Analysis of the Best-Worst Ant System and Its Variants on the QAP

Oscar Cordón, Iñaki Fernández de Viana, and Francisco Herrera

Dept. of Computer Science and Artificial Intelligence. E.T.S.I. Informática
University of Granada, Avda. Andalucía, 38. 18071 Granada, Spain
{ocordon,herrera}@decsai.ugr.es, ijfviana@teleline.es

Abstract. In this contribution, we will study the influence of the three components of the Best-Worst Ant System (BWAS) algorithm. As the importance of each of them as the fact whether all of them are necessary will be analyzed. Besides, we will introduce a new algorithm called Best-Worst Ant Colony System by combining the basis of the Ant Colony System with the special components of the BWAS. The performance of different variants of these algorithms will be tested when solving different instances of the QAP.

1 Introduction

In [3], a new algorithm called BWAS was intended being based on the AS algorithm [4]. In this contribution we develop an study applying the BWAS algorithm and its variants to the QAP. Besides, a new algorithm called Best-Worst Ant Colony System (BWACS) will be considered as well. Our aim is to demonstrate that these algorithms are robust as a whole and to analyze the relative importance among their distinguishing components.

This paper is structured as follows. In Section 2 and 3 the basis of the BWAS, its variants as well as the BWACS algorithm are studied. In section 4, the application of the ACO algorithms to the QAP is reviewed and the results we obtained are presented. We end up by discussing some concluding remarks and future works.

2 The Best-Worst Ant System

The BWAS model tries to improve the performance of ACO models using evolutionary algorithm concepts. The proposed BWAS uses the *AS transition rule*:

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

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 real-valued weights.

Besides, the usual *AS evaporation rule* is used: $\tau_{rs} \leftarrow (1-\rho)\cdot\tau_{rs}, \ \forall\, r, s$, with $\rho \in [0,1]$ being the pheromone decay parameter. Additionally, the BWAS considers the three following daemon actions, that are analyzed in deep in [3]:

Best-Worst performance update rule. This rule is based on the Population-Based Incremental Learning (PBIL) [1] probability array update rule. The *offline pheromone trail updating* is done as follows:

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

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 the solution it generated, $C(S_{global-best})$.

Moreover, the edges present in the worst current ant are penalized: $\forall (r,s) \in S_{current-worst} \ and \ (r,s) \notin S_{global-best}, \ \tau_{rs} \leftarrow (1-\rho)\cdot\tau_{rs}$.

Pheromone trail mutation. The pheromone trails suffer mutations to introduce diversity in the search, as done in PBIL with the memoristic structure. To do so, each row of the pheromone matrix is mutated —with probability P_m— as follows:

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

with a being a random value in $\{0,1\}$, it being the current iteration, $\tau_{threshold}$ being the average of the pheromone trail in the edges composing the global best solution and with $mut(\cdot)$ being a function making a stronger mutation as the iteration counter increases.

Restart of the search process when it gets stuck. We will perform the restart by setting all the pheromone matrix components to τ_0 when the number of edges that are different between the current best and the current worst solutions is lesser than a specific percentage.

A simplified structure of a generic BWAS algorithm 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. and include r in $L_k$
     – While (ant k not in a target node) do
         • Select the next node to visit, $s \notin L_k$, by the AS transition rule.
3. For k=1 to m do
     – Run the local search improvement on the solution generated by ant k, $S_k$.
4. $S_{global-best} \leftarrow$ global best ant tour. $S_{current-worst} \leftarrow$ current worst ant tour.
5. Pheromone evaporation and Best-Worst pheromone updating.
6. Pheromone matrix mutation and restart if condition is satisfied.
7. If (Stop Condition is not satisfied) go to step 2.
```

3 Analysis of Best-Worst Ant System Components

In this section, a new ACO model based on the BWAS components is introduced, as well as the different variants considered from the basic BWAS and the new model.

3.1 The Best-Worst Ant Colony System and Its Variants

This new model is obtained by introducing the three components of the BWAS into the ACS. Hence, the differences between both algorithms are that BWACS considers the usual *ACS transition rule*

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

and that the *online step-by-step updating rule* is applied during the ants trip: $\tau_{rs} \leftarrow (1-\varphi)\cdot\tau_{rs}+\varphi\cdot\Delta\tau_{rs}$, with $\varphi \in [0,1]$ being the pheromone decay parameter.

Hence, the only differences between BWAS and BWACS lie on the step 2 of the algorithm.

3.2 BWAS Variants Analyzed

The main objective of this paper is to study the influence of the three components of BWAS on its application to the QAP. 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. Then, we will also try to establish a ranking of importance among components.

This analysis will be made from a double perspective: i) individualized analysis of components, and ii) cooperative analysis among pairs of components.

It seems that a certain interrelation exists among the three basic elements of BWAS. The updating 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 the stagnation of the algorithm. It can 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 algorithms used in the study are summarized. As can be seen, there are three different groups of algorithms: i) the first one includes the basic models: our two proposals, BWAS and BWACS, and the classical AS and ACS, considered for comparison purposes; ii) the second is composed of 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 the AS and ACS, respectively; iii) the third is comprised by the variants including a pair of the components. The different variants are notated by $BWAS_{-*}$ or $BWACS_{-*}$ standing * for the removed component (R, M or W).

Table 1. ACO models considered.

Parameter	Meaning
AS	Ant System
ACS	Ant Colony System
$BWAS$	Best-Worst Ant System
$BWACS$	Best-Worst Ant Colony System
AS_{+R}	AS with BWAS restart
ACS_{+R}	ACS with BWAS Restart
$BWAS_{-R-W}$	BWAS without restart and worst ant trail updating
$BWAS_{-M-W}$	BWAS without mutation and worst ant trail updating
$BWAS_{-R-M}$	BWAS without restart and mutation
$BWACS_{-R-W}$	BWACS without restart and worst ant trail updating
$BWACS_{-M-W}$	BWACS without mutation and worst ant trail updating
$BWACS_{-R-M}$	BWACS without restart and mutation
$BWAS_{-R}$	BWAS without restart.
$BWAS_{-W}$	BWAS without worst ant trail updating.
$BWAS_{-M}$	BWAS without mutation.
$BWACS_{-R}$	BWACS without restart.
$BWACS_{-W}$	BWACS without worst ant trail updating.
$BWACS_{-M}$	BWACS without mutation.

4 Experiments Developed and Analysis of Results

In this section, the application of ACO to the QAP is reviewed and the experiments developed and the analysis of the results obtained are reported.

4.1 Application of the ACO Algorithms Considered to the QAP

The QAP [2] is among the hardest combinatorial optimization problems. All the QAP instances used in our experimentation have been obtained from the QAPLIB [5]. We have chosen two instances of different sizes from each of the four existing classes [6] in order to perform a fair comparative study. These instances are respectively: tai50a, tai60a, nug20, sko72, bur26a, kra30a, tai50b and tai80b.

When applying the different ACO algorithms selected to solve the QAP, the next steps have to be considered: i) as in [6], we not considering the heuristic information in the transition rule. ii) the 2-opt local search algorithm considered in [6] is used.

4.2 Parameter Settings

The ACO models shown in Table 1 have been used to solve the eight QAP instances. Each model has been run 10 times in a 1400 MHz AMD Athlon processor computer. The parameter values considered are shown in Table 2, with the ones associated to AS and ACS taken from [6] and the BWAS and BWACS ones from [3]. The latter parameter values have not been obtained from

Table 2. Parameter values considered

Parameter	**Value**
Number of ants	$m = 5$
Maximum run time	$Ntime = 600\ seconds$
Pheromone updating rules parameter	$\rho = 0.2$
AS offline pheromone rule updating	$f(C(S_k)) = \frac{1}{C(S_k)}$
ACS offline pheromone rule updating	$f(C(S_{global-best})) = \frac{1}{C(S_{global-best})}$
Transition rule parameters	$\alpha = 1$, $\beta = 0$
ACS transition rule parameters	$q_0 = 0.98$
Initial pheromone amount	$\tau_0 = 10^{-6}$
BWAS **parameters**	
Pheromone matrix mutation probability	$P_m = \{0.1, 0.15, 0.3\}$
Mutation operator parameter	$\sigma = 4$
5%	
Local search procedure parameters	
Neighbor choice rule	best improvement
Number of iterations	1000

a deep study, and only some preliminary experiments with different mutation probability values (0.1, 0.15 and 0.3) have been done.

4.3 Analysis of Results

Tables 3 and 4 collect a summary of the obtained results. Table 3 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 an algorithm i is better than another algorithm j for a problem instance p if the average error[1] obtained by i for p is smaller than that obtained by j. Notice that the values in Table 3 are symetric ($a_{ij} = 100 - a_{ji}$) in all cases but not in those where there have been draws between both algorithms.

A general classification of the models is shown in Table 4 which summarizes the values shown in Table 3. While the first column contains the name of the model, the other three columns collect the number of algorithms regarding which the model has obtained better, worse or similar results, respectively.

If we compare the results of the basic algorithms (AS, ACS, BWAS and BWACS), *BWAS* and *BWACS* present the best behavior. In every case, the best error was obtained by a BWAS model or by any of its variants.

Analyzing the results of the BWACS and its variants, we see how a great trade-off exists among its components. Notice that the elimination of any of them makes the algorithm worsen. For the eight instances, the BWACS has outperformed all its variants. However, we do not find the same in the BWAS. There are instances in which some variants of the BWAS overcome the basic

[1] 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. Pair comparisons between ACO model

Model	AS	ACS	$BWAS$	$BWACS$	AS_{+R}	ACS_{+R}	$BWAS_{-R-W}$	$BWAS_{-M-W}$	$BWAS_{-R-M}$	$BWACS_{-R-W}$	$BWACS_{-M-W}$	$BWACS_{-R-M}$	$BWAS_{-R}$	$BWAS_{-W}$	$BWAS_{-M}$	$BWACS_{-R}$	$BWACS_{-W}$	$BWACS_{-M}$
AS	-	62	0	0	50	50	0	0	37	25	37	62	0	0	0	50	25	37
ACS	37	-	37	25	50	37	37	12	62	50	37	87	37	25	37	50	37	50
$BWAS$	87	62	-	37	87	50	75	37	87	75	50	100	75	37	50	87	62	62
$BWACS$	87	75	37	-	87	62	75	50	100	75	50	100	75	37	62	87	50	50
AS_{+R}	37	50	0	0	-	50	12	12	37	37	25	62	0	12	12	62	25	37
ACS_{+R}	50	62	37	25	50	-	37	12	75	50	37	87	37	25	37	62	25	37
$BWAS_{-R-W}$	87	62	0	0	75	50	-	25	62	62	37	100	50	12	12	62	50	50
$BWAS_{-M-W}$	87	87	37	25	75	75	50	-	62	87	62	100	37	25	37	87	50	75
$BWAS_{-R-M}$	62	37	12	0	62	25	37	37	-	37	25	100	37	0	25	50	25	37
$BWACS_{-R-W}$	50	50	12	12	50	50	25	0	62	-	25	100	25	0	25	75	25	37
$BWACS_{-M-W}$	50	62	25	25	50	50	37	12	75	62	-	100	37	12	37	87	25	50
$BWACS_{-R-M}$	37	12	0	0	37	12	0	0	0	0	0	-	0	0	0	25	12	0
$BWAS_{-R}$	87	62	0	0	87	50	25	37	50	62	37	100	-	25	12	62	50	50
$BWAS_{-W}$	87	75	25	37	75	62	62	50	100	87	62	100	50	-	62	87	75	75
$BWAS_{-M}$	87	62	25	12	75	50	$\underline{50}$	37	75	62	37	100	62	12	-	75	50	50
$BWACS_{-R}$	25	50	0	0	25	37	25	0	50	0	0	75	25	0	12	-	0	0
$BWACS_{-W}$	62	62	12	25	62	50	25	12	75	62	50	87	25	0	25	87	-	37
$BWACS_{-M}$	50	50	12	25	50	37	25	0	62	50	25	100	25	0	25	87	25	-

Table 4. ACO models standing

Model	*Best performance*	*Worst performance*	*Similar performance*
$BWAS$	15	0	2
$BWACS$	15	0	2
$BWAS_{-W}$	15	1	1
$BWAS_{-M-W}$	12	2	3
$BWAS_{-M}$	12	3	2
$BWAS_{-R-W}$	11	5	1
$BWAS_{-R}$	10	5	2
$BWACS_{-M-W}$	9	5	3
$BWACS_{-W}$	10	7	0
$BWACS_{-M}$	6	9	2
$BWACS_{-R-W}$	5	10	2
ACS_{+R}	4	9	4
AS	4	12	1
ACS	2	11	4
$BWAS_{-R-M}$	3	13	1
AS_{+R}	2	13	2
$BWACS_{-R}$	1	13	3
$BWACS_{-R-M}$	0	17	0

BWAS algorithm. In spite of this, the BWAS presents better overall performance than its variants.

On the other hand, when analyzing the individual importance of each component, we see that the restart is the one giving the best results, followed by the mutation and the worst ant update. Among the combinations of two components, those not using the worst ant update achieve the best performance. Besides, those that do not consider the restart component present a very low performance. When eliminating the mutation, the results are neither as bad as if we remove the restart, nor as good as if we remove the worst ant update.

In view of these results, we can conclude that: i) in general, an appropriate trade-off exists among the three components of both BWAS and BWACS; ii) in spite of this balance, we can establish an order of importance among the components: restart, mutation, and worst update.

5 Concluding Remarks and Future Works

In this contribution, a study of all the components of *BWAS* has been done. Besides, a new variant of BWAS called BWACS has been proposed. The performance of these algorithms and the importance of their components has been analyzed when solving eight QAP instances of different sizes and types. It has shown that the best performance have been obtained using the basic versions of BWAS and BWACS algorithms, without removing any component.

Different ideas for future developments arise: i) to study the influence of the appropriate values for the parameters, and ii) to analyze the consideration of other Evolutionary Computation aspects.

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*, Morgan Kaufmann Publishers, pp. 38-46, 1995.
2. R.E Burkard, E. Cela. Quadratic and three-dimensional assignment problem. In M. Dell'Amico, F. Maffioli, and S. Martello (Eds.), *Annotated Bibliographies in Combinatorial Optimization*, 1996.
3. 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.
4. M. Dorigo, G. Di Caro. Ant Algorithms for Discrete Optimization. *Artificial Life* 5(2), pp. 137-172, 1999.
5. E. Rainer, S. Burkard, E. Karish, F. Rendl. QAPLIB-A Quadratic Assignment Problem Library. http://www.opt.math.tu-graz.ac.at/∼karish/qaplib.
6. T. Stützle, M. Dorigo. ACO Algorithms for the Quadratic Assignment Problem. In D. Corne, M. Dorigo and F. Glover, editors, New Ideas in Optimization, McGraw-Hill.