A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System [1]

Oscar Cordón[1], Iñaki Fernández de Viana[2], Francisco Herrera[1], Llanos Moreno[3]

[1] *Dept. of Computer Science and Artificial Intelligence. E.T.S.I. Informática. University of Granada. Avda. Andalucía, 38. 18071 - Granada. Spain*

[2] *E.T.S.I. Informática. University of Granada. Avda. Andalucía, 38. 18071 - Granada. Spain*

[3] *Infotel Información y Telecomunicaciones SA. c/ San Antón, 72, 5ª planta, oficina 16. 18005 - Granada. Spain*

[1]{ocordon,herrera}@decsai.ugr.es, [2]ijfviana@navegalia.com, [3]llanos@infotel.es

Abstract

In this contribution, a new ACO model, the *Best-Worst Ant System*, will be proposed which is based on Evolutionary Computation concepts. Its performance will be analyzed when solving of different instances of the traveling salesman problem and will be compared to two existing previous models, the Ant System and the Ant Colony System.

1 Introduction

In the last few years, Ant Colony Optimization (ACO) [6] has appeared as a new bio-inspired meta-heuristic to solve many complex optimization problems. ACO algorithms mimic the behavior of natural ant colonies. They are thus based on the cooperation among multiple agents, ants, every one generating a possible solution to the problem in each algorithm iteration. To do so, each ant travels a graph which represents a specific problem instance and makes use of two infomation types that are common to the whole colony and specify the preference of the graph edges at every moment:

- *Heuristic information*, which depends on the specific problem instance, is computed before running the algorithm and remains fixed during it. The value associated to each edge (i, j) is noted by η_{ij}.
- *Pheromone trail information*, which is modified during the algorithm run and depends on the number of ants that travelled each edge in the past and on the quality of the solutions they generated. It is usually represented in the form of a pheromone matrix, $\tau = [\tau_{ij}]$, which mimics the real pheromona that natural ants deposit in their movements.

In view of the previous behavior, and as already noted in [6], it can be seen that there are some similarities between the operation mode of ACO algorithms and the Population-Based Incremental Learning (PBIL) [1] evolutionary algorithm (EA) due to the following reasons [3]:

- They both make use of a memoristic structure that undergoes adaption.
- This structure allows possible solutions to the problem to be generated in each iteration and its adaption is guided by the quality of these solutions.

[1]This research has been supported by project CICYT PB98-1319

These similarities were also analyzed in deep by Monmarché et al. in [8], who characterized a common framework, which they called *Probabilistic Search Meta-heuristic*, where a third model, Bit-Simulated Crossover, is included as well.

The said common characteristics were the ones that motivated our idea that the integration of some specific aspects of PBIL, in particular, and of other EAs, in general, could improve the performance of ACO [3] (in fact, two ACO models, AS_{elite} and AS_{rank}, based on this idea are to be found in [2]). The new ACO model so developed will be called *Best-Worst Ant System* ($BWAS$) and will be introduced in this work.

To do so, once analyzed the basic operation mode of ACO algorithms and described some specific models in Section 2, Section 3 will be devoted to briefly introduce the PBIL algorithm. Later, our new ACO model will be described in Section 4 and its performance when solving of some TSP instances will be analyzed and compared with other two ACO algorithms. Finally, some concluding remarks and future works will be mentioned in Section 6.

2 Ant Colony Optimization

The basic operation mode of an ACO algorithm is the following: in each iteration, a population of m ants gradually and concurrently build solutions to the problem according to a *transition rule* which depends on the heuristic and pheromone trail information available. Ants can release pheromone while building the solutions (*online step-by-step pheromone trail updating*), once they have been generated and evaluated (*online delayed pheromone trail updating*) —positively reinforcing the edges travelled with an amount of pheromone directly dependent on the solution quality— or both. Then, all the pheromone trails suffer from evaporation.

Moreover, some *daemon actions* can be performed from a global perspective, such as observing the quality of all the solutions generated and updating an additional pheromone trail only in some of them, or applying a local search procedure to the solutions generated by the ants and depositing additional pheromone. In both cases, the daemon replaces the online delayed pheromone updating and the process is called *offline pheromone trail updating.*

A simplified structure of a generic ACO algorithm for static combinatorial optimization problems is shown as follows:

```
1. Give an initial pheromone value, $\tau_0$, to each edge.
2. For k=1 to m do (in parallel)
    • Place ant k in an initial node r.
    • Include r in $L_k$ (tabu list of ant k keeping a record of the visited nodes).
    • While (ant k not in a target node) do
        – Select the next node to visit, $s \notin L_k$, according to the transition rule.
        – Include s in $L_k$.
        – Optional: Online step-by-step updating of the pheromone trail $\tau_{rs}$ of the travelled
          edge.
```

3. **Optional:** *For k=1 to m do*
   - *Evaluate the solution generated by ant k, S_k.*
   - *For each edge $(r,s) \in S_k$, apply the online delayed pheromone trail updating rule.*
4. *Evaporate pheromone.*
5. **Optional:** *Perform the daemon actions.*
6. *If (Stop Condition is satisfied) Then give the global best solution found as output and Stop Else go to step 2.*

In particular, the two first ACO models, *Ant System* (*AS*) [4] and *Ant Colony System* (*ACS*) [5], implement the algorithm components as follows:

AS:

- *Transition rule*: The destination node s for an ant k located in node r is randomly chosen according to the following probability distribution

$$p_k(r,s) = \begin{cases} \frac{[\tau_{rs}]^\alpha \cdot [\eta_{rs}]^\beta}{\sum_{u \in J_k(r)} [\tau_{ru}]^\alpha \cdot [\eta_{ru}]^\beta}, & \text{if } s \in J_k(r) \\ 0, & \text{otherwise} \end{cases} ,$$

  with τ_{rs} being the pheromone trail of edge (r,s), η_{rs} being the heuristic value (for example, in the TSP , $\eta_{rs} = \frac{1}{l_{rs}}$, where l_{rs} is the length of edge (r,s)), $J_k(r)$ being the set of nodes that remain to be visited by ant k, and with α and β being parameters weighting the relative importance of pheromone trail and heuristic information.

- *Online delayed pheromone updating rule*: It is developed by means of the expression

$$\tau_{rs} \leftarrow (1-\rho) \cdot \tau_{rs} + \sum_{k=1}^{m} \Delta\tau_{rs}^k \qquad \text{where} \qquad \Delta\tau_{rs}^k = \begin{cases} f(C(S_k)), & \text{if } (r,s) \in S_k \\ 0, & \text{otherwise} \end{cases} ,$$

  with $\rho \in [0,1]$ being the pheromone decay parameter and $f(C(S_k))$ being the amount of pheromone to be deposited by ant k, which depends on the quality of the solution it generated, $C(S_k)$ (for example, in the TSP, $f(C(S_k)) = \frac{1}{C(S_k)}$).

  It is noteworthy that, for practical purposes, the rule includes the evaporation of an $(1-\rho)$ per one of the pheromone trail (step 4 of the algorithm).

ACS:

- *Transition rule*: The destination node s is chosen as follows:

$$s = \begin{cases} arg \max\limits_{u \in J_k(r)} \{[\tau_{ru}]^\alpha \cdot [\eta_{ru}]^\beta\}, & \text{if } q < q_0 \\ S, & \text{otherwise} \end{cases} ,$$

  with q being a random value uniformly distributed in [0,1], $q_0 \in [0,1]$ being a parameter defining the balance explotaition-biased exploration, and with S being a random node selected according to the probability distribution given by the AS transition rule.

- *Online step-by-step updating rule*: Each time an ant travels an edge, it is made in the way:

$$\tau_{rs} \leftarrow (1-\rho)\cdot\tau_{rs} + \rho\cdot\Delta\tau_{rs}$$

  In this paper we consider $\Delta\tau_{ij} = \tau_0$, thus dealing with the simple ACS.

- *Offline pheromone updating*: In this case, the deposit of pheromone is done by the *daemon* only considering a single ant, the one who generated the global best solution, $S_{global-best}$:

$$\tau_{rs} \leftarrow (1-\rho)\cdot\tau_{rs}+\rho\cdot\Delta\tau_{rs} \qquad \text{where} \qquad \Delta\tau_{rs} = \begin{cases} f(C(S_{global-best})), & \text{if } (r,s)\in S_{global-best} \\ 0, & \text{otherwise} \end{cases}$$

For a review of other ACO algorithms refer to [6].

3 The PBIL algorithm

PBIL [1] takes a memoristic structure, a probability array $P = (p_1, \ldots, p_n)$ of dimension n equal to the number of problem variables, as a base. This array encodes a probability distribution representing a prototype for good quality solutions and is used to generate a population of possible solutions (binary arrays) in each iteration.

The probability array is the component that undergoes adaption during the algorithm run according to its history. In the PBIL basic model, the array is updated according to the quality of the best solution generated in the current iteration, *Best-solution*, as follows:

$$p_i = (1-LR)\cdot p_i + LR\cdot Best-solution_i$$

with $LR \in [0,1]$ being a parameter controlling the speed of convergence, the *learning rate*. Moreover, the components of P suffer random mutations with probability $MutProb$ to avoid the possibility of a premature convergence of the algorithm. The mutation will be performed in the following way:

$$p_i = \begin{cases} (1-MutShift)\cdot p_i, & \text{if } a=0, \\ (1-MutShift)\cdot p_i + MutShift, & \text{if } a=1 \end{cases}$$

with a being a random value in $\{0,1\}$ and $MutShift \in [0,1]$ being the mutation step size.

Analyzing the algorithm operation mode, it can be seen the similarity between PBIL and ACO. Besides, the updating rule for P is similar to the offline updating rule in ACS.

An extension of the basic PBIL model involves considering also the worst solution in the current population, *Worst-solution*, for the updating:

$$p_i = (1-LR_{neg})\cdot p_i + LR_{neg}\cdot Worst-solution_i \quad \text{if } Worst-solution_i \neq Best-solution_i$$

with $LR_{neg} \in [0,1]$ being the *negative learning rate*.

4 The Best-Worst Ant System

The proposed *BWAS* uses the transition rule of *AS* (see Section 2), does not perform on-line pheromone updatings (nor step-by-step neither delayed) and considers the three following daemon actions. The name of the algorithm is a consequence of the first of them:

• **The global best and current worst solutions are considered respectively to perform positive and negative updatings**, as PBIL algorithm does with the probability array. Of course, our aim is to reinforce the edges contained in good solutions and penalize the ones from bad solutions. To do so, the daemon in *BWAS* first apply a *local search procedure* on the different solutions generated by the ants, and offline updates the pheromone trail only considering the global best solution as done in *ACS*:

$$\tau_{rs} \leftarrow (1-\rho)\cdot\tau_{rs} + \Delta\tau_{rs} \qquad \text{where} \qquad \Delta\tau_{rs} = \begin{cases} f(C(S_{global-best})), & \text{if } (r,s) \in S_{global-best} \\ 0, & \text{otherwise} \end{cases}$$

Then, all the edges existing in the worst solution generated in the current iteration, $S_{current-worst}$, that are not present in the global best one are penalized by another decay of the pheromone trail associated —an additional evaporation— performed as follows:

$$\forall (r,s) \in S_{current-worst} \text{ and } (r,s) \notin S_{global-best},\ \tau_{rs} \leftarrow (1-\rho)\cdot\tau_{rs}$$

• *BWAS* **also includes a restart of the search process when it get stuck**, a key characteristic of the CHC EA [7]. In ACO, this fact happens when the pheromone matrix has evolved to a situation where the pheromone trails associated to the edges belonging to the best solutions are very high, whilst the remaining ones are very close to zero (*stagnation*).

We should note that this aspect is not new in the ACO field, since previous models —such us *MMAS* [9]— have considered it previously as a daemon action with different approaches. In our case, we will perform the restart by setting all the pheromone matrix components to τ_0, the initial pheromone value, when the number of edges that are different between the best and the worst solutions generated in the current iteration is lesser than a specific percentage.

• **The pheromone matrix suffers mutations to introduce diversity in the search process**, as done in PBIL with the memoristic structure —the probability array P—. The mutation operator will perform small changes in the earlier search stages and strong ones in the later stages. Hence, it tries to find new space zones where better solutions than the current global best one can be found in these later stages when the ACO algorithm has converged to a specific space zone, thus encouraging the exploration instead of the exploitation. To do so, each component of the pheromone matrix is mutated —with probability P_m— as follows:

$$\tau'_{rs} = \begin{cases} \tau_{rs} + mut(it, \tau_{threshold}), & \text{if } a = 0 \\ \tau_{rs} - mut(it, \tau_{threshold}), & \text{if } a = 1 \end{cases} \qquad \tau_{threshold} = \frac{\sum_{(r,s)\in S_{global-best}} \tau_{rs}}{|S_{global-best}|}$$

with a being a random value in $\{0,1\}$, it being the current iteration, $\tau_{threshold}$ being the average of the pheromone trail in the edges composing the global best solution and with $mut(\cdot)$ being:

$$mut(it, \tau_{threshold}) = \frac{it - it_r}{Nit - it_r}\cdot\sigma\cdot\tau_{threshold}$$

where Nit is the maximum number of iterations of the algorithm and it_r is the last iteration where a restart was performed.

We should note two aspects of the mutation operator proposed:

- The mutation range comes back to its initial value each time a restart is performed. Then, the algorithm starts with a new search from an exploration phase.
- The parameter σ specifies the power of the mutation with respect to the number of iterations developed till the moment. For example, if $\sigma = 4$, the value to add or subtract will reach $\tau_{threshold}$ each time a 25% of the total number of iterations remaining since the last restart have been run.

It is noteworthy that the choice of the components integrated from the Evolutionary Computation (EC) field has been done to obtain an appropriate balance between the exploration and the exploitation of the search space. Our pheromone updating mechanism causes a strong exploitation that allows the algorithm to obtain good solutions efficiently, whilst the pheromone matrix mutation encourages the space exploration and avoids the algorithm stagnation. Moreover, the restart process also allows the algorithm not to get stuck and not to make unnecesary iterations.

5 Experiments and Analysis of Results

The proposed ACO model and the two first proposals of ACO algorithms, AS and ACS, will be used to the solving of eight symmetric TSP instances. The three algorithms will consider the same parameter values, the use of a candidate list of the same size and the same local search procedure, as well as the same seeds for the random number generator. The tour improvement procedure will be the *restricted 2-opt* procedure, where the candidate nodes are selected inside the candidate list and the *don't look bit* structure is considered.

The parameter values considered are shown in Table 1. We should note that these values have not been selected to obtain the best possible results but to get an appropriate balance between accuracy and efficiency for comparison purposes. For example, longer runs of the local search procedure will be needed to obtain better results in the largest instances.

It is noteworthy that the number of iterations shown, 300, is a maximum threshold, since the algorithm will stop if the optimal solution is found before all these iterations are performed. This will influence the run time of the models, thus allowing us to measure the convergence speed of the three algorithms considered.

Each model has been run 15 times in a computer with a Pentium II processor at 266 MHz. The overall results obtained are shown in Table 2, where each column name has the following interpretations: *Best* means the cost of the best solution found in the 15 runs, *Average* collects the average of the costs of the 15 solutions generated, *Dev.* shows the standard deviations, *Error* stands for the percentage difference between the average and the cost of the optimal solution (which is shown in brackets after the instance name), and *Time* shows the average time consumed by the models, measured in seconds. Finally, the last column named *#Rest.* contains the average number of restarts performed by $BWAS$ in the 15 runs.

The computational results in Table 2 show that generally $BWAS$ presents the best performance. It allows us to obtain the best results in all the eight instances but the largest one, *Fl1577*, where

Table 1: Parameter values considered for the ACO models

Parameter	**Value**
Number of ants	$m = 25$
Maximum number of iterations	$Nit = 300$
Number of runs of each algorithm	15
Pheromone updating rules parameter	$\rho = 0.2$
AS offline pheromone rule positive updating	$f(C(S_k)) = \frac{1}{C(S_k)}$
ACS offline pheromone rule positive updating	$f(C(S_{global-best})) = \frac{1}{C(S_{global-best})}$
Transition rule parameters	$\alpha = 1$, $\beta = 2$
ACS transition rule parameters	$q_0 = 0.8$
Initial pheromone amount	$\tau_0 = \frac{\#cities}{C(S_{Greedy})}$
Candidate list size	$cl = 20$
$BWAS$ **parameters**	
Pheromone matrix mutation probability	$P_m = 0.3$
Mutation operator parameter	$\sigma = 4$
Percentage of different edges in the restart condition	5%
Local search procedure parameters	
Number of neighbors generated per iteration	40
Neighbor choice rule	first improvement

ACS outperforms it. We think that this $BWAS$ result could be improved by relaxing the restart checking condition —a difference on the 5% of the total number of edges between the best and worst solutions generated in the current run—, which seems to be excessively restrictive for very large instances. On the other hand, $BWAS$ gets the optimal solution in the five smaller instances, whilst AS and ACS does so only in the four and three cases, respectively.

Focusing on the average performance in the 15 runs developed, the results show that $BWAS$ overcomes again AS in all the eight instances and ACS in seven of them (the only exception is again the instance *Fl1577*, where $BWAS$ is also outperformed by ACS in average). Moreover, in view of the standard deviation values, which are lesser than AS and ACS ones in every case, we can draw that $BWAS$ is a very robust algorithm.

6 Concluding Remarks

In this contribution, $BWAS$ has been proposed, which is a new ACO model based on the integration of EC concepts. Its performance has been analyzed when solving of eight TSP instances of different sizes and it has shown a good behavior in comparison with AS and ACS.

Different ideas for future developments arise: (i) to improve $BWAS$ performance on significantly large instances, (ii) to study the influence of the different algorithm components in isolation and of the appropriate values for the parameters, and (iii) to analyze the consideration of other EC aspects such us the use of a number of the best and worst ants to positive and negatively update the pheromone trails or the weighting of the pheromone amount each ant does depending on its ranking —as done in AS_{elite} and AS_{rank}, respectively—.

Table 2: Results obtained in the different instances

	Eil51 (426)						**Berlin52 (7542)**					
Model	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*
BWAS	426	426	0	0	1.72	3.33	7542	7542	0	0	0.13	0
AS	426	426.93	0.46	0.22	11.41	—	7542	7542	0	0	0.34	—
ACS	429	436.07	4.28	2.36	7.37	—	7542	7716.53	105.68	2.31	6.14	—
	Brazil58 (25395)						**Kroa100 (21282)**					
Model	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*
BWAS	25395	25395	0	0	0.45	0	21282	21285.07	8.09	0.01	6.73	6.2
AS	25395	25395	0	0	0.59	—	21282	21331.27	34.69	0.23	26.61	—
ACS	25395	25395	0	0	0.31	—	21320	21558.67	155.89	1.3	15.73	—
	Gr120 (6942)						**Att532 (27686)**					
Model	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*
BWAS	6942	6950.87	5.24	0.13	20.96	9.53	27842	27988.87	100.82	1.09	119.84	16.13
AS	7031	7088.27	32.98	2.11	34.88	—	29348	29573.27	156.99	6.82	253.91	—
ACS	7057	7210.73	74.84	3.87	22.72	—	28000	28370.20	162.27	2.47	99.97	—
	Rat783 (8806)						**Fl1577 (22249)**					
Model	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*	*Best*	*Average*	*Dev.*	*Error*	*Time*	*#R*
BWAS	8972	9026.27	35.26	2.50	186.92	13.27	22957	23334.53	187.33	4.88	656.64	0
AS	9586	9647.80	45.96	9.56	421.14	—	26063	26944	376.41	21.10	1251.01	—
ACS	9218	9307.33	52.97	5.69	167.25	—	22749	23122.53	260.92	3.93	865.96	—

References

[1] S. Baluja, R. Caruana. Removing the Genetics from the Standard Genetic Algorithm. In A. Prieditis, S. Rusell (Eds.), *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 38-46. Morgan Kaufmann Publishers, 1995.

[2] B. Bullnheimer, R.F. Hartl, C. Strauss. A New Rank Based Version of the Ant System: A Computational Study. *Central European Journal for Ops. Research and Economics*, 7(1):25-38, 1999.

[3] O. Cordón, F. Herrera, L. Moreno. Integración de Conceptos de Computación Evolutiva en un Nuevo Modelo de Colonias de Hormigas (in spanish). In *Actas de la CAEPIA'99. Seminario Especializado sobre Computación Evolutiva*, Vol. II, pp. 98-105. 1999.

[4] M. Dorigo, V. Maniezzo, A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Trans. on Systems, Man, and Cybernetics*, Part B, 26(1):29-41, 1996.

[5] M. Dorigo, L.M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem. *IEEE Transactions on Evolutionary Computation* 1(1):53-66, 1997.

[6] M. Dorigo, G. Di Caro. Ant Algorithms for Discrete Optimization. *Artificial Life* 5(2): 137-172. 1999.

[7] L.J. Eshelman. The CHC Adaptive Search Algorithm: How to Safe Search when Engaging in Nontraditional Genetic Recombination. In G.J.E. Rawlins (Ed.), *Foundations of Genetic Algorithms*, pp. 265-283. Morgan Kaufmann Publishers, 1991.

[8] N. Monmarché, E. Ramat, G. Dromel, M. Slimane, G. Venturini. On the Similarities Between AS, BSC and PBIL: Toward the Birth of a New Meta-Heuristic. Technical Report 215, Ecole d'Ingénieurs en Informatique pour l'Industrie (E3i), Université de Tours, 1999.

[9] T. Stützle, H. Hoos. The MAX-MIN Ant System and Local Search for the Traveling Salesman Problem. In *Proceedings of the Fourth International Conference on Evolutionary Computation (ICEC'97)*, pp. 308-313. IEEE Press, 1997.