well in this type of problems. On the other hand, there are a lot of GA implementations for these benchmark problems, but they are not the best performing algorithm.

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.

3. THE LOGISTIC SCHEDULING PROBLEM

This paper considers a simplified model of the real-world logistic process at Fujitsu Siemens Computer (Silva, 2001), a company that sells computers and hardware solutions made of components bought from external suppliers. Every day, the logistic systems collects several requests from different clients, designated by *orders* $o_j \in O$, a set of different types of components c_i, in certain quantities q_{ij}. Each order is characterized by a due date d_j and the release date r_j. The components c_i are purchased to external suppliers and delivered at the logistic center after a certain time p_i, ready to be assigned to the orders waiting in the orders list O. The decision process occurs at this point. The system has to observe the stock list and the orders list, and check which orders have all the components available to be delivered. The objective is to match the release date r_j with the due date d_j. Considering that the system never releases orders before the due date, the difference between the completion date and the due date of the order o_j is called the *tardiness* T_j. The orders should be delivered on the correct date, not after, which means that the objective is to have $T_j = 0$ for all orders. This decision step is done once per day, either by some dispatching rule or by some other optimization procedure. After the selection of the orders to be delivered, the orders are shipped to the clients.

The system can be disturbed if the suppliers production times p_i are different than expected, i.e if the components enter the system before or after the expected date. Another disturbance factor appears if the system accepts due dates d_j different from the expected delivery date, because a client is very important and the order cannot be lost to a competitor.

The scheduling problem consists of optimizing daily the cost function defined as *Global Expected Tardiness* (GET). Let O_D be the set of delivered orders and $\overline{O_D} \subset O$ be the complementary set of O_D of orders not delivered, such that $\overline{O_D} \cup O_D = O$. The GET function to be minimized is given by

$$f = \frac{\sum_{j \in O} T_j + \#\overline{O_D}(T_l > 0)}{\#O_D(T_j = 0)} \tag{1}$$

where $\sum_{j \in O} T_j$ accounts for the minimization of the tardiness of the orders O in the system; $\#\overline{O_D}(T_l > 0)$ refers to the minimization of the number of orders that are not delivered and are already delayed; and finally $\#O_D(T_j = 0)$ accounts for the maximization of the number of orders delivered at the correct date.

This problem is NP-Hard (Silva *et al.*, 2003), therefore we use meta-heuristics, like GA and ACO, to optimize it. In principle, GA will perform well, since the optimization concerns the selection of a best set of items. However, the problem can be easily described by a graph, therefore it is expectable that the ACO also performs well. It is also expectable that GA are faster than ACO, although computational time is not a constraint for this application.

3.1 GA implementation

In this problem, a binary encoding is used, since the problem is intrinsically discrete. The solutions are vectors with the size of all the orders waiting to be delivered, n, with value 0 (zero) if the order is not delivered today, and value 1 if it is. The initial population is initialized as random binary strings. The selection is elitist, the rate is 0.4 and it is based on the fitness function is the one defined in (1). The crossover method is the *one-point* crossover, with a rate of 1, i.e. all parents create offsprings. The mutation rate is low.

If an infeasible solution is created after the genetic operations, this solution is transformed into a feasible solution before the algorithm proceeds. The transformation of an infeasible solution into a feasible one consists of randomly checking for each gene with value 1 if there are enough components in the stock to deliver the order associated with that gene. If yes, the gene's value remains 1 and the stock is updated. If there are not enough components in the stock, the gene's value is changed from 1 to 0.

The algorithm runs for $O(g \times N) \approx O(N^2)$ time, where N is the maximum number of iterations allowed and g is the size of the population.

3.2 ACO implementation

Considering the representation of the problem on a disjunctive graph, the nodes of the graph are the orders waiting to be delivered, and the role of the ant is to find the minimum cost path connecting the orders. We consider that each ant is carrying a bag with the available stocks and distributing them among the orders. It only visits orders whose components it is able to deliver, therefore the ACO only builds feasible solutions. When the stocks' bag is empty or the remaining components are not enough to deliver any missing order, the search for this ant is finished. The objective function to be minimize by each ant k is f_L^k defined in (1). The probability of an ant k choosing the next order to deliver is given by:

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

The pheromone matrix τ is limited to $]\tau_{min}, \tau_{max}]$, with $\tau_{min} = 0$ and $\tau_{max} = 1$. The heuristic function η is the order's tardiness: if an order has already a positive tardiness, the ant will feel a stronger attraction to visit it, because the order is already delayed. We define it as an exponential function in the interval $[0, 1]$ where the value 0 is for the order that has the minimum lateness T_{min} and 1 is for the most delayed order T_{max}:

$$\eta_j = \frac{e^{\frac{T_j - T_{min}}{T_{max} - T_{min}}} - 1}{e - 1} \tag{3}$$

Notice that in this case the heuristic information is only order dependent, therefore $\eta_j = \eta_{ij}$. The Tabu list Γ is the list of orders already delivered by the ant and also the orders which is not possible to visit, due to lack of stocks. The parameters α and β measure the relative importance of trail pheromone and heuristic knowledge, respectively. Since the pheromone trails τ_{ij} and the heuristic values η_{ij} are restricted to the interval $[0, 1]$, $\alpha < \beta$ will indicate a higher relative weight of the pheromones trail. The pheromone update, considering an evaporation of $\rho \in [0, 1]$ updates the pheromones deposited on the trails (i, j) followed by ant q that found the best solution $f^q(s)$ for the current iteration.

4. COMPARISON FOR THE LOGISTIC PROBLEM

Every day, there is a new logistic problem to be solved. However, to analyze the performance of the logistic system, it is necessary to consider a larger time window (weeks/months). In this case, we consider a sequence of 30 days in order to compare both methodologies, as described in (Silva *et al.*, 2003). Table 2

Table 2. Solutions after 30 days.

Method	$\#T_j = 0$	$\#T_j > 0$	$\xi(T_j)$	$\max(T_j)$	$\sum_{l \in \overline{O_D}} \lvert T_l \rvert$
GA (best)	388	60	1.62	20	4
GA (ξ, σ)	(383.4,3.44)	(64.8,4.01)	(1.62,0.005)	(19,0.71)	(3.8,0.84)
ACO (best)	387	62	1.62	13	3
ACO (ξ, σ)	(382.8,2.95)	(66,2.74)	(1.62,0.003)	(11.8,0.84)	(3.2,0.45)

Table 3. *t-test* probabilities.

	$\#T_j = 0$	$\#T_j > 0$	$\xi(T_j)$	$\max(T_j)$	$\sum_{l \in \overline{O_D}} \lvert T_l \rvert$
t-test	0.39	0.29	0.26	< 0.05	0.097

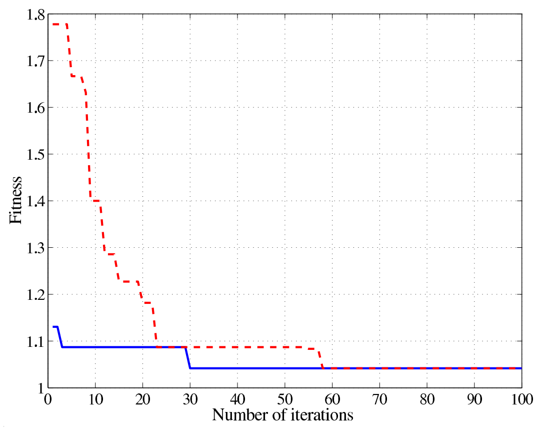

Fig. 1. Convergence in iterations: GA(..) and ACO(-).

presents the results in terms of best result, mean result and standard deviation. Both methods perform equally well, i.e. the differences between the optimization results is not statistical significant, except one. This conclusion is based on the *t-test* (Ross, 1996) results presented in Table 3. We tested the hypothesis that the results of the GA and the ACO methods belong to the same population, considering a *level of significance* of 0.05. This hypothesis was confirmed for all parameters, except the maximum variance $\max(T_j)$. However, this parameter is deceiving, since it might be caused by a single case where an order was delivered with a high delay. Nevertheless, for all the other parameters, the results are identical. The results show further that both the GA and ACO are individually quite robust methods when applied to this problem, since the mean result for each method is very near its best result and the standard deviation is not very high.

A typical run of the GA and ACO algorithms in terms of fitness improvement of the population for a 1 day problem, assuming the same initial conditions, can be seen in Figures 1 and 2. There we present the convergence results both in terms of number of iterations and computational time. From these figures, we observe the main differences between the GA and the ACO algorithm. In Figure 1 the solutions of the ACO are very good right from the start, while the GA take more iterations until they converge to the optimal solution. This can be explained by the fact that the ACO method is using a local heuristic to guide the search while constructing the solution.Although the GA uses the same heuristic to guide the search, this heuristic is only used to improve the solution at the end. This explains also why the GA and ACO results

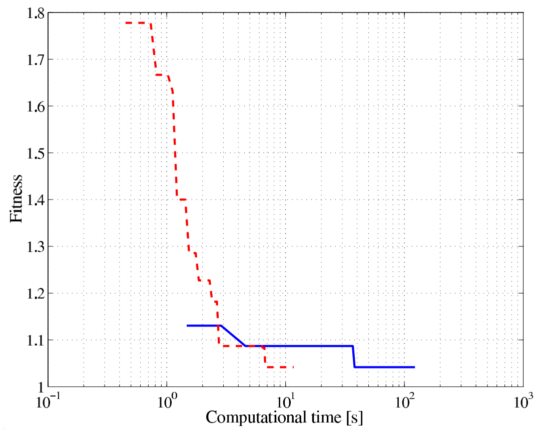

Fig. 2. Convergence in time: GA(..) and ACO(-).

are different for the $\max(T_j)$ parameter: the GA might fail a single order that is either very delayed (because is not in the solution), although the solution is also very good, or that at the end the solution does not take advantage of all the stocks.

In Figure 2 we can see another important difference: the GA algorithm is faster than the ACO algorithm, since the number of basic operations are $O(N^2)$ and $O(N^3)$ respectively (although the ACO takes less iterations to find the optimum). The reason that explains this difference is the fact that the GA algorithm creates a solution in one single step, while the ACO creates a solution by iterating in the disjunctive graph step by step.

The extra time required by the ACO algorithm is reflected in extra information at the end of the algorithm run. The solution is not only a set of solutions, but it also contains a vectorial information that can be very useful in environments with fast dynamics, where orders can be cancelled at any time. If suddenly 1 order is cancelled, the GA has to restart the optimization process or it will end up with a sub-optimal solution; with the ACO algorithm and the vectorial information present in the pheromone matrix, we can quickly define a new optimal solution, just by inspecting the intensity of the pheromone trails left by the ants. This advantage of the ACO algorithm, as a solution constructor algorithm, has already been explored in the optimization of communication networks (Caro and Dorigo, 1998), a highly dynamic problem. Moreover, the pheromone matrix, by keeping an indirect record of the optimization steps towards the optimal solution, can be used in the integration of the scheduling system in a management system of a supply chain: the ants in the ACO framework can read the information of the environment, update and exchange that information. In this way, the scheduling system could be connected to other subsystems of the supply chain using ant colonies as a multi-agent system (Silva *et al.*, 2004).

5. CONCLUSIONS

This paper discusses the optimization methodologies that can be used to solve scheduling problems: GA and ACO. Their performances are similar and the choice upon which should be used has to consider more variables. The GA is faster than the ACO, because the managed information during the optimization process is smaller. At the end, the GA presents a black-box type of solution, while the ACO presents a grey-box type of solution. It is possible to extract more information about the optimization process that can be used either if the problem becomes a dynamic problem, or if the scheduling method is integrated in a multi-agent solution for the global supply chain.

REFERENCES

Barbuceanu, M. and M. Fox (1996). Coordinating multiple agents in the supply chain. In: *Proceedings of the Fifth Workshops on Enabling Technology for Collaborative Enterprises, WET ICE'96*. IEEE Computer Society Press. pp. 134–141.

Bäysy, Olli (2001). Efficient local search algorithms for the vehicle routing problem with time windows. In: *Proceedings of MIC'2001 - 4^{th} Metaheuristics International Conference*.

Bräysy, Olli (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 Univeristy of Vaasa, Research papers 227*.

Caro, Gianni Di and Marco Dorigo (1998). Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research* **9**, 317–365.

Choi, In-Chan, Seong-In Kim and Hak-Soo Kim (2003). A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers & Operations Research* **30**, 773–786.

Dorigo, Marco, Vittorio Maniezzo and Alberto Colorni (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. and J.A. Ferlan (1994). *Quadratic assignment and related problems*. Chap. Genetic hybrids for the quadratic assignment problem, pp. 173–187. Vol. 16. American Mathematical Society.

Gagn, C., W.L. Price and M. Gravel (2002). Comparing an aco algorithm with other heuristics for the single machine scheduling problem with sequence dependent setup times. *Journal of the Operational Research Society* **53**, 895–906.

Gambardella, Luca Maria, Éric Taillard and Giovanni Agazzi (1999). MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In: *New Ideas in Optimization* (David Corne, Marco Dorigo and Fred Glover, Eds.). pp. 63–76. McGraw-Hill.

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

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

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

Merz, Peter and Bernd Freisleben (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*.

Ross, Sheldon M. (1996). *Stochastic Processes. 2nd edition*. John Wiley & Sons, Inc. New York.

Rubin, P.A. and G.L. Ragatz (1995). Scheduling in a sequence dependent setup environment with genetic search. *Computers and Operations Research* **22**(2), 85–99.

Silva, C. A., T. A. Runkler, J. M. Sousa and J. M. Sá da Costa (2003). Optimization of logistic processes in supply–chains using meta–heuristics. In: *LNAI 2902, Progress in Artificial Intelligence* (Fernando Moura Pires and Salvador Abreu, Eds.). pp. 9–23. Springer Verlag. Berlin Heidelberg.

Silva, C.A., J.M. Sousa, T.A. Runkler and J. Sá da Costa (2004). A multi-agent approach for supply chain management using ant colony optimization. In: *Proceedings of the IEEE SMC 2004, International Conference on Systems, Man and Cybernetics*. pp. 1938–1943.

Silva, Carlos A. (2001). Arbeitspaket S1: Systemanalyse - Fujitsu Siemens Computers. Technical Report 1. Siemens AG, Corporate Technology, Department of Neural Computation, CT IC-4.

Stützle, Thomas and Marco Dorigo (1999*a*). *Evolutionary Algorithms in Engineering and Computer Science:Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*. Chap. ACO Algorithms for the Traveling Salesman Problem, p. 163183. John Wiley & Sons.

Stützle, Thomas and Marco Dorigo (1999*b*). *New Ideas in Optimization*. Chap. ACO Algorithms for the Quadratic Assignment Problem. McGrawHill.

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