Comparing ACO Algorithms for Solving the Bi-criteria Military Path-Finding Problem*

Antonio M. Mora[1], Juan J. Merelo[1], Cristian Millán[2], Juan Torrecillas[2], Juan L.J. Laredo[1], and Pedro A. Castillo[1]

[1] Departamento de Arquitectura y Tecnología de Computadores. University of Granada (Spain)
{amorag,jmerelo,juanlu,pedro}@geneura.ugr.es
[2] Mando de Adiestramiento y Doctrina. Spanish Army
{cmillanm, jtorrelo}@et.mde.es

Abstract. This paper describes and compares mono- and multi-objective Ant Colony System approaches designed to solve the problem of finding the path that minimizes resources while maximizing safety for a military unit in realistic battlefields. Several versions of the previously presented CHAC algorithm, with two different state transition rules are tested. Two of them are *extreme* cases, which only consider one of the objectives; these are taken as baseline. These algorithms, along with the Multi-Objective Ant Colony Optimization algorithm, have been tested in maps with different difficulty. hCHAC, an approach proposed by the authors, has yielded the best results.

1 Introduction and Problem Description

The commander of a military unit in the battlefield must consider two main criteria before deciding on the best path to a destination point: the speed (important if the unit mission requires arriving as soon as possible to the target) and the safety (important when the enemy forces are not known or when the unit effectives are very valuable). However, in any situation, both objectives must be considered. This problem is called *military unit path-finding problem* and can be formally defined as: finding the best path for a military unit, from an origin to a destination point in the battlefield, keeping a balance between route speed and safety, considering the presence of enemies (which can shoot against the unit) and taking into account realistic properties and restrictions.

We model this problem considering that the unit has a *level of energy (health)* and a *level of resources*, which are consumed when it moves through the path, so the problem objectives are adapted to minimize the consumption of resources and energy.

The battlefield is also modelled as a grid of hexagonal cells where every one has assigned a *cost in resources* which represents the difficulty of going through it, and a *cost in energy* which means that the unit depletes its human resources or

* Supported by NadeWeb (TIC2003-09481-C04-01) and PIUGR (9/11/06) projects

F. Almeida e Costa et al. (Eds.): ECAL 2007, LNAI 4648, pp. 665–674, 2007.

that vehicles suffer damage when crossing over the cell (*no combat casualties*). Both costs depend on the cell type. In addition there are other costs: one in resources if the unit moves between cells with different heights (more if it goes uphill) and other in energy, *lethality*, which is the damage that a cell could produce due to enemy weapons impact. These features are resumed in Table 1.

Table 1. Energy and Resources description.

	Composed by	*Consumed by*
Energy	global soldiers health, global vehicles status	'no combat' casualties (injuries, tiredness), lethality
Resources	food, fuel, medicines, general supplies, moral	going through difficulty, height difference

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). *Safe* paths, on the other hand, have a low cost in energy.

The cells also have a type, subtype and height associated, plus some properties (for instance, whether they are or not inside the realistic line of sight of the enemy) and restrictions (such as visible distance or maximum height difference that the unit can go trough).

Initially, we solved the problem by implementing the CHAC algorithm [1], and later we improved it with Hexa-CHAC [2]. More details about the problem definition, restrictions and description and test of the algorithms are shown in these articles.

In this work, we have implemented some new algorithms and tested them in the same maps (battlefields) which will allow us to check which algorithm works the best in each circumstance, and to establish baselines for the performance in each of the objectives we are going to optimize. These algorithms are described in the next section.

2 Algorithms Tested in This Paper

In this work, we have tested two MOACOs (Multi-Objective Ant Colony Optimization algorithms [3]): *hCHAC* [2] (with two different state transition rules -STRs from now on-, and extreme values of λ) and *MOACS* [4]. Additionally, we also test a mono-objective approach which combines both objectives in a single aggregative function. So, there are six algorithms in all. All of them are Ant Colony System algorithms [5,6], so the problem is transformed into a graph where each node corresponds to a cell in the map and an edge between two nodes is the connection between neighbour cells in the map. Every edge has two weights associated which are the costs in resources and energy that going through that edge causes to the unit.

Every iteration, ants separately build a complete path (solution), between origin and destination points (if possible), by travelling through the graph. To

guide this movement ants use a STR which combines two kinds of information: pheromone trails and heuristic knowledge. We use ACSs to have better control in the balance between exploration and exploitation by using the characteristic parameter q_0.

The problem we want to solve is a multi-objective (MO) one (see a description of MO problems and algorithms in [7]) with two independent objectives to minimize. These objectives are named $\boldsymbol{f}$, minimization of the resources consumed in the path (fast path or speed maximization) and $\boldsymbol{s}$, minimization of the energy consumed in the path (safe path or safety maximization). That is why ***hexa-CHAC*** (*hCHAC* from now on)[2] is an ACS adapted to deal with two objectives, and uses two pheromone matrices (τ_f,τ_s) and heuristic functions (η_f,η_s) (one per objective), a single colony, and two STRs: (*Combined State Transition Rule, CSTR*), similar to the one proposed in [8], which combines the pheromone and heuristic information for each objective weighted using α, β and λ parameters; and (*Dominance State Transition Rule, DSTR*), which ranks neighboring cells according to how many they dominate [1]. These rules use the parameter $\lambda \in (0,1)$, which is user-defined, and sets the importance of the objectives in the search (which one has the highest 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, it has to be close to 0. This value is constant during the algorithm for all ants, so *hCHAC* searches always in the same zone of the space of solutions (the zone related to the chosen value for λ).

The local and global pheromone updating formulae [1] are based in the MACS-VRPTW algorithm proposed in [9,4], 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).

On the other hand, ***Mono-hCHAC*** is an ACS that combines the two previous objectives in one. It uses formulae similar to those of *hCHAC* (heuristic, pheromone updating and evaluation function (see in [1])), but only one in each case and adapted to consider only one objective (by including specific terms for each objective).

The *Heuristic Function* is as follows:

$$\eta(i,j) = \frac{\omega_r}{C_r(i,j)} + \frac{\omega_e}{C_e(i,j)} + \frac{\omega_d}{Dist(j,T)} + (\omega_{zo} \cdot ZO(j)) \qquad (1)$$

Where C_r and C_e are respectively the cost in resources and energy when moving from node i to node j, *Dist* is the Euclidean distance between two nodes (T is the target node of the problem) and ZO is a score (between 0 and 1) for a cell, being 1 when the cell is hidden to all the enemies (or to all the cells in a radius when there are no enemies) and decreasing exponentially when it is seen. ω_r, ω_e, ω_{fd} and ω_{fo} are weights that assign relative importance to the terms in the formula. The values for the two first are the same as in the *hCHAC* formulas, and the values for the two last have been calculated as an average of the correspondent parameters in those formulas. So, all terms are important.

The *STR* which guides the search is the typical formula in mono-objective ACSs:
If $(q \leq q_0)$

$$j = \arg\max_{j \in N_i} \left\{ \tau(i,j)^{\alpha} \cdot \eta(i,j)^{\beta} \right\} \tag{2}$$

else

$$P(i,j) = \begin{cases} \dfrac{\tau(i,j)^{\alpha} \cdot \eta(i,j)^{\beta}}{\sum_{u \in N_i} \tau(i,u)^{\alpha} \cdot \eta(i,u)^{\beta}} & if\ j \in N_i \\ 0 & otherwise \end{cases} \tag{3}$$

where $q_0 \in [0,1]$ is the standard ACS parameter, q is a random value in [0,1]. τ is the pheromone trails matrix and η is the heuristic function (Equation 1). α and β are the usual weighting parameters and N_i is the current feasible neighbourhood for the node i.

This STR works as follows: when an ant is building a path and is placed at one node i, a random number q in [0,1] is generated, if $q \leq q_0$ the best neighbour j is selected as the next node in the path (Equation 2). Otherwise, the algorithm decides the next one by using a roulette wheel considering *P(i,j)* as probability for every feasible neighbour j (Equation 3).

The *Local Pheromone Updating* is performed when a new node j is added to the path that an ant is building:

$$\tau(i,j) = (1-\rho) \cdot \tau(i,j) + \rho \cdot \tau_0 \tag{4}$$

where ρ in [0,1] is the common evaporation factor and τ_0 is the initial amount of pheromone in every edge:

$$\tau_0 = \frac{1}{n_c \cdot ((MAX_R + MAX_E)/2)} \tag{5}$$

with n_c as the number of cells in the map to solve, MAX_R as the maximum amount of resources going through a cell may require, and MAX_E as the maximum cost in energy that going through a cell may produce.

The *Global Pheromone Updating* is performed at the end of every iteration of the algorithm, once all the ants have built a solution path:

$$\tau(i,j) = (1-\rho) \cdot \tau(i,j) + \frac{\rho}{F_{fs}} \tag{6}$$

F_{fs} is the *Evaluation Function* which assigns a global cost value to every solution found by each ant. It considers the cost in resources, the cost in energy, and the visibility of each node (cell) in the path:

$$F_{fs}(Psol) = \sum_{n \in Psol} [Cr(n-1,n) + Ce(n-1,n) + \omega_{zo}^{F} \cdot (1 - ZO(n))] \tag{7}$$

where *Psol* is the solution path to evaluate and ω_{zo}^{F} is the weight which sets the importance of visibility of the cells in the path. The other terms are the same as in Equation 1.

MOACS was proposed by Baran et al. [4], as a variation of the MACS-VRPTW introduced by Gambardela et al. in [9], the main difference being the use of a single pheromone matrix for both objectives (instead of one per objective). We have adapted it to solve this problem, so we use the same heuristic and evaluation functions (see in [1]), but different STR and pheromone updating formulas. The STR is similar to the *hCHAC* CSTR, but using only one pheromone matrix (as we previously said). It is defined as follows:
If $(q \leq q_0)$

$$j = \arg\max_{j \in N_i} \left\{ \tau(i,j) \cdot \eta_f(i,j)^{\beta \cdot \lambda} \cdot \eta_s(i,j)^{\beta \cdot (1-\lambda)} \right\} \tag{8}$$

else

$$P(i,j) = \begin{cases} \dfrac{\tau(i,j) \cdot \eta_f(i,j)^{\beta \cdot \lambda} \cdot \eta_s(i,j)^{\beta \cdot (1-\lambda)}}{\sum_{u \in N_i} \tau(i,u) \cdot \eta_f(i,u)^{\beta \cdot \lambda} \cdot \eta_s(i,u)^{\beta \cdot (1-\lambda)}} & if\ j \in N_i \\ 0 & otherwise \end{cases} \tag{9}$$

where τ is the pheromone matrix, η_f and η_s are the heuristic functions for the objectives, and the rest of terms and parameters are the same as in Equation 3. This rule also uses λ to set the importance of the objectives in the search and takes a constant value during the algorithm for all ants, unlike the original proposal of Baran et al. [4] in which the parameter takes a value of 0 for the first ant, and it grows for every ant until it takes a value of 1 for the last one. The reason is that the algorithm must yield solutions following the user desires (like *hCHAC*), which means solutions in a concrete zone of the solutions space. The rule works as we previously explain.

Since *MOACS* is an ACS, there are two levels of pheromone updating, local and global. The equation for *Local Pheromone Updating* is:

$$\tau(i,j) = (1-\rho) \cdot \tau(i,j) + \rho \cdot \tau_0 \tag{10}$$

considering:

$$\tau_0 = \frac{1}{n_c \cdot MAX_R^{\lambda} \cdot MAX_E^{(1-\lambda)}} \tag{11}$$

with the same parameters and terms as in Equations 4,5, but also weighted using λ parameter to consider again the relative importance of each objective set by the user.

There is a reinitialization mechanism, so the value of τ_0 is not fixed during the algorithm run, as usual in ACS, but it undergoes adaptation. Every time an ant h builds a complete solution, it is compared to the Pareto set P generated until now to check if the former is a non-dominated solution. At the end of each iteration, τ_0' is calculated following the formula:

$$\tau_0' = \frac{1}{n_c \cdot \bar{C}r(PS)^{\lambda} \cdot \bar{C}e(PS)^{(1-\lambda)}} \tag{12}$$

where $\bar{Cr}$ and $\bar{Ce}$ are respectively the average consumption of resources and energy for the solution paths currently included in the Pareto set.

Then, if $\tau_0' > \tau_0$, the current initial pheromone value, the pheromone trails are reinitialized considering the new value for $\tau_0 \leftarrow \tau_0'$. It means a better Pareto Set has been found.

Otherwise the Global Pheromone Updating is made for every solution in the Pareto Set:

$$\tau_f(i,j) = (1-\rho) \cdot \tau_f(i,j) + \frac{\rho}{F_f \cdot F_s} \tag{13}$$

where F_f and F_s are the evaluation functions for each objective.

We will also introduce what we call ***Extreme hCHAC*** (in two versions, *extremely fast* and *extremely safe*) in this section to include them as baseline for comparison with the other algorithms. And both of them have been tested using CSTR and DSTR.

The *extremely fast hCHAC* only considers the minimization of resources consumption objective (speed), so λ is set to 1 and all the weights related to safety objective, such as visibility of cells or energy consumption, take values equal to 0. On the other hand, the *extremely safe hCHAC* only takes into account the minimization of energy consumption objective (safety), so λ is set to 0 and all the weights related to speed objective, such as distance to target point or resources consumption, also take values equal to 0. From now on, we will refer to both approaches as *extr-hCHAC*.

3 Experiments and Results

We have performed experiments in two of the same 45x45 cell realistic maps used in a previous paper [2], which are part of 2 Panzer General^{TM} game maps. All the algorithms presented in the previous section have been run in these maps using the same parameter values (except in extreme approaches, of course), namely: α=1, β=2, ρ=0.1 and q_0=0.4. We have used different values for λ parameter: 0.9 and 0.1 to consider one objective with higher priority than the other, and 1 and 0 in extreme approaches. The mono-objective implementation does not use this parameter.

All the 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 only consider one (chosen by the military staff considering their own criteria and the features of each problem). The mono-objective approach yields a single solution which is evaluated using the same functions as in *hCHAC* and *MOACS* in order to obtain a multi-objective valuation which can be compared with the solutions of the other algorithms and methods.

We have made 30 runs per scenario, using each algorithm or method (STRs) and using each value for λ: 0.9 and 0.1 for *hCHAC-CSTR, hCHAC-DSTR, mono-hCHAC* and *MOACS*, to find the fastest and safest path (but considering the other criteria too); and 0 and 1 for *extr-hCHAC-CSTR* and *extr-hCHAC-DSTR*

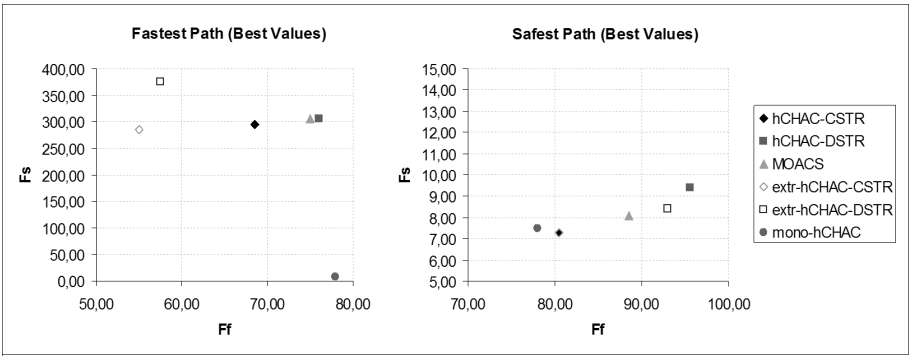

Fig. 1. Best results yielded by all algorithms for Map 1: speed (right) and safety (left) optimization.

Table 2. Results for Map 1. (1500 iterations, 50 ants)

		Fastest (λ=0.9)		Safest (λ=0.1)	
		F_f	F_s	F_f	F_s
hCHAC-CSTR	Best	68.50	295.40	80.50	**7.30**
	Mean	75.20 ±7.87	184.54 ±132.49	85.00 ±3.32	8.10 ±0.49
hCHAC-DSTR	Best	76.00	306.10	95.50	9.40
	Mean	81.63 ±3.02	271.11 ±39.98	108.00 ±5.70	10.40 ±0.52
MOACS	Best	75.00	306.00	88.50	8.10
	Mean	82.95 ±5.13	256.20 ±90.02	103.52 ±5.56	9.50 ±0.70
		Extreme Fast (λ=1)		Extreme Safe (λ=0)	
		F_f	F_s	F_f	F_s
extr-hCHAC-CSTR	Best	**55.03**	285.50	80.50	**7.30**
	Mean	**58.73 ±1.79**	309.53 ±27.63	84.07 ±3.56	**7.89 ±0.56**
extr-hCHAC-DSTR	Best	57.54	375.60	93.00	8.40
	Mean	63.63 ±2.45	329.29 ±38.09	106.90 ±5.85	10.22 ±0.65
mono-hCHAC	Best	78.00	7.50		
	Mean	85.63 ±3.68	8.41 ±0.43		

in order to find the fastest and safest path respectively (without consider the other objective in each case).

Finally, statistical t-Student tests have been used to evaluate obtained results and to test whether differences among means are significant.

In *the first map* there is a single unit on watch between the origin and target point of the unit. There are some hidden zones, but the enemy controls the area the unit must pass in order to get to the target as soon as possible. It has associated a medium difficulty from the military point of view. Results for this map are shown in Table 2 and Fig. 1.

Fig. 1 shows that extreme approaches results can be taken as reference, being always the best in the objective which the highest priority, because they do not consider the other objective at all (they have a high cost value for that objective). We can see that *extr-hCHAC-CSTR* also yields a good result for cost in the non-considered objective, even better than those of the other approaches. The reason is that, in this map, a fast solution can be also a safe one (if the path

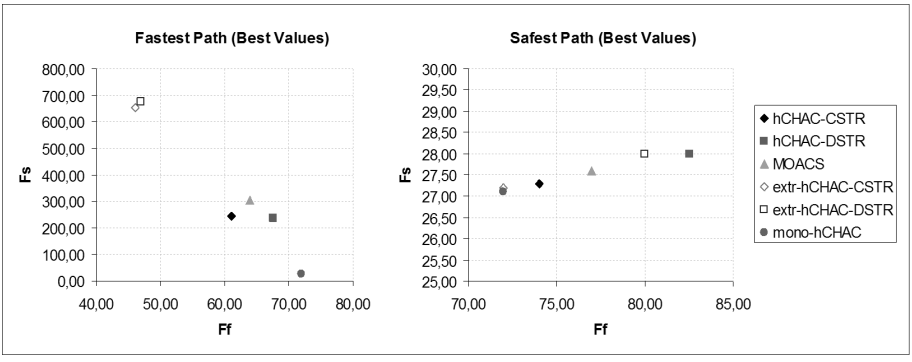

Fig. 2. Best results yielded by all the approaches for Map 2 searching for fastest (left) and safest (right) path.

moves hidden to the enemy) and vice versa. DSTR implementations yield worse results than CSTR ones, because this STR needs a higher exploitation factor (is more explorative) to get similar results. So, even the extreme *extr-hCHAC-DSTR* approach yields worse best solution than some others for the main objective. In the analysis of the main algorithms, it can be seen that *hCHAC-CSTR* yields very good solutions, close to the extreme ones (even equal in the safest case) and, considering the dominance concept, it always yields solutions only dominated by extreme CSTR. DSTR approach is worse for the reason we explain above. *MOACS* also yields good solutions, better than DSTR methods ones (except in fastest path search), but worse than *hCHAC-CSTR* results.

mono-hCHAC is a special case, because it only yields one solution which combines speed and safety. We have represented this solution considering the same costs yielded by it in both searches. Both cost are quite good (low), but they are not the best in comparing with the other approaches, when they search for the fastest or the safest path.

Table 2 shows that all approaches yield results with a low standard deviation in the prioritary objective, which implies robustness (its solutions are similar between runs). Best, mean and standard deviation in the secondary objective are logically worse, because it has little importance. But in this case the differences between the security cost (F_s) when it is prioritary and secondary objective are enormous. The reason is fast paths are usually unsafe due to visibility of the cells, and in this experiments we penalize much the term of visibility in the cost function. Results were verified using t-Student statistical tests. Significant differences were found and the confidence level is 99%.

In *Map 2* there are two enemy units between the origin and target point of the unit. One of them is just watching over and the other one, which is nearer to the target point and in the middle of the straight path, is watching over and firing to some strategic points (some bridges) and in a zone surrounding itself. There are little zones where the unit can hide (slight patches of forest). It has

Table 3. Results for Map 2. (1500 iterations, 50 ants)

		Fastest (λ=0.9)		Safest (λ=0.1)	
		F_f	F_s	F_f	F_s
hCHAC-CSTR	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-DSTR	Best	67.50	235.60	82.50	28.00
	Mean	72.92 ±2.63	236.97 ±42.74	95.93 ±7.25	29.43 ±0.72
MOACS	Best	64.00	304.90	77.00	27.60
	Mean	70.77 ±2.43	294.66 ±79.44	93.60 ±6.93	29.23 ±0.68
		Extreme Fast (λ=1)		Extreme Safe (λ=0)	
		F_f	F_s	F_f	F_s
extr-hCHAC-CSTR	Best	**46.04**	654.60	72.00	27.20
	Mean	**49.92 ±2.29**	467.30 ±172.03	80.64 ±4.52	**28.05 ±0.45**
extr-hCHAC-DSTR	Best	47.04	674.70	80.00	28.00
	Mean	53.02 ±2.58	403.61 ±130.55	96.35 ±8.16	29.60 ±0.79
mono-hCHAC	Best	72.00	**27.10**		
	Mean	78.33 ±4.24	52.23 ±42.97		

associated a medium-hard difficulty from the military point of view. The results for the different approaches are show in Table 3 and Figure 2.

Fig. 2 shows that *extr-hCHAC-CSTR* yields the best results in both searches, but only in the main objective. In fact, in the fastest path search there is an overwhelming increase of the cost in energy (or safety) F_s, since that path goes through cells affected by enemy weapons. *hCHAC-CSTR* yields solutions dominated only by those of *extr-hCHAC-CSTR* and *mono-CHAC* (see below) in the safest case. DSTR approach is again bad for the reason we explain above. *MOACS* again yields good solutions, better than DSTR methods ones (except in fastest path search), but worse than *hCHAC-CSTR* results.

mono-CHAC results yield the best safety result this time, which might seem odd, however, due to the stochastic nature of these approaches, it is possible to find a very good solution sometimes; average solutions are not as good, as shown by the high standard deviation. As a result, the cost in resources F_f is high, which happen sometimes in difficult maps like this one.

Table 3 shows results similar to the previous experiment, with a higher increasing of cost in F_s in the search for fastest path, due to the lethality of the cells. Again, significant differences were found applying t-Student tests, obtaining a confidence level of 95 or 99%.

4 Conclusions and Future Work

In this work we have tested and compared six different ACS approaches to solve the bi-criteria military path-finding problem (find the best path considering speed and safety as objectives), finding that *hCHAC-CSTR* is the best approach, maintaining a good balance between speed and safety in all cases, and considering always both objectives (with different priorities, depending on the search). It yields better results than extreme cases (where only one of the objectives is considered); even more so in difficult maps, where the cost for the objective not being minimized can increase dramatically. *MOACS* and *mono-objective* algorithms yield good solutions too, generally better than hCHAC-DSTR approach,

but worse than those obtained by hCHAC-CSTR. Differences between methods have been proved significant after the application of t-Student tests.

As future work and following the same researching line, we are going to implement some other algorithms in order to include them in the comparison and in the study of the best approach for this problem. We also are going to make other studies (statistical, for instance) to have more criteria to evaluate the performance of every approach. Besides, we will also try to separate completely all objectives: speed, safety, visibility, and distance to objective (in each step), using more pheromone matrices, but taking into account each problem separately into a truly multi-objective approach.

References

1. Mora, A.M., Merelo, J.J., Millán, C., Torrecillas, J., Laredo, J.L.J.: CHAC. a MOACO algorithm for computation of bi-criteria military unit path in the battlefield. In: Pelta, D.A., Krasnogor, N. (eds.) Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization. NICSO'2006, June 2006, pp. 85–98 (2006)
2. 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. Applications of Evolutionary Computing. LNCS, vol. 4448, pp. 712–721. Springer, Heidelberg (2007)
3. 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)
4. Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: IASTED International Multi-Conference on Applied Informatics. Number 21 in IASTED IMCAI, 97–102 (2003)
5. 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 Academic Publishers, Dordrecht (2002)
6. Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 11–32. McGraw-Hill, New York (1999)
7. Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Dordrecht (2002)
8. 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)
9. Gambardella, L., Taillard, E., Agazzi, G.: Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 73–76. McGraw-Hill, New York (1999)