Studying the Influence of the Objective Balancing Parameter in the Performance of a Multi-Objective Ant Colony Optimization Algorithm

A.M. Mora, J.J. Merelo, P.A. Castillo, J.L.J. Laredo,
P. García-Sánchez, and M.G. Arenas

Abstract. Several multi-objective ant colony optimization (MOACO) algorithms use a parameter λ to balance the importance of each one of the objectives in the search. In this paper we have studied two different schemes of application for that parameter: keeping it constant, or changing its value during the algorithm running, in order to decide the configuration which yields the best set of solutions. We have done it considering our MOACO algorithm, named hCHAC, and two other algorithms from the literature, which have been adapted to solve the same problem. The experiments show that the use of a variable value for λ yields a wider Pareto set, but keeping a constant value for this parameter let to find better results for any objective.

1 Introduction

The *military unit path-finding problem* consists in getting the best path for a military unit, from an origin to a destination point in a battlefield, keeping a balance between route speed and safety, considering the presence of enemies (which can fire against the unit) and taking into account some properties and restrictions which make the problem more realistic. Being speed (important if the unit mission requires arriving as soon as possible to the target) and safety (important when the enemy forces are not known or when the unit effectives are very valuable), the two main criteria that the commander of a unit takes into account inside a battlefield in order to accomplish the mission with success.

To solve this problem we designed an Ant Colony Optimization algorithm [4] adapted to deal with two objectives (see [3] for a survey on multi-objective optimization), named hCHAC [10] so it is a Multi-Objective Ant Colony Optimization Algorithm (MOACO [5]).

A.M. Mora · J.J. Merelo · P.A. Castillo · J.L.J. Laredo · P. García-Sánchez · M.G. Arenas
Dpto. Arquitectura y Tecnología de Computadores. University of Granada, Spain
e-mail:{amorag,jmerelo,pedro,juanlu,
pgarcia,mgarenas}@geneura.ugr.es

J.R. González et al. (Eds.): NICSO 2010, SCI 284, pp. 163–176, 2010.
springerlink.com

As most of the metaheuristics, it considers a set of parameters which determines the behaviour in the search, usually having an effect in the exploration and exploitation balance. But in hCHAC case, there is also an additional (and key) parameter, named λ. It was introduced in the algorithm rule of decision, to set the relative importance of each one of the objectives in the search. But actually, it sets the area in the space of solutions, that each of the ants explores, yielding somewhat of 'specialised' ants in each one of the objectives (or both of them).

Some of the parameters of hCHAC and their influence in the search, were analysed in a previous work [8] using statistical methods, reaching some conclusions in addition to the best set of values for them.

In the present work the λ parameter has been studied, but statistics have not been applied in the analysis, because λ just weights the relative importance of one objective with respect to the other (there are just two objectives in this problem) and does not take concrete (and sometimes hard-coded) values as the rest of parameters. Moreover, two different parameter schemes can be applied in the algorithm, and the aim of the analysis is to study their influence in the search and decide which one is the best scheme.

The rest of the paper is structured as follows. Firstly, the problem to solve and its modelling in a simulator environment is briefly described in Section 2. Then the hCHAC and the literature algorithms (adapted to solve the problem) are introduced respectively in Sections 3 and 4. The parameter to study (λ)) is commented in Section 5. Section 6 shows the performed analysis, by presenting some problem instances and the results of the experiments. Finally, in Section 7 the conclusions and the future work in this line are exposed.

2 The Military Unit Bi-criteria Pathfinding Problem

The MUPFP-2C is modelled considering that the unit has got a *level of energy (health)* and a *level of resources*, which are consumed when it moves along the path, so the problem objectives are adapted to minimize the resources and energy consumption. The problem is located inside a battlefield which is modelled as a grid of hexagonal cells with a *cost in resources*, which represents the difficulty of going through it, and a *cost in energy/health*, which means the unit depletes its human resources or vehicles suffer damage when crossing over the cell (*no combat casualties*). Both costs depend on the cell type. Besides, moving between cells with different heights also costs resources (more if it goes up), and falling in a weapons impact zone depletes energy. All these features are represented using different colors in the maps.

Figure 1 shows an example of real world battlefield and the information layer associated to it, which has been created using a custom-made application [1].

We consider *fast* paths (if speed is constant) when the total cost in resources is low (it is not very difficult to travel through the cells, so it takes little time). On the other hand *safe* paths, have associated a low cost in energy/health.

See [10] for further details about the problem definition and modelling.

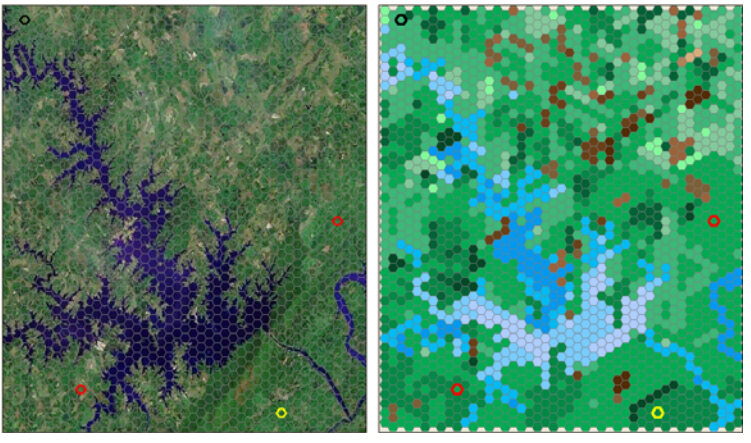

Fig. 1 Example Map (45x45 cells). The image on the left-hand side is a real world picture showing a lake surrounded by some hills and lots of vegetation. On the right-hand side it is shown its associated information layer. The different shades in the same color models height (light color) and depth (dark color). There are two enemies with red border, an origin point (in the top-left corner of the images) with black border and a destination point (in the bottom-right) with yellow border

3 hCHAC Algorithms

There were designed some algorithms to solve the commented problem, all of them were included in the so-called *hCHAC*[1] family. In this work two of them will be considered.

The main approach, also known as ***hCHAC*** [9] is an Ant Colony System (ACS) [4] adapted to deal with two objectives (MOACO) [3, 5]. Since it is an ant algorithm, the problem is transformed into a graph where each node corresponds to a cell in the map, and each edge between two nodes is the connection between neighbor cells in the map. Every edge has associated two weights, which are the costs in resources and health that going through that edge causes to the unit.

In every iteration, the ants separately build a complete path (solution), between the origin and destination points (if possible), by travelling through the graph. To guide this movement they use a State Transition Rule (STR) which combines two kinds of information: pheromone trails (learnt information) and heuristic knowledge.

The MUPFP-2C has two independent objectives to minimize. These objectives are named $\boldsymbol{f}$, minimization of the resources consumed in the path (fast path) and $\boldsymbol{s}$, minimization of the energy/health consumed in the path (safe path).

hCHAC uses two pheromone matrices (τ_f, τ_s) and two heuristic functions (η_f, η_s) (one per objective), a single colony, and two STRs: (*Combined State Transition*

[1] Which means *Compañía de Hormigas ACorazadas* (Armored Ant Company) with the prefix 'hexa' due to the grid topology where it works.

Rule, CSTR), similar to the one proposed in [6] and (*Dominance State Transition Rule, DSTR*), which ranks neighbouring cells according to how many (of the neighbours) they dominate.

The local and global pheromone updating formulae are based in the MACS-VRPTW algorithm proposed in [2], with some changes due to the use of two pheromone matrices. Finally, there are two evaluation functions (used to assign a global cost value to every solution found) named F_f (minimization of resources consumption) and F_s (minimization of energy consumption).

The definition and formulae of all these features can be found in [9].

The CSTR uses the λ parameter, so it is the rule which we are going to consider in the present study. It is defined as:

If ($q \leq q_0$)

$$j = \arg\max_{j \in N_i} \left\{ \tau_f(i,j)^{\alpha \cdot \lambda} \cdot \tau_s(i,j)^{\alpha \cdot (1-\lambda)} \cdot \eta_f(i,j)^{\beta \cdot \lambda} \cdot \eta_s(i,j)^{\beta \cdot (1-\lambda)} \right\} \quad (1)$$

Else

$$P(i,j) = \begin{cases} \dfrac{\tau_f(i,j)^{\alpha \cdot \lambda} \cdot \tau_s(i,j)^{\alpha \cdot (1-\lambda)} \cdot \eta_f(i,j)^{\beta \cdot \lambda} \cdot \eta_s(i,j)^{\beta \cdot (1-\lambda)}}{\sum_{u \in N_i} \tau_f(i,u)^{\alpha \cdot \lambda} \cdot \tau_s(i,u)^{\alpha \cdot (1-\lambda)} \cdot \eta_f(i,u)^{\beta \cdot \lambda} \cdot \eta_s(i,u)^{\beta \cdot (1-\lambda)}} & if\ j \in N_i \\ 0 \quad otherwise \end{cases} \quad (2)$$

In that rule, $q_0 \in [0,1]$ is the standard ACS parameter and q is an uniformly random selected value in [0,1]. τ_f, τ_s and η_f, η_s are the previously commented matrices and functions. α and β are the usual (in ACO algorithms) weighting parameters for pheromone and heuristic information respectively, and N_i is the current feasible neighbourhood for the node i. As can be seen, the λ parameter is used to weight the terms (pheromone and heuristic values) related to each one of the objectives, using its value 'λ' for the first term and the complementary '(1 - λ)' for the second.

This state transition rule works as follows: when an ant is building a solution path and is placed at one node i, a random number q in [0,1] is generated, if $q \leq q_0$ the best neighbor j is selected as the next node in the path (Equation 1). Otherwise, the algorithm decides which node is the next by using a roulette wheel considering *P(i,j)* as probability for every feasible neighbour j (Equation 2).

The DSTR has not been taken into account in this work, because it does not use the λ parameter, since the objectives are considered as completely independent in the rule, and they are not weighted and combined (it is not an aggregative function).

The other proposed algorithm which has been studied in this work is ***hCHAC-4*** [7], a redefinition of the bi-criteria hCHAC focused to deal with four objectives, since each one of the main considered criteria can be subdivided into two sub-objectives, this way: *speed* can be defined as distance to target point minimization

(straight paths) and cost in resources minimization; while *safety* can be understood as visibility[2] to enemies and cost in energy/health minimizations.

So, the four objectives to minimize, have been considered separately: *resources consumption* (**r**), *distance to target point* (**d**), *energy consumption* (**e**) and *visibility to enemies* (**v**); the two first are related to speed, and the others to safety.

hCHAC-4 is also a MOACO algorithm that works in a graph (which models the battlefield), but considering four weights in each edge. It is again an ACS, so it uses the q_0 parameter in the STR. All the elements of the algorithm have been adapted to deal with four objectives, so there are four heuristic functions, four pheromone matrices, and four evaluation functions. In addition, and as in hCHAC, there are two different state transition rules, which work considering four objectives this time.

The CSTR-4 is similar to the CSTR of hCHAC, but involving four terms (one per objective). Each one of these terms is defined as follows:

$$T_x(i,j) = \tau_x(i,j)^{\alpha} \cdot \eta_x(i,j)^{\beta} \tag{3}$$

τ is the correspondent pheromone trails matrix, η is the heuristic function, and $x = r,d,e,v$. α and β are the usual (in ACO algorithms) weighting parameters for pheromone and heuristic information respectively. So, the STR for four objectives (CSTR-4) is:

If $(q \leq q_0)$

$$j = \arg\max_{j \in N_i} \left\{ T_r(i,j)^{\lambda} \cdot T_d(i,j)^{\lambda} \cdot T_e(i,j)^{(1-\lambda)} \cdot T_v(i,j)^{(1-\lambda)} \right\} \tag{4}$$

Else

$$P(i,j) = \begin{cases} \dfrac{T_r(i,j)^{\lambda} \cdot T_d(i,j)^{\lambda} \cdot T_e(i,j)^{(1-\lambda)} \cdot T_v(i,j)^{(1-\lambda)}}{\sum_{u \in N_i} T_r(i,u)^{\lambda} \cdot T_d(i,u)^{\lambda} \cdot T_e(i,u)^{(1-\lambda)} \cdot T_v(i,u)^{(1-\lambda)}} & if\ j \in N_i \\ 0 & otherwise \end{cases} \tag{5}$$

Where all the parameters and terms are the same as in the CSTR of hCHAC. As can be seen, the λ parameter is also used in this equation. It sets the importance of all objectives at the same time, since they are related with speed (resources consumption and distance to target point) or with safety (energy consumption or visibility to enemies). The rule works as the previously commented STR.

Again, the DSTR (for four objectives this time) will not be considered in this study since it does not use the λ parameter.

[2] It is considered a *cost in visibility* with regard to the enemies, which is minimum if the unit is hidden (at a point) to all the enemies, and it increases exponentially when it is visible to any (or some) of them. With no enemy present, it is computed taking into account whether it is visible to the surrounding cells in a radius, calculating a score, so the higher number of cells can see the unit, the higher the score is.

4 Algorithms to Compare

In previous works we considered two MOACO algorithms to make results comparisons which those yielded by hCHAC family methods. They had been presented by other authors in literature and were adapted to solve the MUPFP-2C. Both of them use the λ parameter in the STR to weight the objectives and to address the search performed by the ants. We will consider them in this study in order to get more general conclusions.

The first one is ***MOACS*** (Multi-Objective Ant Colony System), which was proposed by Baran et al. [2], to solve the Vehicle Routing Problem with Time Windows (VRPTW). It uses a single pheromone matrix for both objectives (instead of one per objective).

It has been adapted to solve the MUPFP-2C [9], so it considers the same heuristic and evaluation functions (see them in [10]), but different STR and pheromone updating formulae. The STR is similar to the CSTR in hCHAC, but using only one pheromone matrix (as we previously said). It is defined as follows:

If $(q \leq q_0)$

$$j = \arg\max_{j \in N_i} \left\{ \tau(i,j) \cdot \eta_f(i,j)^{\beta \cdot \lambda} \cdot \eta_s(i,j)^{\beta \cdot (1-\lambda)} \right\} \tag{6}$$

Else

$$P(i,j) = \begin{cases} \dfrac{\tau(i,j) \cdot \eta_f(i,j)^{\beta \cdot \lambda} \cdot \eta_s(i,j)^{\beta \cdot (1-\lambda)}}{\sum_{u \in N_i} \tau(i,u) \cdot \eta_f(i,u)^{\beta \cdot \lambda} \cdot \eta_s(i,u)^{\beta \cdot (1-\lambda)}} & if\ j \in N_i \\ 0 & otherwise \end{cases} \tag{7}$$

Where all the terms and parameters are the same as in Equation 2. This rule also uses λ to balance the importance of the objectives in the search. The rule works as we previously explain for hCHAC and hCHAC-4.

Since *MOACS* is an ACS, there are two levels of pheromone updating, local and global. There has been defined new equations (in respect to the author's original algorithm definition) for both tasks, in addition to a new reinitialization mechanism, which can be also consulted in [9].

The evaluation functions for each objective F_f and F_s, are the same as in the previous approaches.

The second algorithm is ***BiAnt*** (BiCriterion Ant), which was proposed by Iredi et al. [6] as a solution for a multi-objective problem with two criteria (the Single Machine Total Tardiness Problem, SMTTP). It is an Ant System (AS) which uses just one colony, and two pheromone matrices and heuristic functions (one per objective). So, the STR is similar to the CSTR of hCHAC, but without consider the q_0 parameter, it is:

$$P(i,j) = \begin{cases} \dfrac{\tau_f(i,j)^{\alpha\cdot\lambda}\cdot\tau_s(i,j)^{\alpha\cdot(1-\lambda)}\cdot\eta_f(i,j)^{\beta\cdot\lambda}\cdot\eta_s(i,j)^{\beta\cdot(1-\lambda)}}{\sum_{u\in N_i}\tau_f(i,u)^{\alpha\cdot\lambda}\cdot\tau_s(i,u)^{\alpha\cdot(1-\lambda)}\cdot\eta_f(i,u)^{\beta\cdot\lambda}\cdot\eta_s(i,u)^{\beta\cdot(1-\lambda)}} & if\ j\in N_i \\ 0 \quad otherwise \end{cases} \tag{8}$$

Where all the terms and parameters are again the same as in Equation 2. In addition, the rule uses the λ parameter to weight the objectives in the search. The rule works in the same way that hCHAC CSTR but without consider the random number q, just directly calculating the probability for the feasible nodes using this formula and using a roulette wheel to choose the next node in the path.

The definition of the heuristic and evaluation functions are the same as in hCHAC. But, the pheromone updating scheme is different since BiAnt is an AS. So it is just performed a global pheromone updating, including evaporation in all nodes and contribution just in the edges of the best paths to the moment (those included in the Pareto Set (PS)).

5 The λ Parameter

As shown in the equations applied in the algorithms, there are some parameters considered in their expressions. Most of them were previously analysed [8] in order to determine their influence on the behaviour of the main algorithm (hCHAC) and the best set of values that they should take to yield the best solutions.

However, there is a very important parameter which has not been studied yet. It is present in all the STRs (of the commented algorithms) and, since these rules are the most important factor in every ACO algorithm, this parameter is key in the algorithm performance. It is λ, the parameter which determines the importance of each one of the objectives in the STR.

$\lambda \in [0,1]$, has been to the moment user-defined[3], determining which objective has higher priority and how much. If the user decides to search for a fast path, λ will take a value close to 1, on the other hand, if he wants a safe path, close to 0.

This value has been considered as constant during the algorithm for all ants, so the algorithms always search in the same zone of the space of solutions (the zone related to the chosen value for λ). That is, all the ants search in the same area of the Pareto Front (PF) [3], yielding solutions (in average) with similar costs in both objectives. These cost would maintain the relationship determined by the weight set through the λ value.

This was the initial idea applied in hCHAC and in hCHAC-4 [7, 10], and this scheme has been also implemented in the adapted (to the MUPFP-2C) versions of MOACS and BiAnt, commented in previous section.

These algorithms were initially defined [2, 6] with a different policy for λ, which consists in assign a different value for the parameter to each ant h, following the expression:

[3] This algorithms have been applied inside a simulator where an user can be determine in advance the importance of each objective in the search.

$$\lambda_h = \frac{h-1}{m-1} \qquad \forall h \in [1,m] \tag{9}$$

Considering that there are m ants, the parameter takes an increasing value that goes from 0 for the first ant to 1 for the last one. This way, the algorithms search in all the possible areas of the space of solutions (each ant is devoted to a zone of the PF).

This is the recommended scheme for solving classical MO problems, in which the biggest (and fittest) Pareto Set (PS) is wanted to be obtained, but the MUPFP-2C is addressed to get a set of solutions according to the user decision, that is, a set of solutions with the desired relationship of importance between the objectives.

So, the idea could be to use this search scheme and to restrict the yielded solutions using λ once the final PS has been obtained.

This way, the aim of the study is to decide the best scheme for applicating λ: the *constant scheme*, where the value for the parameter is set at the beginning of the algorithm for all the ants; and the *variable scheme*, where every ant considers its own value for the parameter during the search and the user-criteria is applied at the end of the run for restrict the solutions in the PS.

6 Experiments and Results

In order to study the different configurations for λ, some problems have been solved using each one of the commented algorithms, considering in addition each one of the schemes: constant and variable λ application.

So, the experiments have been performed in three different (and realistic) maps, modelled from some screens of the PC Game Panzer General™. These maps are *PG-Forest Map* (Figure 2), *PG-River Map* (Figure 3) and *PG-Mountain Map* (Figure 4).

All the algorithms have been run in these maps using the same parameter values: α=1, β=2, ρ=0.1 and q_0=0.4 (tending to an exploitative search more as usual in ACO algorithms). The λ parameter has taken values 0.9 and 0.1 in the constant scheme to consider one objective with higher priority than the other.

All these MOACOs yield a set of non-dominated solutions, but less than usual in this kind of algorithms, since it only searches in the region of the ideal Pareto front determined by the λ parameter. In addition, we usually only consider one (which is chosen by the military staff following their own criteria and the features of each problem).

The considered evaluation functions are: F_f (minimization of the resources consumed in the path, or fast path), and F_s (minimization of the energy consumed in the path, or safe path). As a reminder, even in the hCHAC-4 algorithm, the final solutions are evaluated using these functions to compare with the yielded results by the other algorithms. 30 independent runs (1500 iterations and 30 ants) have been performed with each one of the algorithms, using both schemes for λ, and searching for the fastest and safest paths (in two different runs) in the case of the constant scheme, and searching for both types of paths in the variable scheme.

Fig. 2 PG-Forest Map. 45x45 cells map where some patches of forest are shown, there are also some small villages and hills. The unit is located at the north (black border cell), the target at the south (with yellow border), and there is one enemy placed in the centre (red border cell). On the right figure it is shown the underlying information layer which models the map on the left figure

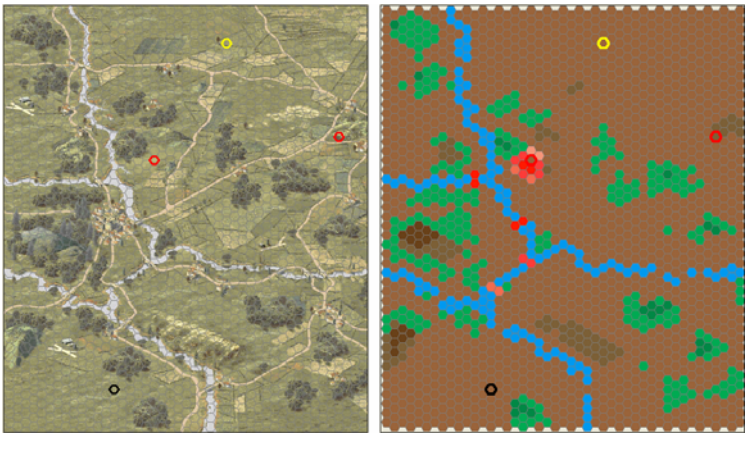

Fig. 3 PG-River Map. 45x45 cells map where it is modelled an scenery with some villages and cities, there are also some rivers and bridges, a patch of forest and some hills. The unit is placed at the south (black border cell) and the target point at the north (with yellow border). There are two enemies (cells with red border), one of them firing at the zone surrounding him and also at some bridges (cells in red color), and the other one watching over on the top of a hill. On the right figure it is shown the underlying information layer which models the map on the left

At the end of every run, and depending on the scheme, some of the solutions in the PS have been chosen, so in the constant approach the best solution in the correspondent PS (fast paths PS or safe paths PS), looking at the cost related to

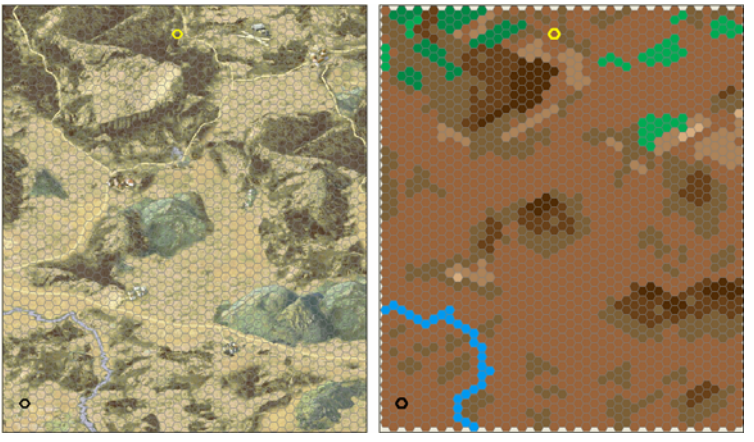

Fig. 4 PG-Mountain Map. 45x45 cells map modelling a mountainous zone, with many mountains, hills, hollows and valleys. The problem unit is placed at the south-west (with black border) and the target point at the north (with yellow border). There is no known enemy. The right figure shows the underlying information layer which models the map on the left

the preferred objective is selected: the one with the smallest F_f cost in the case of fast paths (speed objective with higher priority), and the one with the smallest F_s cost in the case of safe paths (safety objective with higher priority). In the variable approach, the best solutions depending on each one of these costs are selected, but just in a single PS (containing fast and safe paths). Once the best solutions in all the runs have been chosen, the mean and standard deviation of all of them have been calculated and presented in the Tables 1, 2 and 3.

Table 1 λ study results for the four algorithms in PG-Forest Map. 1500 iterations, 50 ants

			Fastest (λ=0.9)		Safest (λ=0.1)	
			F_f	F_s	F_f	F_s
Constant λ	hCHAC	Best	**68.50**	295.40	80.50	**7.30**
		Mean	**77.88 ±7.84**	166.20 ±131.08	84.67 ±3.64	**8.02 ±0.55**
	hCHAC-4	Best	**70.00**	305.50	89.00	**8.30**
		Mean	**79.52 ±6.67**	322.28 ±37.59	110.33 ±14.18	**73.94 ±86.66**
	MOACS	Best	**74.00**	286.00	89.50	**8.20**
		Mean	**83.03 ±5.08**	227.03 ±111.94	101.95 ±6.71	9.29 ±0.53
	BiAnt	Best	**84.50**	297.00	146.50	**13.90**
		Mean	**123.82 ±32.86**	320.49 ±75.44	158.75 ±32.25	**284.47 ±152.01**
Variable λ	hCHAC	Best	**68.50**	295.40	80.50	**7.30**
		Mean	80.60 ±6.36	85.84 ±118.90	84.98 ±3.34	8.11 ±0.51
	hCHAC-4	Best	74.50	285.90	96.00	9.30
		Mean	93.62 ±10.27	290.07 ±101.75	110.98 ±15.78	195.59 ±89.03
	MOACS	Best	77.50	286.20	92.50	**8.20**
		Mean	86.53 ±5.51	214.02 ±110.22	97.28 ±5.66	**9.16 ±0.68**
	BiAnt	Best	101.00	238.80	129.00	12.30
		Mean	138.37 ±32.13	314.30 ±111.90	145.47 ±29.88	288.32 ±111.31

Table 2 λ study results for the four algorithms in PG-River Map. 1500 iterations, 50 ants

			Fastest (λ=0.9)		Safest (λ=0.1)	
			F_f	F_s	F_f	F_s
Constant λ	hCHAC	Best	**61.00**	244.90	74.00	27.30
		Mean	**66.42 ±3.29**	225.19 ±90.26	84.68 ±4.89	28.36 ±0.48
	hCHAC-4	Best	**66.00**	285.20	81.00	**28.00**
		Mean	**71.70 ±3.70**	316.66 ±58.73	98.13 ±15.99	**108.46 ±63.79**
	MOACS	Best	**64.00**	304.90	77.00	27.60
		Mean	**70.77 ±2.43**	294.66 ±79.44	93.60 ±6.92	29.23 ±0.67
	BiAnt	Best	**74.00**	256.00	116.50	**41.20**
		Mean	**100.27 ±16.71**	279.70 ±153.73	135.90 ±31.96	287.33 ±135.75
Variable λ	hCHAC	Best	64.50	235.30	72.00	**27.10**
		Mean	68.23 ±3.41	178.12 ±47.92	82.37 ±5.48	**28.14 ±0.54**
	hCHAC-4	Best	68.50	295.40	111.00	50.90
		Mean	81.40 ±10.01	302.24 ±46.05	101.22 ±19.55	212.00 ±56.22
	MOACS	Best	64.50	295.00	76.00	**27.50**
		Mean	71.67 ±2.90	244.90 ±61.14	91.00 ±6.67	**28.97 ±0.66**
	BiAnt	Best	75.50	316.00	139.50	43.50
		Mean	119.63 ±33.12	325.58 ±143.87	128.55 ±33.90	**272.38 ±150.70**

Table 3 λ study results for the four algorithms in PG-Mountains Map. 1500 iterations, 50 ants

			Fastest (λ=0.9)		Safest (λ=0.1)	
			F_f	F_s	F_f	F_s
Constant λ	hCHAC	Best	**74.36**	352.66	80.53	336.18
		Mean	**76.43 ±0.99**	352.39 ±8.98	81.66 ±2.49	354.61 ±11.86
	hCHAC-4	Best	**75.99**	365.25	82.75	360.59
		Mean	**84.33 ±5.81**	398.48 ±32.09	88.88 ±6.45	**395.40 ±29.07**
	MOACS	Best	**79.15**	378.63	85.31	**351.86**
		Mean	84.45 ±2.73	388.24 ±15.40	87.09 ±2.27	382.93 ±17.44
	BiAnt	Best	87.57	397.36	96.47	**415.56**
		Mean	**116.65 ±20.98**	528.56 ±96.95	138.06 ±26.57	620.65 ±117.80
Variable λ	hCHAC	Best	75.84	362.32	86.12	**308.06**
		Mean	78.45 ±1.53	340.12 ±15.50	83.43 ±2.12	**322.75 ±14.60**
	hCHAC-4	Best	82.93	404.62	87.66	**359.07**
		Mean	89.57 ±4.37	415.88 ±28.63	91.52 ±5.93	408.38 ±26.80
	MOACS	Best	80.23	370.26	89.42	354.34
		Mean	**83.99 ±2.18**	386.22 ±16.12	85.80 ±3.68	**376.42 ±13.45**
	BiAnt	Best	**86.58**	417.52	86.58	417.52
		Mean	124.29 ±28.31	570.14 ±135.73	124.39 ±28.36	**570.13 ±135.73**

In these tables, data is grouped mainly into two big columns, depending on the criteria with the higher priority (fastest or safest), so the cost function corresponding to this criteria is the most interesting (F_f in the fastest case, and F_s in the safest case) and the other one takes always worse values, since it corresponds to the secondary objective in the search.

Table 1 shows that results corresponding to the constant scheme are better in general; the best cost only slightly, but clearly on average. In general, the costs associated to the preferred objective are better in the constant scheme, sometimes the best solutions are the same (as in Table 1 for hCHAC), but the mean and standard deviation demonstrate that they are worse.

There is a fact to point out, the mean results in the variable scheme for the objective which is not being minimised (the less important), are better than in the constant

case. It is reasonable since in this case, the ants search in the whole space of solutions, yielding good solutions not only in one of the objectives, but in both of them. This should be interesting in most MO problems, where the aim is finding a good solution in all the objectives, or yielding as much solutions in the PF as possible, but in the MUPFP-2C, the user just want one solution in each case (or a set of solutions) which minimises the most important objective. So the best option would be the constant scheme.

In the hCHAC-4 case, the differences are more remarkable looking at the best solutions and much more remarkable looking at the means. The reason is that considering four objectives to minimise, makes the search space bigger, so it is quite difficult to get a good set of solutions in the same time. In fact, it is more complicated to yield good solutions considering just one objective (the preferred one), if the algorithm explores the whole space of solutions (variable scheme), than if the algorithm search just in a concrete area (constant scheme). But again, better solutions for the secondary objective are obtained.

Relating to MOACS, there are small differences between the results of both schemes, being a bit better the constant approach, except in one case in which the mean for the variable configuration is better, but the standard deviation is worse, so in an averaged sense, results favours again to the constant scheme.

BiAnt shows stronger differences favouring to the constant configuration too, due to the higher exploration factor associated to the algorithm (since it is an AS), as can be seen in the high standard deviation of its results. So this approach takes advantage of the extra exploitation factor that adds the constant λ scheme.

Looking at the Table 2 results, they are quite similar to those commented (on the previous table), being better in general for the constant configuration. But this time there are some exceptions in which the variable scheme yields better solutions, even considering the mean and standard deviation (MOACS case). This happens always in the F_s cost for the searching of the safest paths, since there are just a few number of safe paths, all of them moving in a concrete area of the map, since both enemies are watching over the greater part of the scenery. So just the left-side zone, which is far from them, and behind some forest patches and hills is a safe zone. This way, the most of the safest solutions move through that area (they are quite similar), and the variable scheme explores some more possibilities inside the good ones.

The third map (which results are showed in Table 3) is an special case, since there are no known enemies on it. So both, the fastest and the safest paths, are quite straight from the origin to the target point (moving through the most hidden cells, to their environment, in the safest case). This means that in the constant scheme just a small area in the search space is explored (the one surrounding the most straight solution), but in the variable approach, there is a higher exploration around this straight zone, which yields better solutions on average when the algorithms search for safe solutions. This map has been included in the study to show that the application of a variable scheme could be better in maps with a 'very restricted' set of solutions, where there are a small zone of the space of solutions to explore due to the problem definition.

7 Conclusions and Future Work

In this paper, two different schemes of application for the parameter which sets the relevance of two objectives, in four MOACOs designed to solve the military unit path-finding problem, considering speed and safety, have been analysed. This parameter is known as λ, and it is applied in the State Transition Rule of the algorithms.

The *variable scheme* consists in assign a different value to each of the ants in the algorithm in order to search in the whole space of solutions (every ant searches in an area). The other configuration, named *constant scheme*, determines in advance one value which is used by all the ants, so they search in the same area of the space of solutions. Some experiments have been performed over three realistic maps and the general conclusion reached is the constant approach yields better solutions most of times, following the user (of the application which applies the algorithms) criteria.

In general the use of the constant λ scheme implies a higher exploitation of solutions (since all the ants search in a concrete area). On the other hand, the variable λ scheme adds an exploration factor to the algorithms, since each ant searches in a different area of the space of solutions, yielding different solutions (better sometimes) and a bigger Pareto Set (in the ideal case). The second approach would be better for most of the multi-objective algorithms, to solve ordinary or common multi-objective problems, since the aim is to find as much solutions as possible (the biggest PS). But in the problem addressed in this work, the aim is to get the best solution considering a set priority for both objectives, so the constant scheme has demonstrated to be better. This way, the two algorithms taken from the literature and adapted to solve this problem (MOACS and BiAnt), show that the constant configuration yields better results in this case, but they were defined considering a variable approach, since their aim were to solve general MO problems.

There are some ideas as future work in this line. The first one is to test both schemes in some other problems, considering the best set of values for the other parameters in each one of the algorithms, since some of them should show some differences between the results yielded by both schemes when they have a correct exploitation factor, as BiAnt case. Another approach consists in the implementation of the proposed algorithms to solve common MO problems. In this case, we will perform a new experimental set to determine the best λ application scheme.

Finally, we would like to implement the auto-evaluation of the scenario to fix the best set of values before the running of the algorithm. Also including the decision of the most appropriate scheme for applicating λ.

Acknowledgements

This paper has been funded in part by the Spanish MICYT projects NoHNES (Spanish Ministerio de Educación y Ciencia - TIN2007-68083) and TIN2008-06491-C04-01 and the Junta de Andalucía P06-TIC-02025 and P07-TIC-03044.

References

[1] Mini-Simulator hCHAC (2008), http://forja.rediris.es/frs/download.php/1355/mss_chac.zip
[2] Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: IASTED International Multi-Conference on Applied Informatics. IASTED IMCAI, vol. 21, pp. 97–102 (2003)
[3] Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Dordrecht (2002)
[4] Dorigo, M., Stützle, T.: The ant colony optimization metaheuristic: Algorithms, applications, and advances. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 251–285. Kluwer, Dordrecht (2002)
[5] García-Martínez, C., Cordón, O., Herrera, F.: An empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 61–72. Springer, Heidelberg (2004)
[6] Iredi, S., Merkle, D., Middendorf, M.: Bi-criterion optimization with multi colony ant algorithms. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 359–372. Springer, Heidelberg (2001)
[7] Mora, A., Merelo, J., Laredo, J., Castillo, P., Sánchez, P., Sevilla, J., Millán, C., Torrecillas, J.: hCHAC-4, an ACO algorithm for solving the four-criteria military path-finding problem. In: Krasnogor, N., Nicosia, G., Pavone, M., Pelta, D. (eds.) Proceedings of the International Workshop on Nature Inspired Cooperative Strategies for Optimization. NICSO 2007, pp. 73–84 (2007)
[8] Mora, A., Merelo, J., Castillo, P., Laredo, J., Cotta, C.: Influence of parameters on the performance of a moaco algorithm for solving the bi-criteria military path-finding problem. In: WCCI 2008 Proceedings, pp. 3506–3512. IEEE Press, Los Alamitos (2008)
[9] Mora, A.M., Merelo, J.J., Millán, C., Torrecillas, J., Laredo, J.L.J., Castillo, P.A.: Comparing aco algorithms for solving the bi-criteria military pathfinding problem. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 665–674. Springer, Heidelberg (2007)
[10] Mora, A.M., Merelo, J.J., Millán, C., Torrecillas, J., Laredo, J.L.J., Castillo, P.A.: Enhancing a MOACO for solving the bi-criteria pathfinding problem for a military unit in a realistic battlefield. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 712–721. Springer, Heidelberg (2007)