European Journal of Operational Research 197 (2009) 1185–1191

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Ant colony optimization with a specialized pheromone trail for the car-sequencing problem

Sara Morin, Caroline Gagné *, Marc Gravel

Département d'informatique et de mathématique, Université du Québec à Chicoutimi, 555, Boul. de l'Université, Chicoutimi, Québec, Canada G7H 2B1

ARTICLE INFO

Article history:
Available online 29 March 2008

Keywords:
Ant colony optimization
Pheromone trail
Scheduling
Car-sequencing problem

ABSTRACT

This paper studies the learning process in an ant colony optimization algorithm designed to solve the problem of ordering cars on an assembly line (car-sequencing problem). This problem has been shown to be NP-hard and evokes a great deal of interest among practitioners. Learning in an ant algorithm is achieved by using an artificial pheromone trail, which is a central element of this metaheuristic. Many versions of the algorithm are found in literature, the main distinction among them being the management of the pheromone trail. Nevertheless, few of them seek to perfect learning by modifying the internal structure of the trail. In this paper, a new pheromone trail structure is proposed that is specifically adapted to the type of constraints in the car-sequencing problem. The quality of the results obtained when solving three sets of benchmark problems is superior to that of the best solutions found in literature and shows the efficiency of the specialized trail.

1. Introduction

Metaheuristics are well adapted to solve combinatorial optimization problems in practical situations. They allow an interesting compromise between the solution quality and computation time. This is one reason why the domain has grown so much in the last years. Many methods were introduced, among which are ant colony optimization (ACO), a solution strategy based on actual ant behavior.

The ACO metaheuristic mimics the behavior of evolutionary sharing of learning (stigmergy) observed in actual ant colonies using a central memory device called the "pheromone trail" which is updated by every individual in the colony. Ants use this memory not only as a guide to building solutions but also in updating the memory itself proportionally to the quality of the solutions they produce. ACO is then a constructive approach where the choice of the next element in a solution is made by compromising between a shared learning and a greedy heuristic specially designed for the problem.

This metaheuristic was found efficient in solving many problems since its introduction in the 90s. The first ACO algorithm, called ant system (AS) [1–3], was designed to solve the traveling salesman problem (TSP). At that time, AS produced encouraging initial results although it was not competitive with the best known TSP algorithms [4]. Since then, many propositions have been made to improve its performance and today, the most advanced version of the ant algorithms is the ant colony system (ACS), which is one of the most powerful in solving combinatorial optimization problems [5].

Pheromone trail management is the central element in the different ACO algorithms and covers, in particular, all aspects of the pheromone update scheme and its use in the transition rule. Pheromone trail management is the property that is most often modified in literature [4] and which has led to different versions of the ACO algorithm such as the ant system, the Elitist ant system, the rank-based ant system and the MAX–MIN ant system. However, there have been very few publications dealing with changing the internal structure of the trail (encoding pheromone). In fact, the structure is generally a direct transposition of the one used for the TSP, even if the nature of the addressed problems is different.

In this paper, the *car-sequencing* problem that consists in sequencing cars on an assembly line is studied. This combinatorial optimization problem is defined as NP-hard and is of great practical interest. The objective of this paper is to propose a new pheromone trail structure specially designed to solve this problem and to evaluate its performance.

2. Description of the car-sequencing problem

The *car-sequencing* problem seeks to determine the best sequence of production of a number of vehicles (*nb_cars*) in a manufacturing process comprising three shops: body, paint and assembly. In reviewing literature, one often finds that only assembly is taken into account in determining a sequence which is then

* Corresponding author. Tel.: +1 418 545 5011; fax: +1 418 545 5017.
E-mail addresses: sara_morin@uqac.ca (S. Morin), caroline_gagne@uqac.ca (C. Gagné), marc_gravel@uqac.ca (M. Gravel).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.03.033

		Classes											
Option	Dispersion constraint (r_o/s_o)	1	2	3	4	5	6	7	8	9	10	11	12
1	1/2		•	•				•	•	•			
2	2/3	•		•	•	•		•				•	•
3	1/3		•				•			•	•	•	
4	2/5				•		•		•				•
5	1/5		•			•							
	c_i	3	1	2	4	3	3	2	1	1	2	2	1

(a) Data problem

Position	.	.	9	10	11	12	13	14	.	.
S / *option*			**2**	**11**	**5**	**8**	**4**	**7**		
1			•			•		•		
2				•	•		•	•		
3			•	•				•		
4						•	•			
5			•		•					

(b) Partial evaluation of the solution

Fig. 1. Sample *car-sequencing* problem.

applied to the entire process [6,7]. During assembly, various components are added to the already painted body. Each car is equipped with a particular set of complex options (air-conditioning, sunroof, etc.) chosen from a set *O*. Cars requiring these options must be dispersed throughout the sequence so as to smooth the workload at various critical workstations. The dispersion required for balancing the workload is expressed by a ratio r_o/s_o meaning that in a consecutive sequence of s_o vehicles, at most r_o vehicles can require option *o*. If at some point in the sequence such a *dispersion constraint* is not respected, we say that there is a *conflict*. Optimization of the sequence is then the minimization of the number of conflicts linked to the dispersion constraints. In the solution process, identical cars – those having the same set of options – are grouped into *classes* of vehicles. For each of the *v* classes so formed, we know c_i the number of vehicles that must be produced for class *i*. These requirements constitute the production constraints of the problem.

In Fig. 1, we present sample problem data so as to illustrate how a solution is evaluated. The problem data appear in Fig. 1a; we see that there are 25 vehicles distributed among 12 classes derived from 5 options. In Fig. 1b, a portion of a solution *S* is presented. The shaded areas indicate, for each option, the positions in the sequence for which the dispersion constraints are not satisfied.

The *car-sequencing* problem with dispersion constraints in the assembly shop has been shown to be NP-hard in the strong sense [8]. Lopez and Roubellat [9] published a review of the literature for this problem and reported on exact methods, tree-search methods, constructive methods and neighborhood-search methods of solution. Recent works in solving the *car-sequencing* problem include neighborhood-search approaches [10–12], simulated annealing [13], genetic algorithms [14], neural network [15] and various ACO algorithms [12,16,17].

3. Literature review on solving the car-sequencing problem with ACS

The authors previously designed an algorithm, which will referred to as ACS-2D ("2D" refers to the 2-dimensional nature of the pheromone trail matrix), for the solution of the *car-sequencing* problem [16]. The results produced by ACS-2D will serve to benchmark the compared algorithms in this paper. The main elements of this algorithm are presented to facilitate understanding of the modifications and adaptations proposed in the remainder of this paper. Fig. 2 presents a summary of the algorithm's main steps.

The pheromone trail is stored in a two-dimensional symmetric matrix τ of size $v \times v$, where *v* is the number of classes in the problem. Matrix element $\tau_{i,j}$ is a measure of the desirability of placing a car of class *i* adjacent to a car of class *j*. The matrix diagonal is not empty, since it is possible to place cars of the same class adjacent in the sequence. Pheromone trail updates are done in symmetric fashion. At the start of the algorithm, each element of the matrix is initialized to τ_0, which is a small positive value.

The algorithm proceeds for a number of cycles (NbCycles), or for as long as the number of conflicts (L^{Bg}) in the best known solution (S^{Bg}) is greater than zero. At the beginning of each cycle, we determine the starting vehicle class for each of the *m* ants. From position 2 to position *nb_cars*, vehicle classes *j* are determined in sequential fashion using a *transition rule* (see Eq. (1)). This rule is based on the pseudo-proportional random rule [5] and includes both a deterministic decision rule and a second, probabilistic rule. A parameter q_0 allows the determination of the proportion of decisions that are made using the deterministic rule. In addition to the pheromone trail, the transition rule proposed by the authors uses two visibility terms. The first favors the choice of vehicle classes that generates the least number of new conflicts ($\eta 1_j$) and the second favors the classes of vehicles that are most difficult to place in the sequence ($\eta 2_j$). The exponents that determine the relative importance of the pheromone trail and to the two visibility terms are α, β and δ, respectively

$$j = \begin{cases} \arg\max_{\ell \notin tabu_k} \{[\tau_{i,\ell}]^{\alpha} \cdot [\eta 1_\ell]^{\beta} \cdot [\eta 2_\ell]^{\delta}\} & \text{if} \quad q \leqslant q_0 \\ J & \text{if} \quad q > q_0, \end{cases} \tag{1}$$

where *J* is chosen according to the probability :

$$P_{ij}^{k} = \frac{[\tau_{i,j}]^{\alpha} \cdot [\eta 1_j]^{\beta} \cdot [\eta 2_j]^{\delta}}{\sum_{\ell \notin tabu_k} [\tau_{i,\ell}]^{\alpha} \cdot [\eta 1_\ell]^{\beta} \cdot [\eta 2_\ell]^{\delta}}.$$

In addition, the algorithm uses a list of *cl* candidate classes of vehicles in the transition rule to reduce computing time by restricting the number of classes from which class *j* is chosen by a given ant.

```
t = 0;
Set the pheromone trail matrix τ at τ0 for each pair of classes ij;
WHILE t < NbCycles and L^Bg > 0
        Initialize the first position of the sequence for each ant by a random choice from among
        the classes of cars
        FOR pos = 2 to nb_cars DO
                FOR k = 1 to m DO
                        Choose the class j in position pos, j ∉ Tabu_k, to be added to the sequence
                        after the class i among the cl candidate classes using the transition rule
                        (Equation 1);
                        Local update of the pheromone trail for (i,j) (Equation 2);
        FOR k = 1 to m DO
                Evaluate the number of conflicts L^k for the solution k;
                Global update of the pheromone trail using S^Bc, the best solution of the current
                cycle (Equation 3);
                Update the best solution so far S^Bg
        t = t+1
```

Fig. 2. Summary presentation of the ACS-2D algorithm for *car-sequencing* [16].

In addition, we ensure that any sequence produced by an ant respects the production constraints. To achieve this, each ant k uses a list $tabu_k$ containing the classes for which no more vehicles remain to be placed. More details concerning the transition rule are contained in the original article [16].

Following the choice by an ant of a class j vehicle to follow a class i vehicle, a *local update* of the pheromone trail for element $\tau_{i,j}$ is carried out so as to slightly reduce the quantity of pheromone (Eq. (2)). This helps to avoid excessive choice of the same classes by other ants and favors diversification in the search process. The parameter ρ_ℓ represents the degree of local persistence of the pheromone trail

$$\tau_{i,j} = \rho_\ell \cdot \tau_{i,j} + (1 - \rho_\ell) \cdot \tau_0. \tag{2}$$

At the end of the cycle, a *global update* of the pheromone trail is done for the ant having obtained the best solution (S^{Bc}) for the cycle. As presented in Eq. (3), an evaporation of the pheromone is first carried out on the elements of the matrix τ using a degree of global persistence of the trail ρ_g. Then, pheromone is added to each pair of classes of vehicles ij belonging to S^{Bc}. The increase is proportional to the quality of the solution L^{Bc} and to the number x of times the pair ij appears in the solution. Contrary to the TSP, which involves the construction of a Hamiltonian tour, the *car-sequencing* problem is transformed by the grouping of identical cars into classes, which invites the repetition of particular sequences of classes in a feasible solution. It is for this reason that the repetition of these patterns is taken into account in the global update of the pheromone trail.

$$\tau_{i,j} = \rho_g \cdot \tau_{i,j} + (1 - \rho_g) \cdot \Delta\tau_{i,j}^{Bc} \text{ where } \Delta\tau_{i,j}^{Bc} = \begin{cases} x_{i,j} \cdot \frac{1}{L^{Bc}} & \text{if } ij \in S^{Bc} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

In general, the values of the different parameters of the ACS-2D algorithm follow the recommendations of Dorigo and Gambardella [18]. In the remaining cases, the parameters were chosen following numerical experiments and sensitivity analysis. The following values were assigned to the different parameters for all problems: $\tau_0 = 0.005$, $\rho_g = \rho_\ell = 0.99$, $m = 15$, $q_0 = 0.9$, $NbCycles = 1000$ and $\{\alpha, \beta, \delta\} = \{1, 6, 3\}$. The results produced by the ACS-2D algorithm are presented in the following section.

4. Proposal of a new pheromone trail structure specially designed for car-sequencing problem solving

The structure of the pheromone trail used by Gravel et al. [16], inspired by that used originally for the TSP, attained a good level of performance in solving the car-sequencing problem. The nature of the car-sequencing problem is, however, different and that has motivated an important modification both of the structure and the use of the pheromone trail. Steps that led to this modification are presented in Sections 4.1 and 4.2, and a demonstration of the performance of the resulting specialized trail is done in Section 4.3.

4.1. Use of the pheromone trail on a horizon

The pheromone trail used by Gravel et al. [16] is structured to represent the advantage of having a car of class i next to a car of class j. However, due to the fact that the dispersion constraints r_o/s_o of the options spread over several positions, the classes of cars a little farther from the current position have as much an influence on the choice of the next class of car as the ones closer. Indeed, a class of car may generate conflicts with the s_{max} surrounding positions, where s_{max} is the spread of the longest constraints among the O dispersion constraints of type r_o/s_o. For the car-sequencing problem, it may then be suitable to look for the benefit of placing a car of a given class by considering its interaction not only with the adjacent classes, but also with the set of classes of the cars placed near it.

In order to reflect the interest of inserting a car of a given candidate class at the next position in the sequence, the use of the transition rule is modified to consider the sum of the accumulated trail between this candidate class and the classes of the s_{max} preceding cars. The structure of the pheromone trail matrix remains unchanged, only its use is modified, as well as the steps of both local and global pheromone trail updates. Note that the summation of accumulated pheromone trail was previously introduced by Merkle and Middendorf [19,20].

The use of the pheromone trail in the transition rule (Eq. (1)) is modified in order to reflect the benefit of having a candidate class j in position *pos*, near the last classes of cars in the sequence. As defined in Eq. (4) and illustrated in Fig. 3, a sum of the accumulated pheromone trails between the candidate class j and each of the

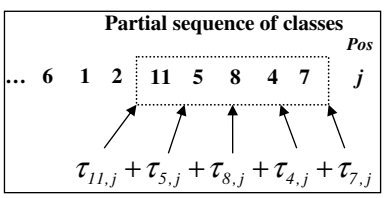

Fig. 3. Computing the pheromone trail for insertion of a vehicle of class j in position *pos* (ACS-H).

classes i of the preceding $s_{\max}$ positions is now used. In the example in Fig. 3, $s_{\max}$ is equal to 5. This sum brought to the exponent α replaces the corresponding term $[\tau_{ij}]^{\alpha}$ in the transition rule presented before

$$\left[\sum_{y=pos-s_{\max}}^{pos-1} \tau_{ipos-y,j}\right]^{\alpha}. \tag{4}$$

Once a class of car has been chosen using the transition rule, a local update of the pheromone trail is carried out, according to Eq. (2′), between class j placed in position pos and each of the classes i of the $s_{\max}$ preceding positions. This modification involves a total of $s_{\max}$ local updates of the pheromone trail.

For each class i in position y between $(pos - s_{\max})$ and $(pos - 1)$

$$\tau_{ij} = \rho_{\ell} \cdot \tau_{ij} + (1 - \rho_{\ell}) \cdot \tau_0. \tag{2′}$$

As for ACS-2D, the best solution of the cycle (S^{Bc}) contributes to global update of the pheromone trail at the end of a cycle. As well, the global update of the pheromone trail is now made between a class of car i at a given position y and each of the classes j at the $s_{\max}$ following positions, according to Eq. (3′).

Evaporation on all the elements of the matrix τ according to ρ_g

For each class i in position y of the solution S^{Bc}

For each class j in position y' between $(y+1)$ and $(y+s_{\max})$

$$\tau_{ij} = \tau_{ij} + (1 - \rho_g) \cdot \tfrac{1}{L^{Bc}}. \tag{3′}$$

This modified version of the ACS-2D using a sum of the pheromone trails on a s_{max} positions horizon is called ACS-H afterwards. As stated in Section 4.3, ACS-H does not produce very satisfying results but the concept of using a range of positions is further explored in the following section.

4.2. Three dimensional pheromone trail matrix

The next element that will be explored is the impact of the number of positions separating the class of the cars to insert into the sequence and the classes of the preceding vehicles. Again, due to the dispersion constraints r_o/s_o of the options composing the classes of cars, there may be a benefit in placing two classes at a certain distance from each other, while it would be totally unfavorable to put them close to each other. This explains why an additional dimension has been added to the pheromone trail matrix τ in order to distinguish the benefit of having two cars of different classes close to each other, as a function of the distance between them. The pheromone trail is then contained in a tridimensional matrix of size $v \times v \times s_{\max}$, where $\tau'_{i,j,d}$ is the advantage of having cars of class i and j at a distance of d positions from each other. This new pheromone trail structure is called 3D and its impacts on the algorithm's behavior are explained in the following paragraphs. These concern the use of τ' in the transition rule and both the local and global pheromone trail updates. The use of the pheromone trail in the transition rule (Eq. (1)) is modified in a way to reflect the benefit of placing a car of candidate class j in position pos near the classes of the last cars in the sequence. As defined in Eq. (5) and shown in Fig. 4, a sum of the accumulated pheromone trails between this candidate class j and each of the classes i of the previous $s_{\max}$ cars is used, as in the ACS-H. But this time, the distance between the cars is taken into account. This sum raised to the power α replaces the term $[\tau_{ij}]^{\alpha}$ in the transition rule of the ACS-2D.

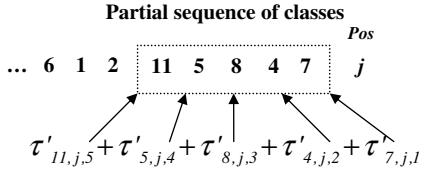

Fig. 4. Computing the 3D pheromone trail for insertion of a vehicle of class j in position pos (ACS-3D).

$$\left[\sum_{y=pos-s_{\max}}^{pos-1} \tau'_{i,j,pos-y}\right]^{\alpha}. \tag{5}$$

The local pheromone trail update is carried out according to Eq. (2″), between the class j in position pos and each of the classes i of the $s_{\max}$ preceding positions, considering explicitly the distance between them. As in ACS-H, this modification involves a total of $s_{\max}$ local updates of the trail.

For each class i in position y between $(pos - s_{\max})$ and $(pos - 1)$

$$\tau'_{i,j,pos-y} = \rho_{\ell} \cdot \tau'_{i,j,pos-y} + (1 - \rho_{\ell}) \cdot \tau_0. \tag{2″}$$

The global update of the 3D pheromone trail is also made between a class of car i at a given position y and each of the classes j of the next $s_{\max}$ positions, considering explicitly the distance between classes, according to Eq. (3″).

Evaporation of all the elements of the matrix τ' according to ρ_g

For each class i in position y of the solution S^{Bc}

For each class j in position y' between $(y+1)$ and $(y+s_{\max})$

$$\tau'_{i,j,y'-y} = \tau'_{i,j,y'-y} + (1 - \rho_g) \cdot \tfrac{1}{L^{Bc}}. \tag{3″}$$

In the next section, the results on the performance of the ACS-H and the ACS-3D in relation to those of the ACS-2D are presented.

4.3. Experimentation and results

The performance of the ACS-H and ACS-3D is evaluated using three sets of benchmark problems available on the Internet. Those are the same problems used by Gravel et al. [16] to demonstrate the performance of the ACS-2D. The first set (SET1) [21] consists of 70 instances, each with 200 cars to order, 5 options and from 17 to 30 classes of cars. Notice that there is a conflict-free solution for each of the problems. The number of cars to sequence is not the only way to determine the level difficulty of a problem, as some instances with fewer cars may be harder to solve. This is the case of the second set of problems (SET2) proposed by Gent and Walsh [22] which consists of 9 instances of 100 cars, 5 options and between 19 and 26 classes. Notice that for 5 of those 9 instances, no conflict-free solution is known. Finally, the third set (SET3), proposed by Gravel et al. [16] contains 30 difficult instances of greater size (200, 300, and 400 cars), with the same characteristics as those of SET2.

Results presented in this paper were obtained using a Xeon 3.6 GHz computer, running Windows XP Pro, with 1024 Mb of RAM. It is, in fact, the same machine that was used to obtain the results of the ACS-2D [16]. The algorithms are coded in C/C++ language, with Microsoft's Visual C++ .NET compiler. Each problem has been solved 100 times by each algorithm.

The values of the parameters of the ACS-H are the same as the ACS-2D, whereas numerical experiments have shown that the exponent α should be increased in the ACS-3D for better performance. This parameter allows the control of the relative importance given to the pheromone trail when choosing the classes of the cars in the transition rule. Indeed, numerical experiments show that the best results were produced when increasing α from 1 to 4.

This is an interesting fact since recent literature tends to ignore the trail power parameter by giving it a value of one. Our results thus tend to put this trend in question.

Results obtained by ACS-2D, ACS-H and ACS-3D on SET1 are not formally presented since the three algorithms produced conflict-free solutions for all 70 instances in less than 1 second. These problems are therefore unsuitable for comparing the real performances of three algorithms.

Table 1 shows results obtained by each of the three algorithms, on all nine SET2 instances. The columns in this table are, in order, problem name, number of conflicts for the best known solution, and for each algorithm, mean number of conflicts, standard deviation and mean number of cycles required to find the best solution. The statistics were obtained from 100 trials. Notice that ACS-3D allows an improvement of the mean number of conflicts on four problems (10_93, 19_71, 21_90 and 36_92), that its performance is equivalent to ACS-2D in two other cases and that all three algorithms produce the same results on three cases. We ignore the trivial difference in results for problem 4-72. ACS-2D proved to be more efficient only for problem 16-81, when ACS-H seems truly less efficient than the two others. Best results are identified with grey shading in Table 1.

From among 100 trials, the three algorithms are regularly able to find a solution as good as the known solution having the least number of conflicts, except for problem 10–93. In this case, ACS-3D manages to find best known solution (with 3 conflicts) 18 times, compared to only twice for ACS-2D, whereas ACS-H never achieves it. One may also notice that all algorithms produce a low variability in results. When we look at the mean cycle at which the best solution of the run is found, the three algorithms converge

Table 1
Average results obtained by ACS-2D, ACS-H and ACS-3D on SET2

Problem	Best known solution	ACS-2D		ACS-H		ACS-3D	
		Mean conflicts (Std. dev.)	Mean cycle	Mean conflicts (Std. dev.)	Mean cycle	Mean conflicts (Std. dev.)	Mean cycle
10_93	3	4.24 (0.47)	298.93	5.33 (0.55)	306.20	4.03 (0.74)	284.78
16_81	0	0.12 (0.33)	329.10	0.94 (0.31)	280.04	0.58 (0.59)	234.01
19_71 *	2	2.08 (0.27)	309.48	2.93 (0.26)	209.82	2.04 (0.20)	217.74
21_90	2	2.63 (0.49)	324.39	2.98 (0.14)	183.08	2.02 (0.14)	179.90
26_82 *	0	0.00 (0.00)	83.72	0.86 (0.35)	84.15	0.00 (0.00)	40.68
36_92	2	2.29 (0.46)	346.02	3.35 (0.52)	357.92	2.03 (0.17)	139.84
4_72 *	0	0.00 (0.00)	58.72	0.09 (0.29)	279.40	0.01 (0.10)	69.15
41_66 *	0	0.00 (0.00)	5.84	0.00 (0.00)	12.62	0.00 (0.00)	4.34
6_76 *	6	6.00 (0.00)	1.01	6.00 (0.00)	1.01	6.00 (0.00)	1.00

Table 2
Average results obtained by ACS-2D, ACS-H and ACS-3D on SET3

Problem	Best known solution	ACS-2D		ACS-H		ACS-3D	
		Mean conflicts (Std. dev.)	Mean cycle	Mean conflicts (Std. dev.)	Mean cycle	Mean conflicts (Std. dev.)	Mean cycle
200_01	0	3.80 (0.62)	293.44	8.38 (0.86)	293.44	2.00 (0.62)	394.76
200_02	2	4.14 (0.51)	422.62	4.85 (0.36)	422.62	2.38 (0.49)	332.53
200_03	3	8.90 (0.59)	350.44	12.36 (0.73)	350.44	7.45 (0.59)	370.86
200_04	7	9.86 (0.51)	274.59	12.38 (0.63)	274.59	7.87 (0.34)	196.39
200_05	6	8.81 (0.42)	201.67	10.10 (0.46)	201.67	7.29 (0.46)	313.68
200_06	6	6.87 (0.39)	232.09	8.15 (0.54)	232.09	6.03 (0.17)	261.79
200_07	0	2.99 (0.36)	344.48	7.18 (0.80)	344.48	0.67 (0.53)	422.98
200_08 *	8	8.00 (0.00)	51.06	8.01 (0.10)	51.06	8.00 (0.00)	21.91
200_09	10	11.85 (0.48)	284.31	12.73 (0.45)	284.31	10.97 (0.17)	95.00
200_10	19	21.44 (0.57)	313.96	25.48 (0.73)	313.96	20.19 (0.61)	341.63
300_01	0	5.33 (0.71)	386.80	8.06 (0.76)	386.80	3.89 (0.57)	392.36
300_02	12	13.15 (0.39)	307.19	17.76 (0.92)	307.19	12.57 (0.50)	230.48
300_03	13	14.54 (0.54)	370.63	20.50 (0.93)	370.63	13.85 (0.36)	213.29
300_04	7	10.33 (0.68)	390.43	12.49 (0.73)	390.43	8.69 (0.51)	327.08
300_05	27	40.55 (1.06)	371.53	48.83 (1.04)	371.53	42.54 (1.07)	406.08
300_06	2	7.59 (0.74)	255.48	11.05 (0.73)	255.48	5.79 (0.56)	406.44
300_07	0	2.89 (0.63)	173.35	7.45 (0.90)	173.35	0.97 (0.50)	417.38
300_08 *	8	9.17 (0.38)	229.22	14.48 (1.37)	229.22	8.95 (0.22)	105.82
300_09	7	9.05 (0.56)	274.34	11.19 (0.81)	274.34	8.00 (0.20)	297.19
300_10	21	34.63 (1.04)	237.32	38.29 (1.11)	237.32	32.56 (1.15)	427.52
400_01	1	3.01 (0.56)	201.01	4.44 (0.56)	201.01	3.50 (0.64)	280.90
400_02	15	23.28 (0.82)	280.48	27.30 (1.14)	280.48	23.82 (0.81)	360.76
400_03	9	11.65 (0.50)	212.39	12.56 (0.50)	212.39	13.64 (0.64)	269.08
400_04	19	21.96 (0.75)	454.46	28.82 (1.22)	454.46	20.38 (0.51)	343.25
400_05	0	3.48 (0.96)	346.67	4.93 (0.95)	346.67	2.68 (0.76)	485.32
400_06	0	4.20 (0.97)	263.12	6.44 (0.92)	263.12	1.53 (0.59)	348.61
400_07	4	7.65 (0.83)	260.60	8.48 (0.83)	260.60	8.68 (0.94)	346.20
400_08	4	11.54 (1.62)	179.93	12.42 (1.16)	179.93	12.67 (1.50)	83.66
400_09	5	17.98 (1.10)	209.19	23.68 (1.38)	209.19	16.01 (1.17)	430.45
400_10	0	4.24 (0.99)	77.19	7.83 (0.77)	77.19	2.66 (0.67)	425.60

quite quickly and do not use all 1000 available cycles. However, ACS-3D has a slightly faster convergence. Calculation times remain around 6 seconds on average for all algorithms.

Using the same format, Table 2 contains results obtained by the three algorithms on each of the 30 instances of SET3. Best results are identified by grey shading. We notice that ACS-3D gets best results as measured by mean number of conflicts in 23 cases and ties in one other. For its part, ACS-2D gives the best performance in 6 cases, which are mainly from among the larger problems (400 cars). Looking at the best known solutions, we notice that this set of benchmark problems is harder to solve and encourages us to seek further improvements that may eventually be included in ACS-3D. Nevertheless, we observe that in this problem set, ACS-H surpasses ACS-3D on problems 400-03, 400-07 and 400-08, although, in most of the cases, results produced by ACS-H are significantly different from those of the two other algorithms. When looking at the mean cycle where the best solution of the run is found, we notice this time that ACS-2D and ACS-H found their best solution faster than ACS-3D on average (275 cycles for ACS-2D compared to 311 for ACS-H). Once again, a fast convergence of the three algorithms is observed. Concerning computation times, all algorithms carry out their 1000 cycles in about 14 seconds, 22 seconds and 29 seconds for instances of sizes 200, 300, and 400, respectively.

ACS-H seems considerably less efficient compared to the two other algorithms and a statistical test has been carried out to determine if ACS-3D is really more efficient than ACS-2D. When both algorithms obtain many identical results, due to the low variability, it is difficult to use a statistical test. However, the results of Conover and Iman [23] show how to use a parametrical t-test on mean rank with cases where we get less than 80% of identical results. In that way, 7 cases identified by an asterisk (*) in Tables 1 and 2 have been removed to perform the test. Table 3 presents the statistical test results for all 32 remaining instances. These results allow the confirmation of the conclusion that, for a level of confidence of 99%, ACS-3D produces solutions of better quality than ACS-2D.

Other statistical tests have been carried out by grouping problems of SET3 by size (200, 300, and 400 vehicles). Results from those tests are presented in Table 4 and also favor ACS-3D, in all cases, even for those with 400 cars, where ACS-2D gets better conflict counts than ACS-3D for 5 cases.

Table 3
t-test for comparison of mean ranks between ACS-2D and ACS-3D for the 32 problems considered: two samples with different variances

	ACS-2D		ACS-3D
Mean rank	124.96		76.04
Variance	2313.64		2101.39
Number of cases		3200	
t-value		41.65	
2-tail probability		0.000	

The significance level for the test is 1%.

Table 4
t-test for comparison of mean ranks between ACS-2D and ACS-3D for each group of problem set SET3: two samples with different variances

	200 vehicles		300 vehicles		400 vehicles	
	ACS-2D	ACS-3D	ACS-2D	ACS-3D	ACS-2D	ACS-3D
Mean rank	145.76	55.24	131.85	69.15	106.46	94.54
Variance	955.32	687.09	2018.61	1718.48	3011.80	2998.77
Number of cases	900		900		1000	
t value	67.01		30.77		4.86*	
2-tail probability	0.000		0.000		0.000	

The significance level for the test is 1%.
* t-test with equal variances.

Table 5
Average results obtained by ACS-3D with local search on SET2 and SET3

Problem	Best known solution	ACS-3D with local search	
		Mean conflicts (Std. dev.)	Mean cycle
10_93	3	3.37 (0.49)	459.60
16_81	0	0.03 (0.17)	240.92
19_71	2	2.00 (0.00)	96.15
21_90	2	2.00 (0.00)	84.43
26_82	0	0.00 (0.00)	41.80
36_92	2	2.00 (0.00)	67.53
4_72	0	0.00 (0.00)	39.65
41_66	0	0.00 (0.00)	4.48
6_76	6	6.00 (0.00)	1.00
200_01	0	0.41 (0.53)	780.89
200_02	2	2.00 (0.00)	271.17
200_03	3	5.76 (0.88)	929.41
200_04	7	7.00 (0.00)	505.18
200_05	6	6.16 (0.37)	651.20
200_06	6	6.00 (0.00)	87.84
200_07	0	0.00 (0.00)	186.94
200_08	8	8.00 (0.00)	22.47
200_09	10	10.00 (0.00)	367.76
200_10	19	19.02 (0.14)	667.81
300_01	0	1.87 (0.72)	875.07
300_02	12	12.01 (0.10)	396.55
300_03	13	13.02 (0.14)	517.63
300_04	7	7.70 (0.69)	854.32
300_05	27	32.53 (1.18)	1001.00
300_06	2	3.59 (0.89)	958.70
300_07	0	0.08 (0.27)	588.29
300_08	8	8.00 (0.00)	407.90
300_09	7	7.18 (0.39)	693.35
300_10	21	22.25 (0.76)	1001.00
400_01	1	2.55 (0.58)	876.79
400_02	15	17.46 (0.95)	1001.00
400_03	9	10.08 (0.39)	1001.00
400_04	19	19.01 (0.10)	914.80
400_05	0	0.02 (0.14)	984.15
400_06	0	0.10 (0.30)	701.56
400_07	4	5.58 (0.77)	1001.00
400_08	4	5.24 (0.88)	1001.00
400_09	5	7.31 (0.94)	1001.00
400_10	0	0.89 (0.67)	859.85

To complete the analysis of the performance of the ACS-3D algorithm, the results of Table 5 show the application of a local search method to the best solution found at each cycle as well as to the best overall solution. The local search procedure applied to the best solution of a cycle (S^{Bc}) is to reverse the order of a sub-sequence. First, one randomly selects the position of a car forming part of sequence of length s_o containing a conflict for some option. A second position is randomly chosen among all positions and the sub-sequence including and linking these two positions is inversed. A similar approach called Lin2Opt is described by Putcha and Gottlieb [11] except that the two positions are randomly chosen among all positions. If the solution is improved, it is retained as the best current solution. The procedure is repeated ($2 * nc$) times or is halted if a conflict-free solution is found. The local search procedure applied to the best overall solution (S^{Bg}) is the Lin2Opt procedure of Putcha & Gottlieb [11] and it is repeated ($2000 * nc$) times or is halted if a conflict-free solution is found.

The results of Table 5 show that the performance of the ACS-3D algorithm with local search is improved but the best known solutions were not attained at each trial. We note that the best solution is generally found after a larger number of cycles than was the case for the problems of Tables 1 and 2. When the local search procedure is applied to the best overall solution found by the ACS-3D, it very often finds the best known solution for the problem.

5. Conclusions

In this paper, the nature of the *car-sequencing* problem allowed the proposal of a new specialized pheromone trail structure. The dispersion constraints found in the current problem inspired the refinement of the ACS-2D algorithm learning process. The efficiency of the new pheromone trail structure in solving the *car-sequencing* problem has been demonstrated by numerical results on three sets of benchmark problems. The quality of the obtained results is superior to that of the best published results and clearly shows the efficiency of the specialized trail. Yet, in order to yield its best performance, the value of the parameter α controlling its relative importance had to be augmented. This goes against the current trend in the literature that is to assign a value of one or simply to eliminate the parameter. This new pheromone trail may also apply to different problems, particularly to other sequencing problems. For example, in the sequencing of production orders with the objective of minimizing total tardiness, the position of an order in relation to the others may be taken into account in the learning process.

The results of this paper should therefore encourage researchers to further consider that essential element in ACO algorithm implementation. Based on the best known solutions and the results obtained, the third set of benchmark problems suggests that we consider future improvements to the ACS-3D. Among other things, new visibility elements and dynamic adjustment of the different metaheuristic parameters would allow a better coverage of the solution space and a more efficient use of the available cycles.

References

[1] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Technical Report No. 91-016, Politecnico di Milano, Italy, 1991, p. 20.

[2] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in: F. Verela, P. Bourgine (Eds.), Proceedings of the First European Conference on Artificial Life (ECAL'91), MIT Press, Cambridge, Mass, USA, 1992.

[3] M. Dorigo, Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[4] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004. p. 328.

[5] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.

[6] B.D. Parello, W.B. Kabat, L. Wos, Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem, Journal of Automated Reasoning 2 (1986) 1–42.

[7] B.D. Parello, CAR WARS: The (almost) birth of an expert system, AI Expert, vol. 3, 1988. pp. 60–64.

[8] T. Kis, On the complexity of the car-sequencing problem, Operations Research Letters 32 (4) (2004) 331–336.

[9] P. Lopez, F. Roubellat, Ordonnancement de la Production, Hermès Science Publications, Paris, 2001.

[10] A. Davenport, E. Tsang, Solving constraint satisfaction sequencing problems by iterative repair, in: Proceedings of the First International Conference on the practical Applications of Constraint Technologies and Logic Programming, Practical Applications Company, London, England, 1999, pp. 345–357.

[11] M. Puchta, J. Gottlieb, Solving car-sequencing problems by local optimization, in: S. Cagnoni, J. Gotlieb, E. Hart, M. Middendorf, G.R. Raidl (Eds.), Proceedings EvoWorkshops, Lecture Notes in Computer Science, vol. 2279, Springer, Kinsale Ireland, 2002, pp. 132–142.

[12] J. Gottlieb, M. Puchta, C. Solnon, A study of greedy local search and ant colony optimization approaches for car-sequencing problems, in: G.R. Raidl et al. (Eds.), Applications of Evolutionary Computing Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 2003, pp. 246–257.

[13] T.L. Chew, J.M. David, A. Nguyen, Y. Tourbier, Solving constraint satisfaction problem with simulated annealing: The car-sequencing problem revisited, in: 12th International Conference on Artificial Intelligence, Expert Systems and Natural Language, 1991, pp. 405–416.

[14] T. Warwick, E.P.K. Tsang, Tackling car-sequencing problem using a generic genetic algorithm, Evolutionary Computation, vol. 3, No. 3, MIT Press, 1995. pp. 267–298.

[15] K. Smith, M. Palaniswami, M. Krishnamoorthy, Traditional heuristic versus Hopfield neural network approaches to a car-sequencing problem, European Journal of Operational Research 93 (2) (1996) 300–317.

[16] M. Gravel, C. Gagné, W.L. Price, Review and comparison of three methods for the solution of the car-sequencing problem, Journal of the Operational Research Society 56 (11) (2005) 1287–1295.

[17] C. Gagné, M. Gravel, W.L. Price, Solving real car-sequencing problems with ant colony optimization, European Journal of Operational Research 174 (3) (2005) 1427–1448.

[18] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1997) 53–66.

[19] D. Merkle, M. Middendorf, An ant algorithm with a new pheromone evaluation rule for the total tardiness problem, in: E.J.-W. Boers (Ed.), Lecture Notes in Computer Science, vol. 2037, Springer-Verlag, 1999, pp. 287–296.

[20] D. Merkle, M. Middendorf, H. Schmeck, Ant colony optimization for resource constrained project scheduling, IEEE Transactions on Evolutionary Computation 6 (2002) 333–346.

[21] J. Lee, H. Leung, H. Won, Performance of a comprehensive and efficient constraint library using local search, in: G. Antoniou, J.K. Stanley (Eds.), 11th Australian Joint Conference on Artificial Intelligence, Springer-Verlag, Heidelberg Brisbane, Australia, 1998, pp. 191–202.

[22] I.P. Gent, T. Walsh, CSPLib: A benchmark library for constraints, in Research Reports of the APES Group, APES-09-1999, 1999, available from <http://4c.ucc.ie/~tw/csplib/schedule.html>.

[23] W.J. Conover, R.L. Iman, Rank transformations as a bridge between parametric and nonparametric statistics, The American Statistician 35 (1981) 124–129.