Mathware & Soft Computing 9 (2002)

Analysis of the Best-Worst Ant System and its Variants on the TSP

Oscar Cordón, Iñaki Fernández de Viana, Francisco Herrera
Dept. of Computer Science and Artificial Intelligence
University of Granada. 18071 - Granada. Spain
e-mail: {ocordon, herrera}@decsai.ugr.es, ijfviana@teleline.es

Abstract

In this contribution, we will study the influence of the three main components of Best-Worst Ant System: the best-worst pheromone trail update rule, the pheromone trail mutation and the restart. Both the importance of each of them and the fact whether all of them are necessary will be analyzed. The performance of different variants of this algorithm will be tested when solving different instances of the TSP.

1 Introduction

In the last few years, Ant Colony Optimization (ACO) [10] has become a popular metaheuristic for solving complex optimization problems like the classical traveling salesman problem or the routing in telecommunication networks. ACO algorithms mimic the behavior of natural ant colonies, being 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 information types that are common to the whole colony and specify the preference of the graph edges/nodes at every moment:

- *Heuristic information*, which depends on the specific problem instance, is computed before running the algorithm and remains fixed at run time (in static optimization problems). The value associated to each edge (r, s) is denoted by η_{rs}.
- *Pheromone trail information*, which is modified during the algorithm run and depends on the number of ants that traveled 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, $\tau = [\tau_{rs}]$, which mimics the real pheromone that natural ants deposit while moving.

There has been proposed a number of different ACO models [10] like Ant System (*AS*) [8], Ant Colony System (*ACS*) [9], Rank-based Ant System (AS_{rank}) [4], and Max-Min Ant System ($\mathcal{MM}AS$) [15]. In [5, 6], a new variant called Best-Worst Ant System (*BWAS*) was introduced. *BWAS* is characterized by integrating three components from Evolutionary Computation [1]: the best-worst pheromone trail update rule, the pheromone trail mutation and the restart.

In this paper, we extend the preliminary study of the *BWAS* components that was developed in [7] by applying the *BWAS* algorithm and its variants to a large number of TSP instances. Notice that when we use the term variants, we refer to those algorithms obtained from the basic *BWAS* by removing one or two of its three distinguishing components. Hence, our aim is to demonstrate that *BWAS* is an algorithm as a whole, i.e., that a trade-off exists among all its components, and to analyze the relative importance of each of them.

This paper is structured as follows. In Section 2, the basis of the *BWAS* algorithm are introduced. In Section 3, the different *BWAS* variants studied are presented. In Section 4, we will consider the application of the ACO algorithms to the TSP and we will present the results obtained by the *BWAS* family of algorithms for several instances. We end by concluding remarks and proposals for future work in Section 5. Moreover, an appendix with the individual result tables is included.

2 Best-Worst Ant System

After the large development of the ACO metaheuristic, several authors recognized the similarities existing between ACO algorithms and a specific family of evolutionary algorithms guided by probability distribution adaption [13]. This was the starting point of our *BWAS* proposal when we saw that the synergy obtaining from the hybridization of components of these algorithms could result in significant performance improvements. Hence, the *BWAS* model tries to improve ACO performance using evolutionary algorithm concepts [5, 6].

As AS_{rank} and $\mathcal{MM}$AS, *BWAS* constitutes another extension of *AS* since it uses the *AS transition rule* and *pheromone evaporation mechanism* [8].

On the one hand, the transition rule is applied as follows:

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

with τ_{rs} being the pheromone trail of edge (r,s), η_{rs} being the heuristic value, $J_k(r)$ being the set of nodes that remain to be visited by ant k, and with α and β being parameters that weight the relative importance of pheromone trail and heuristic information.

On the other hand, the evaporation mechanism operates by applying the formula:

$$\tau_{rs} \leftarrow (1-\rho) \cdot \tau_{rs}, \ \ \forall\, r,s$$

with $\rho \in [0,1]$ being the pheromone evaporation rate.

Besides, BWAS always considers the systematic exploitation of local optimizers to improve the ants' solutions.

Together with the previous elements, *BWAS* considers the three following daemon actions:

Best-worst pheromone trail update rule

This rule is based on the Population-Based Incremental Learning (PBIL) [2] probability array update rule. The global best solution is considered to perform a positive update of trails, while the pheromone trails associated to the worst solution generated in the current iteration are penalized to reduce their desirability.

To reinforce the edges contained in good solutions, the daemon in *BWAS* first offline updates the pheromone trail by only considering the global best solution:

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

with $f(C(S_{global-best}))$ being the amount of pheromone to be deposited by the global-best ant, which depends on the quality of its solution, $C(S_{global-best})$.

Then, all the edges existing in the current worst solution, $S_{current-worst}$, 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:

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

Pheromone trail mutation

The pheromone trails suffer mutations to introduce diversity in the search process, as done in PBIL with the memoristic structure. To do so, each row of the pheromone matrix is mutated —with probability P_m— by adding or subtracting the same amount of pheromone to the selected trail (a value which depends on the current iteration) as follows:

$$\tau'_{rs} = \begin{cases} \tau_{rs} + mut(it, \tau_{threshold}), & if \; a = 0 \\ \tau_{rs} - mut(it, \tau_{threshold}), & if \; a = 1 \end{cases}$$

with a being a random value in $\{0,1\}$, it being the current iteration, $\tau_{threshold}$ being the average of the pheromone trail on 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.

Notice that the mutation range comes back to its initial value each time a restart is applied and the parameters σ and $\tau_{threshold}$ specify the maximum power of the mutation.

We should mention that the $mut(\cdot)$ function does not prevent pheromone values to be negative. Hence, there is a need to check that they are correct after each application of this component.

Restart of the search process when it gets stuck

This is a key characteristic of the CHC evolutionary algorithm [11] and is also common in ACO, where it happens when the pheromone matrix has reached the stagnation phase.

Other ACO models —such us $\mathcal{MMAS}$ [15]— have previously considered it 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 global-best solution is not improved during a fixed number of iterations.

A simplified structure of a generic *BWAS* algorithm is shown as follows:

1. *Give an initial pheromone value, τ_0, to each edge.*
2. *For k=1 to m do (in parallel)*
   - *Place ant k on 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 AS transition rule.*
     - *Include s in L_k.*
3. *Pheromone evaporation.*
4. *For k=1 to m do*
   - *Evaluate the solution generated by ant k, S_k.*
   - *Local search improvement.*
5. *$S_{global-best} \leftarrow$ global best ant tour. $S_{current-worst} \leftarrow$ current worst ant tour.*
6. *Best-Worst pheromone update.*
7. *Pheromone trail mutation.*
8. *Restart if condition is satisfied.*
9. *If (Termination Condition is satisfied)*
   Then give the global-best solution found as output and Stop
   Else go to step 2.

For more information on *BWAS*, we refer to [6].

3 Analysis of the BWAS Components

As we said, the main objective of this paper is to study the influence of the three components of *BWAS* on its application to the TSP. With this study, we want to know if all of them are really important or some of them can be removed without negatively affecting the performance of the *BWAS* algorithm. Additionaly, we also try to establish a ranking of importance among components.

This analysis will be made from a double perspective:

- Individualized analysis of components, i.e., we will run the *BWAS* algorithm using only one of its components.
- Cooperative analysis among pairs of components. In this case, we will run variants of the *BWAS* algorithm including two of its components.

It seems that a certain interrelation exists among the three basic elements of *BWAS*. The update of pheromone trails by the worst ant allows the algorithm to quickly discard areas of the search space while the mutation and the restart avoid a stagnation of the algorithm. It may seem that the latter two components can be redundant since they both have the same aim but we will see that a high cooperation arises between both.

In Table 1, all the algorithmic variants used in the study are summarized. As can be seen, there are three different groups of algorithms. The first one includes the basic models: our proposal, *BWAS*, and the classical *AS* and *ACS*, which are considered for comparison purposes. The second group comprises variants including a single component: restart, mutation or worst-update. The models AS_{+R} and ACS_{+R} are included in this group by adding the *BWAS* restart to *AS* and *ACS*, respectively. Finally, the third group comprises the variants including a pair of the components. The different variants are denoted by $BWAS_{-*}$ where * stands for the removed component (R, M or W)[1].

4 Experimental Results

In this section, we will review the TSP and how we can apply an ACO model to solve this problem. Then, we will present the experimental results.

4.1 The Traveling Salesman Problem

The traveling salesman problem [3], or TSP for short, can be described as: given a finite number of "cities" along with the cost of traveling between each pair of them, find the cheapest way of visiting all the cities and returning to the starting point.

More formally, it can be represented by a complete weighted graph, $G = (N, A)$, with N being the set of cities and A the set of edges fully connecting the nodes N.

[1] Although the $BWAS_{-M-W}$ one-component variant can seem to be the same algorithm than *ACS* with restart (ACS_{+R}), they both are different as: i) $BWAS_{-M-W}$ applies pheromone evaporation to all edges, while ACS_{+R} only evaporates the pheromone trails of the edges travelled by the global-best ant, and ii) both algorithms consider different transition rules.

Table 1: ACO models studied.

Parameter	Meaning
AS	Ant System
ACS	Ant Colony System
$BWAS$	Best-Worst Ant System
AS_{+R}	Ant System with $BWAS$ restart
ACS_{+R}	Ant Colony System with $BWAS$ restart
$BWAS_{-R-W}$	$BWAS$ without restart and worst ant pheromone update
$BWAS_{-M-W}$	$BWAS$ without mutation and worst ant pheromone update
$BWAS_{-R-M}$	$BWAS$ without restart and mutation
$BWAS_{-R}$	$BWAS$ without restart
$BWAS_{-W}$	$BWAS$ without worst ant pheromone update
$BWAS_{-M}$	$BWAS$ without mutation

Table 2: TSP instances used.

TSP instances	
Eil51	Lin318
Berlin52	Pcb442
Brazil58	Att532
Kroa100	Rat783
Gr120	U1060
D198	Fl1577

Each edge is assigned a value d_{rs}, which is the length of edge $(r, s) \in A$. The TSP is the problem of finding a minimal length Hamiltonian circuit of the graph, where a Hamiltonian circuit is a closed tour visiting exactly once each of the $n = |N|$ cities of G.

All the TSP instances used in our experimentation have been obtained from TSPLIB [14]. As shown in Table 2, we have chosen 12 instances of different sizes in order to perform a fair comparative study.

4.2 Application of the ACO Algorithms Considered to the TSP

Initially, all the pheromone trails are set to $\tau_0 = \frac{1}{C(S_{Greedy})\cdot n}$, with $C(S_{Greedy})$ being the cost of the solution obtained by a greedy algorithm for the TSP, and each of the m ants is placed on a randomly chosen city. An ant constructs a tour as follows. At a city r, the ant chooses a unvisited city s probabilistically, biased by the pheromone trail strength τ_{rs} on the edge between cities r and s and on the heuristic information of that edge (which is a function of the edge length, $\eta_{rs} = \frac{1}{d_{rs}}$). This way, ants prefer to move to a city which is close to the current

one and which is connected by an edge of a high pheromone trail.

After all ants have built their solutions, a local search technique is used [12]. In this paper we will use the 2-opt algorithm. This technique proceeds by systematically testing if the current tour can be improved by replacing two edges. To reduce the run-time of 2-opt, we apply three different techniques:

- Restricting the set of movements which are examined to those contained in the candidate list of the nearest neighbors ordered by distances.
- Considering a fixed radius nearest neighbor search: at least one newly introduced edge has to be shorter that any of the two removed edges.
- Using *don't look bits* associated with each node. Initially, all don't look bits are turned off. If for a node no improving movement can be found, the don't look bit is turned on. In case an edge incident to a node is changed by a movement, the node don't look bit is turned off.

Finally, the pheromone trails are updated.

4.3 Parameter Settings

The ACO models shown in Table 1 have been used to solve the twelve TSP instances selected. The parameter values considered are shown in Table 3, where the parameters of AS and ACS are taken from [9]. For the $BWAS$ model, parameters P_m and σ are taken from [6]. The values of the latter two parameters and of the percentage of iterations without improvement in the restart condition have not been obtained from any previous study, and a deep analysis of the influence of appropriate values for the $BWAS$ parameters is to be done in future work.

Each model has been run 10 times on a 1400 MHz. AMD Athlon processor. The maximum run time depends on the TSP instance size. If n <500 then the maximum run time is 600 seconds. If $500 \leq n < 1000$ then the maximum run time is 1200 seconds. Finally, if n> 1000 the maximum run time is 3600 seconds. It is noteworthy that the time intervals shown, 600–3600 seconds, are a maximum threshold, since the algorithm will stop if the optimal solution is found before the maximum time wastes.

4.4 Analysis of Results

Tables 4 and 5 collect a summary of the obtained results, while the complete tables of results can be consulted in the Appendix.

Table 4 compares the algorithms two by two. Each cell a_{ij} shows the percentage of cases in which algorithm i has outperformed algorithm j. We will say that algorithm i is better than algorithm j for a problem instance p if the error[2] obtained by i for p is smaller than the error obtained by j. Notice that the values in Table 4 are symmetric ($a_{ij} = 100 - a_{ji}$) in all cases but in those where there have been

[2] Error stands for the percentage difference between the average cost obtained in the performed runs and the cost of the best solution known for the instance.

Table 3: Parameter values considered for the ACO models.

Model	**Meaning**
Number of ants	$m = 15$
Maximum run time	$Ntime = 600$ to 3600 *seconds*
Number of runs of each algorithm	10
Pheromone update rules parameter	$\rho = 0.2$
AS offline pheromone rule update	$f(C(S_k)) = \frac{1}{C(S_k)}$
ACS offline pheromone rule update	$f(C(S_{global-best})) = \frac{1}{C(S_{global-best})}$
Transition rule parameters	$\alpha = 1$, $\beta = 2$
ACS transition rule parameter	q_0 =0.98
Initial pheromone amount	$\tau_0 = \frac{1}{C(S_{Greedy}) \cdot n}$
Candidate list size	$cl = 20$
BWAS parameters	
Pheromone matrix mutation probability	$P_m = 0.3$
Mutation operator parameter	$\sigma = 4$
Restart condition	$N_{it} \cdot 0.2$
Local search procedure parameters	
Local search algorithm	2-opt
Number of neighbors generated per iteration	40
Neighbor choice rule	first improvement

draws between the two algorithms (i.e., both algorithms have obtained the same error in any of the twelve instances).

A general classification of the models is shown in Table 5 which summarizes the values of Table 4. While the first column contains the name of the model, the second and third columns collect the number of algorithms compared to which that model has obtained better or worse results, respectively. Notice that both the maximum possible value for each column and the sum of each row is 10, since there are eleven different algorithms in our experimental comparison. To compute the previous values, we consider that an algorithm presents better results than another when the former has outperformed the latter a higher percentage of the times and *viceversa*.

Let us first analyze the results of the $BWAS$ variants. When we apply those variants only including one component ($BWAS_{-W-R}$, $BWAS_{-M-R}$ and $BWAS_{-M-W}$), a very bad performance is obtained in almost all the instances. We can explain this bad behavior as follows. One of the $BWAS$ design goals is to achieve an appropriate balance between exploration and exploitation. Mutation and restart are exploration components, while worst ant update rule is clearly an exploitation component. Using only one of these components, we obtain an algorithm that only encourages either the exploration or the exploitation of the search space, thus not having a good balance between these two main aspects. This bad trade-off is the

Table 4: Pair comparisons between ACO models.

Model	AS	ACS	$BWAS$	AS_{+R}	ACS_{+R}	$BWAS_{-R-W}$	$BWAS_{-M-W}$	$BWAS_{-R-M}$	$BWAS_{-R}$	$BWAS_{-M}$	$BWAS_{-W}$
AS	–	8	0	8	8	66	66	66	0	58	0
ACS	66	–	0	50	33	75	83	75	8	58	8
$BWAS$	75	75	–	75	75	83	83	91	41	83	50
AS_{+R}	66	25	0	–	25	66	66	66	0	58	0
ACS_{+R}	66	41	0	50	–	75	75	75	8	58	8
$BWAS_{-R-W}$	25	16	0	25	16	–	16	33	8	25	8
$BWAS_{-M-W}$	25	8	0	25	16	58	–	75	8	50	8
$BWAS_{-R-M}$	25	16	0	25	16	50	16	–	16	16	16
$BWAS_{-R}$	75	58	0	75	66	75	75	75	–	66	16
$BWAS_{-M}$	25	25	0	25	25	66	41	66	16	–	8
$BWAS_{-W}$	75	66	0	75	66	75	75	75	33	66	–

reason of the poor performance. Anyway, among all one-component $BWAS$ variants, $BWAS_{-M-W}$, based on the use of the restart component, obtains the best performance in eight of the twelve cases. It is outperformed by $BWAS_{-R-W}$ in just two of them and by $BWAS_{-R-M}$ in another one, while the results are the same in the remaining instance. However, it is important to note that the three cases where $BWAS_{-M-W}$ does not get the best result correspond to three of the four larger instances (att532, u1060 and fl1577).

On the other hand, it can be seen that *combining two components is enough to achieve a good performance*. The results obtained are better than those of the AS and ACS algorithms for all the $BWAS$ two-component variants in nine of the twelve instances, and the same performance is obtained in the remaining three. Analyzing the three two-component variants, it can be found two different behaviors

Table 5: ACO models standing.

Model	*Better performance*	*Worse performance*
$BWAS$	10	0
$BWAS_{-W}$	9	1
$BWAS_{-R}$	8	2
ACS_{+R}	7	3
ACS	6	4
AS_{+R}	5	5
AS	4	6
$BWAS_{-M-W}$	3	7
$BWAS_{-M}$	2	8
$BWAS_{-R-W}$	1	9
$BWAS_{-R-W}$	0	10

on them. On the one hand, $BWAS_{-R}$ and $BWAS_{-W}$ outperform *AS* and *ACS* in eight instances and only obtain worse results in one of them, the largest instance fl1577. On the other hand, the remaining variant $BWAS_{-M}$ presents the opposite behavior, as it only outperforms the classic ACO algorithms in three cases (with one of them being the fl1577 instance) and looses in other seven cases. This can be due to the fact that the latter variant does not have a good balance between exploration and exploitation because it does not use the strongest exploration component: the pheromone trail mutation. This way, it presents a poor performance similar to one-component *BWAS* variants. Besides, notice that the inclusion of mutation in a two-component variant make it more robust, what can be drawn in view of the smaller standard deviations associated to the two algorithms of this kind.

As said, the most significant exception to the previous generic behavior is the largest instance, fl1577, where it is better to remove the mutation component than the restart or the worst ant update. We think that this is a consequence of the large and specific search space associated to this instance, which requires of a strong and effective exploitation in order to achieve good performance.

On the other hand, *the best overall results have been obtained using the BWAS algorithm with its three components.* Thus, *BWAS* always achieves better or the same performance than all its variants. Notice that for every problem instance, *BWAS* gets the best error. However, for the two largest ones, u1060 and fl1577, the best individual solution is obtained by another BWAS variant, $BWAS_{-M}$, and by ACS_{+R}, respectively. This could be solved by a better choice of the *BWAS* parameter values (we should remind that no previous experimental study was developed to select appropriate values for them). This task will be part of the further work to be done in the future.

In view of these results, we can conclude that:

- *BWAS* can improve the performance of classical ACO algorithms like *ACS* and *AS*.
- There exists an appropriate trade-off among the three components of *BWAS*, i.e., the algorithm has a good balance between exploration and exploitation. If we remove one or more components, the performance worsens.
- The order of importance among the components seems to be clear. The mutation is the component with the highest influence on the behavior of the algorithm. Between the restart and the worst update, the differences are smaller, and none of them is preferred to the other.

5 Concluding Remarks and Future Works

In this contribution, a study of all the components of *BWAS* has been done. The performance of the resulting algorithms and the importance of their components has been analyzed when solving twelve TSP instances of different sizes. It has been shown that the best performance is obtained when using the full version of the *BWAS* algorithm.

Two main ideas for future developments arise: (i) study the influence of parameter settings on *BWAS* behavior, and (ii) analyze the consideration of other Evolutionary Computation aspects such us the use of a number of the best and worst ants to positively and negatively update the pheromone trails —as done in PBIL [2]— or the weighting of the pheromone amount each ant deposits depending on the ranked quality of its solution —as done in AS_{rank} [4]—.

Appendix: Tables of Results

The overall results obtained are shown in Tables 6 to 9, where each column name stands for the following: *Best* means the cost of the best solution found in the 10 runs, *Average* collects the average of the costs of the 10 solutions generated, *Dev.* shows the *standard* deviations, *Error* stands for the percentage difference between the average and the cost of the best solution known (which is shown in brackets after the instance name). Finally, the last column named $\#R$ contains the average number of restarts performed by the algorithm in the 10 runs developed.

References

[1] T. Bäck, D. Fogel, Z. Michalewicz (Eds.). *Handbook of Evolutionary Computation*, Institute of Physics Publishing, Bristol, 1997.

[2] 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*, Morgan Kaufmann Publishers, pp. 38-46, 1995.

[3] J.L. Bentley. Fast Algorithms for Geometric Travelling Salesman Problem. *ORSA Journal on Computing*, 4(4), pp. 387-411, 1992.

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

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

[6] O. Cordón, F. Herrera, I. Fernández de Viana, L. Moreno. A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System. *Proc. of ANTS'2000. From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, Brussels, Belgium, September 7-9, pp. 22-29, 2000.

[7] O. Cordón, I. Fernández de Viana, F. Herrera. Análisis de las tres componentes que distinguen al Sistema de la Mejor-Peor Hormiga (in Spanish). *Actas*

Table 6: Results obtained in the different instances (I).

	Eil51 (426)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	427	427,5	0,53	0,53	0
ACS	**426**	426,7	0,48	0,48	0
$BWAS$	**426**	426	0	**0**	1
AS_{+R}	**426**	426,2	0,42	0,05	3,9
ACS_{+R}	**426**	426,5	0,53	0,12	2,4
$BWAS_{-R-W}$	429	436,5	4,27	2,40	0
$BWAS_{-M-W}$	**426**	429,3	2,75	0,76	5,5
$BWAS_{-R-M}$	428	433,8	4,15	1,79	0
$BWAS_{-R}$	**426**	426	0	**0**	0
$BWAS_{-M}$	**426**	429,9	2,37	0,90	5,4
$BWAS_{-W}$	**426**	426	0	**0**	1
	Berlin52 (7542)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	**7542**	7542	0	**0**	0
ACS	**7542**	7542	0	**0**	0
$BWAS$	**7542**	7542	0	**0**	0
AS_{+R}	**7542**	7542	0	**0**	0
ACS_{+R}	**7542**	7542	0	**0**	0
$BWAS_{-R-W}$	**7542**	7684,3	87,34	1,85	0
$BWAS_{-M-W}$	**7542**	7560,9	44,18	0,24	2,8
$BWAS_{-R-M}$	**7542**	7668,4	98,80	1,64	0
$BWAS_{-R}$	**7542**	7542	0	**0**	0
$BWAS_{-M}$	**7542**	7542	0	**0**	2,4
$BWAS_{-W}$	**7542**	7542	0	**0**	0
	Brazil58 (25395)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	**25395**	25395	0	**0**	0
ACS	**25395**	25395	0	**0**	0
$BWAS$	**25395**	25395	0	**0**	0
AS_{+R}	**25395**	25395	0	**0**	0
ACS_{+R}	**25395**	25395	0	**0**	0
$BWAS_{-R-W}$	**25395**	25395	0	**0**	0
$BWAS_{-M-W}$	**25395**	25395	0	**0**	0
$BWAS_{-R-M}$	**25395**	25395	0	**0**	0
$BWAS_{-R}$	**25395**	25395	0	**0**	0
$BWAS_{-M}$	**25395**	25395	0	**0**	0
$BWAS_{-W}$	**25395**	25395	0	**0**	0

Table 7: Results obtained in the different instances (II).

	Kroa100 (21282)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	**21282**	21282	0	**0**	0
ACS	**21282**	21282	0	**0**	0
$BWAS$	**21282**	21282	0	**0**	0
AS_{+R}	**21282**	21282	0	**0**	0
ACS_{+R}	**21282**	21282	0	**0**	0
$BWAS_{-R-W}$	**21282**	21617,1	212,50	1,55	0
$BWAS_{-M-W}$	**21282**	21395,8	134,73	0,53	4,5
$BWAS_{-R-M}$	**21282**	21667,1	256,84	1,77	0
$BWAS_{-R}$	**21282**	21282	0	**0**	0
$BWAS_{-M}$	**21282**	21426,2	139,67	0,67	5,1
$BWAS_{-W}$	**21282**	21282	0	**0**	0
	Gr120 (6942)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	6944	6954,1	6,06	0,17	0
ACS	**6942**	6946,1	5,49	0,06	0
$BWAS$	**6942**	6942	0	**0**	0,7
AS_{+R}	6944	6948,9	3,75	0,1	6,9
ACS_{+R}	**6942**	6943,8	3,79	0,03	1,1
$BWAS_{-R-W}$	**6942**	7143,1	164,93	2,81	0
$BWAS_{-M-W}$	**6942**	7030,3	106,50	1,25	5,9
$BWAS_{-R-M}$	6957	7126,2	128,38	2,58	0,4
$BWAS_{-R}$	**6942**	6942	0	**0**	0
$BWAS_{-M}$	**6942**	6997,1	46,17	0,78	4,8
$BWAS_{-W}$	**6942**	6942	0	**0**	0,5
	D198 (15780)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	15796	15811,9	9,50	0,2	0
ACS	**15780**	15784,9	5,67	0,03	0
$BWAS$	**15780**	15780,4	0,51	**0**	3,3
AS_{+R}	15796	15806,2	8,73	0,17	3,5
ACS_{+R}	**15780**	15782,9	4,31	0,02	2,3
$BWAS_{-R-W}$	**15780**	15781,1	1,10	**0**	0
$BWAS_{-M-W}$	**15780**	15781,2	1,03	**0**	5,1
$BWAS_{-R-M}$	**15780**	15781,7	2,31	0,01	0
$BWAS_{-R}$	**15780**	15780,4	0,51	**0**	0
$BWAS_{-M}$	**15780**	15782,2	4,87	0,01	6,2
$BWAS_{-W}$	**15780**	15780,3	0,48	**0**	3

Table 8: Results obtained in the different instances (III).

	Lin318 (42029)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	42205	42348,4	122,43	0,75	0
ACS	**42029**	42230	148,48	0,48	0
$BWAS$	**42029**	42090,2	57,79	**0,14**	5
AS_{+R}	42189	42238,4	45,45	0,50	6,5
ACS_{+R}	**42029**	42182,4	118,12	0,36	5
$BWAS_{-R-W}$	42072	42545,2	408,30	1,21	0
$BWAS_{-M-W}$	42143	42421,1	181,99	0,92	8,6
$BWAS_{-R-M}$	42143	42525,1	176,18	1,16	0
$BWAS_{-R}$	42029	42138,7	81,73	0,26	0
$BWAS_{-M}$	42155	42583,2	277,80	1,30	8,1
$BWAS_{-W}$	**42029**	42129,2	44,84	0,23	7,4
	Pcb442 (50778)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	51213	51284,1	53,04	0,99	0
ACS	50919	51048	75,29	0,53	0
$BWAS$	**50785**	50889,5	79,32	**0,21**	7,9
AS_{+R}	51148	51209,5	41,16	0,84	5,6
ACS_{+R}	50860	51147,5	173,11	0,72	8
$BWAS_{-R-W}$	51069	51604	379,20	1,60	0
$BWAS_{-M-W}$	51065	51293,8	176,12	1,00	9,4
$BWAS_{-R-M}$	51069	51604	379,20	1,60	0
$BWAS_{-R}$	50795	51017	107,80	0,46	0
$BWAS_{-M}$	51024	51545,1	366,20	1,48	10,1
$BWAS_{-W}$	50809	50943,1	88,39	0,32	7,3
	Att532 (27686)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	27796	27843,5	23,83	0,87	0
ACS	27705	27810,3	64,44	0,45	0
$BWAS$	**27686**	27713	16	**0,09**	8,2
AS_{+R}	27755	27786	14,78	0,36	8,7
ACS_{+R}	27745	27835	57,56	0,54	7
$BWAS_{-R-W}$	27830	27953	88	0,95	0
$BWAS_{-M-W}$	27860	28010	87	1,15	8
$BWAS_{-R-M}$	27879	28093	118	1,44	0
$BWAS_{-R}$	27703	27734	27	0,17	0
$BWAS_{-M}$	27854	27982	73	1,05	10,9
$BWAS_{-W}$	27698	27744	26	0,20	8,1

Table 9: Results obtained in the different instances (IV).

	Rat783 (8806)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	8873	8886	13,11	0,90	0
ACS	8857	8892,7	20,93	0,97	0
$BWAS$	**8816**	8837,9	19,23	**0,36**	9,1
AS_{+R}	8850	8863,1	8,33	0,64	7,1
ACS_{+R}	8875	8899,5	22,33	1,05	7,6
$BWAS_{-R-W}$	8922	9185,6	253,42	4,13	0
$BWAS_{-M-W}$	8942	8986,3	38,80	2,10	13,1
$BWAS_{-R-M}$	8958	9063,1	170,44	2,83	0
$BWAS_{-R}$	8817	8838	12,46	**0,36**	0
$BWAS_{-M}$	8922	9042,4	159,62	2,61	10,7
$BWAS_{-W}$	**8816**	8844,4	17,90	0,43	8,9
	U1060 (224094)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	227413	228732	789,05	2,03	0
ACS	225675	226387,8	668,90	1,01	0
$BWAS$	225219	225713	337	**0,71**	7,2
AS_{+R}	228422	229032,2	381,06	2,16	4
ACS_{+R}	225243	226501	1059,57	1,06	2
$BWAS_{-R-W}$	225767	226415	481	1,02	0
$BWAS_{-M-W}$	226045	226426	285	1,02	7,75
$BWAS_{-R-M}$	225826	226223	428	0,94	0
$BWAS_{-R}$	225533	226394	507	1,01	0
$BWAS_{-M}$	**225202**	226321	833	0,98	7,9
$BWAS_{-W}$	225275	226315	619	0,98	7,5
	Fl1577 (22249)				
Model	*Best*	*Average*	*Dev.*	*Error*	*#R*
AS	22732	23213,9	221,09	4,16	0
ACS	22313	22480,8	129,78	1,03	0
$BWAS$	22290	22389,9	81,43	**0,62**	9,1
AS_{+R}	22722	23107,75	191,53	3,72	7,63
ACS_{+R}	**22282**	22454,4	147,85	0,91	4,8
$BWAS_{-R-W}$	22375	22480,2	71,96	1,02	0
$BWAS_{-M-W}$	22354	22546,2	86,33	1,31	8,8
$BWAS_{-R-M}$	22356	22505,1	91,35	1,13	0
$BWAS_{-R}$	22516	22775,1	143,62	2,31	0
$BWAS_{-M}$	22291	22442,2	104,67	0,86	9,1
$BWAS_{-W}$	22452	22613,2	93,73	1,61	9,1

del Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados (AEB '02). Mérida, Spain, 7-9 Febrero, pp. 260-265, 2002.

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

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

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

[11] 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*, Morgan Kaufmann Publishers, pp. 265-283, 1991.

[12] D. S. Johnson, L.A. McGeoch. The Travelling Salesman Problem: A Case Study in Local Optimization. In E.H.L. Arts, J.K. Lenstra (Eds.), *Local Search in Combinatorial Optimization*, John Wiley & Sons, pp. 215-310, 1997.

[13] 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.

[14] G. Reinelt, TSPLIB - A Travelling Salesman Problem Library. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft.

[15] T. Stützle, H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems Journal*, 16(8), pp. 889-914, 2000.