Balancing Safety and Speed in the Military Path Finding Problem: Analysis of Different ACO Algorithms.*

A.M. Mora
Dpto. Arquitectura y Tecnologia de Computadores
18072 ETSIIT
Granada, Spain
amorag@geneura.ugr.es

J.J. Merelo
Dpto. Arquitectura y Tecnologia de Computadores
18072 ETSIIT
Granada, Spain
jmerelo@geneura.ugr.es

J.L.J. Laredo
Dpto. Arquitectura y Tecnologia de Computadores
18072 ETSIIT
Granada, Spain
juanlu@geneura.ugr.es

P.A. Castillo
Dpto. Arquitectura y Tecnologia de Computadores
18072 ETSIIT
Granada, Spain
pedro@geneura.ugr.es

C. Millan
Mando de Adiestramiento y Doctrina (ETE)
Acuartelamiento Las Descalzas
Granada, Spain
cmillan@et.mde.es

J. Torrecillas
Mando de Adiestramiento y Doctrina (ETE)
Acuartelamiento Las Descalzas
Granada, Spain
jtorrelo@et.mde.es

ABSTRACT

hCHAC, a MOACO implemented to solve the problem of finding the path that minimizes resources, while maximizing safety for a military unit in realistic battlefields, is compared with some other approaches: two extreme methods, which only considers one objective in the search, and a mono-objective algorithm, which combines the two objectives terms of the formulae in a single. In addition, two state transition rules (combined and dominance-based) have been used in some of the approaches. All of them have been tested in different difficulty maps and hCHAC using the combined state transition rule has been considered the best approach.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.6.5 [**Computing Methodologies**]: Model Development—*Modeling methodologies*

General Terms

Algorithms

Keywords

Ant Colony Optimization, Multiobjective, Pathfinding, Military problems

*This work has been supported by the NADEWeb project (TIC2003-09481-C04).

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

1. INTRODUCTION

Within a battlefield, the choice of the best path for a military unit must be made taking into account two main criteria: speed and safety. A safe path is followed when the situation of the enemy forces is not known, so the unit must move through hidden zones in order to avoid detection. On the other hand, a fast path is followed when the unit priority is get to the destination point as soon as possible because of requirements of its mission (for example, strong necessity of resources at destination point). The other criteria must also be taken into account in both cases: in the safe case there is a time limit, and in the fast case the unit must get to destination with enough personnel and supplies to accomplish its mission.

In this work, we have developed an Ant Colony Optimization algorithm [3] adapted to deal with a bi-criteria problem called the *Military Unit Pathfinding Problem*. This problem tries to find the best path from an origin to a destination point for a military unit in a battlefield taking into account two different criteria: *safety* and *speed*. Since the unit has a level of energy/health (the sum of the status of vehicles and health of the soldiers of the unit) and a level of resources (supplies, fuel and food, for example), the problem tries to find the path which has a desirable level of associated consumption in both levels. So, we consider that a safe path corresponds to one with low energy requirements, and a fast path one with a low resource cost.

The battlefield of our problem is modelled as a realistic scenario, and is composed by a grid of hexagonal cells overlaid with an information layer, which assigns to every cell a type of terrain, height and depth. In addition, there may be some enemy units watching over and/or shooting their weapons against the unit or to strategic points in the battlefield. Fig. 1 shows an example of real world battlefield and the information layer associated to it.

Every cell corresponds to a 500x500 meters zone in the real world (the same as a real deployed unit), and there are four types of terrain: normal (sand), forest, water and obstacle. In addition, there are two penalization levels associated with

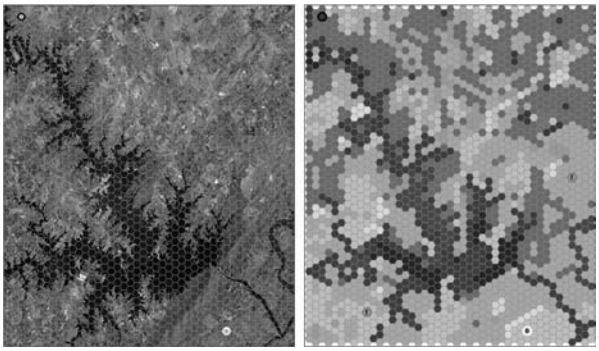

Figure 1: Example Map (45x45 cells). Right image is a real world picture of a lake surrounded by some hills and lots of vegetation. Left image shows its associated information layer, where it can be seen the types corresponding to the same hexagons in the other image, so there are many water and forest cells and some normal terrain cells. The different shades in the same color models height (light color) and depth (dark color). There are two enemies labelled with 'E', an origin point (in the top-left corner of the images) labelled with 'O' and a destination point (in the bottom-right) labelled with 'D'. These labels are black on the image at right and white on the left.

every cell concerning to the consumption in energy/health and resources which implies going through it. The enemy weapons impact is considered as an extra (and higher) penalization in energy/health, with different levels depending on the shooting direction and distance. It is called *lethality*.

Finally, there are some restrictions in the problem such as: enemies and problem units have an *acquisition capability* associated, which is the longest distance that they can see (to locate other units); in addition, the problem unit cannot cross artificial obstacles cells and cannot move between two cells with a difference in height greater than 2 (natural obstacles).

We have developed an application (using Delphi 7) to create and edit battlefields, and also to execute the algorithms and visualize the solutions easily.

Initially, we implemented the CHAC algorithm [7], and later we improve it with Hexa-CHAC [8]. 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 whith algorithm works the best and to establish baselines for the performance. These algorithms have been implemented to be used by military staff, so the user can assign a priority level to one objective over the other, depending on the target of the mission.

2. ACOS IMPLEMENTED

Our main algorithm is CHAC, whose name means *Compañía de Hormigas ACorazadas* (Armored Ant Company) to relate the ACO algorithms with the military scope. Hexa-CHAC (or hCHAC) is an Ant Colony System (ACS) [4] adapted to deal with several objectives, that is, a Multi-objective Ant Colony Optimization (MOACO) [2, 5] algorithm. The Hexa prefix refers to the topology of the search space (a grid of hexagons).

In order to use an ACO algorithm, the problem must be transformed into a graph where every node corresponds to a cell in the scenario (map) and an edge between two nodes is the existent connection between neighbour cells. Every edge has associated two weights (one for each objective of the problem) which are the costs in resources and energy that going through that edge causes to the unit.

The two objectives are named $\boldsymbol{f}$, minimization the resources consumed in the path (speed maximization) and $\boldsymbol{s}$, minimization the energy consumed in the path (safety maximization).

The algorithm is a combination of two others: Iredi's Bi-Criterion Ant Algorithm [6] and Baran's MOACS [1]. It uses two pheromone matrices and two heuristic functions each pair dedicated to one objective, and a single colony. We use an Ant Colony System (ACS) to have better control in the balance between exploration and exploitation by using the parameter q_0 in [0,1], whose value have high influence in the choice of the next node in the path that an ant is building, so if it is close to 0, all the nodes have possibilies of be chosen, on the other hand if it is close to 1, surely the best node will be chosen. We have implemented two state transition rules: the first one is similar to the proposed by Iredi et al. in [6] (*Combined State Transition Rule, CSTR*) and the second one, introduced in [7], based on dominance of neighbours (*Dominance State Transition Rule, DSTR*).

The local and global pheromone updating formulas are based in the proposal of Baran et al. [1], with some changes due to the use of two pheromone matrices. So only the solutions in the Pareto set make the global updating.

The heuristic functions have been designed respectively to search for the fastest path (trying to minimize the consumption of resources and the distance to target point), and to search for the safest path (assigning priority to minimization of energy consumption and visibility of the cell from the enemies point of view). They both use some weights to assign a priority to every term in the formula.

The evaluation functions assign a cost to every solution path. Again, there are one function per objective which are named F_f (cost in resources) and F_s (cost in energy/health). These functions also use weights in their terms.

All these formulas can be consulted in the previous paper [7].

In this work, we have also implemented a mono-objective ACS approach (as a new development). This is a classical ACS [4], so it uses a single ant colony as well as one pheromone matrix and one heuristic function. There is a local pheromone updating in every edge that an ant follows, while builds a solution, and a global updating (at the end of every iteration) performed in the edges of the global best solution. The heuristic and the evaluation functions have been designed by merging each pair of the same functions in hCHAC, in order to take into account and evaluate the two criteria in the search. We call this algorithm *mono-hCHAC* from now on.

In addition we will test two extreme approaches of hCHAC, *extremely fast* and *extremely safe*. The first one 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 second one

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. They will be described in next section. From now on, we will refer to both approaches as *extr-hCHAC*.

3. EXPERIMENTS

We have performed experiments in two different and realistic scenarios, which have been captured from the game Panzer General^{TM}and modelled using our application.

In these experiments the problem and the enemies units have associated an acquisition capability of 18 cells (which corresponds to 9 Km in the real world); the unit cannot going through cells with a difference in height greater than 2; and finally, we consider that the problem unit has enough points of energy and resources to follow any path in the scenario.

Relating to the parameters of the algorithm, we have used their common values (α=1, β=2, ρ=0.1), excepting q_0=0.4 which determines more exploitation as usual. On the other hand, we have used two sets of weights in heuristic and evaluation functions: weights in the first set, take values to tend to safe paths (low consumption of energy and low visibility) in the terms of the functions related with the safety objective; and to tend to fast paths (low consumption of resources and minimization of distance to target point) in the terms of the functions related to the speed objective. In the second set, the weights take values in order to give as high a priority as possible to the terms directly related to the objectives in each formula. This means that in the heuristic function for safety objective, there is a weight equal to 0 for minimization of distance to target point term; and in the heuristic and evaluation function for speed objective, there are weights equal to 0 for the terms concerning to unit visibility, for instance.

The user will decide the value for λ parameter, which gives relative importance to one objective over the other, so if it is near 1, finding the fastest path would be more important and if it is near 0, the other way round.

As we previously said, we have performed new experiments (relating to those made in our previous work), using each extreme value for λ, 0 and 1, and the set of weights which assigns the highest priority to each objective, in order to obtain extreme results. So the algorithm yields solutions considering only one objective. In addition, there is a comparison with mono-objective results, which have been obtained using almost extreme values for λ, 0.1 and 0.9, and the other set of weights (as well as hCHAC results), in order to consider both objectives (one with higher priority, of course).

hCHAC yields a set of non-dominated solutions (it is a MO algorithm), but less as usual in common MO algorithms, because it only searches in the region of the ideal Pareto front determined by the λ parameter. In addition, we only consider one (chosen by military participation considering their own criteria and the features of every problem).

We have carried out 30 runs for every scenario, using each state transition rule and using each value for λ (0, 0.1, 0.9 and 1) in order to find the fastest or the safest path in each case. All the algorithms use the same number of iterations (1500) and ants (50).

3.1 Map One (single enemy unit)

First scenario presents only one enemy located in the middle of the straightforward path between origin and destination points. There are some patches of forest and 3 different rivers (and bridges). In addition, there are some hills in the map. Figure 2 shows this map along with the best solutions found. Cells marked with black and white border compose the solution; those with black border are seen by enemy and those with white border are hidden. We have labelled in white: origin and destination cells with 'O' and 'D' respectively, and enemy unit with 'E'. Looking at Table 1, it is possible to notice that, in general, results obtained by hCHAC using CSTR are better than those yielded using DSTR because this last method has associated a higher exploration factor, so it needs to increase the explotation by running more iterations, for instance. Even so, the solutions are close to the CSTR ones. Both results are rather different (and worse) from the extremes values, more in the comparison between DSTR in normal hCHAC and in extreme configuration. This is a logical matter, because in hCHAC executions both objectives are always considered (with different priority level, of course), and in extreme approaches only the main objective is taken into account.

In all cases, differences between the safety cost F_s) when it is the primary and secondary objective are enormous. The reason is that fast paths are usually unsafe due to the visibility of the cells, and the visibility term is too much penalized in the cost function because a detected unit will be attacked immediately after.

Looking at the visual solutions in the map (Fig. 2), in the fastest paths, the unit goes in a rather straight way to the destination point, more straight in extreme (fastest) solutions, and less straight in DSTR method and mono-objective algorithm. All the paths have visible (from enemy) cells because they move near the enemy, but they also have hidden ones because they go through forest or behind hills, the reason being that safety is important even when searching for fast paths. This does not occur in the fastest path. The safest path found with CSTR and DSTR (in both configurations) represents a curve (distance to target point has little importance) which increases speed cost, but the unit goes through all hidden cells. Each solution moves by a zone of the map, but they two surround the enemy acquisition capability area and go through some forest cells or through mountains to avoid to be seen (safe cells). This behaviour is excellent from military tactical point of view.

The mono-objective solution tends to be on the safe side, yielding a path similar to those obtained by hCHAC with priority in searching for safe paths, but it is a little more direct in some sections. The reason is that security is more important than speed, so the weights of the aggregative function are set to ensure this.

3.2 Map Two (no enemy unit)

This scenario represents a rough terrain. It is located in a hill-covered zone and shows the origin and destination points in the top of two mountains; the destination hill is higher and rather craggy. Since there is no enemy, there are no absolutely visible or hidden cells, which means that all of them are visible in some measure (the algorithm calculates the visibility in a radius surrounding the unit in each step of the path, depending on the type and height of the cells inside that radio). Figure 3 shows this map along with the

Table 1: Results for Map One. (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
		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		

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

		Fastest (λ=0.9)		Safest (λ=0.1)	
		F_f	F_s	F_f	F_s
hCHAC-CSTR	Best	75.15	348.58	80.59	335.28
	Mean	77.26 ±1.58	352.03 ±5.68	80.53 ±1.98	344.96 ±4.51
hCHAC-DSTR	Best	79.36	352.77	82.74	350.21
	Mean	86.26 ±3.52	371.02 ±10.37	86.70 ±3.50	368.81 ±9.71
		Extreme Fast (λ=1)		Extreme Safe (λ=0)	
		F_f	F_s	F_f	F_s
extr-hCHAC-CSTR	Best	**60.03**	301.60	63.03	**271.51**
	Mean	**64.40 ±2.13**	305.21 ±8.90	63.89 ±1.52	**283.37 ±4.26**
extr-hCHAC-DSTR	Best	61.03	289.35	65.53	285.04
	Mean	66.61 ±3.05	312.58 ±12.76	68.61 ±2.75	300.90 ±9.33
mono-hCHAC	Best	73.78	281.03		
	Mean	75.32 ±1.59	286.21 ±4.46		

best solutions found by every approach. Cells marked with black border compose the solution, because there are no absolutely seen or hidden cells (there is no enemy watching over); the labels are the same as before.

The previous explained matter (about visibility when there are no enemy), is the reason why the results in the security cost (F_s) are high in all cases (see Table 2). In addition, all the solutions have similar costs in both objectives because a straight path to destination point is a good result even when searching for safe routes (it moves through mountain cells which are often hidden). The extreme solutions are again better, but the differences are less important than in the previous scenario.

Visually (in Fig. 3), all solutions are similar (straightforward paths from origin to destination point), and avoid to climb the higher mountain when searching for fast paths by surrounding its more difficult side (which corresponds to a natural obstacle for the unit), and sometimes, scaling the less difficult side in the safe paths (where it is feasible) to get a higher hidden level from surrounding cells. Again, all of this actions correspond to a sensible military behaviour.

4. CONCLUSIONS

In this work we have implemented, tested and compared 5 different ACS approaches to solve the bi-criteria military pathfinding problem (find the best path considering speed and safety as objectives). There are a multi-objective and a mono-objective algorithm and some approaches based on the configuration os parameters and weights (including extreme searches, which only consider one of the objectives). In addition, some of them use two different searching methods (CSTR and DSTR).

As result of the comparison, we consider that *hCHAC-CSTR* is the best approach because it maintains a very good balance between speed and safety in all cases, by considering always both objectives (with different priorities, depending on the search). This is a better method than extreme ones; even more so in difficult maps, where the cost for the objetive is not being minimized can be increased dramatically. *mono-objective* algorithm yield good solutions too, generally better than hCHAC-DSTR approach, but worse than those obtained by hCHAC-CSTR.

5. REFERENCES

[1] B. Barán and M. Schaerer. 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, pages 97–102, 2003.

[2] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[3] M. Dorigo and G. D. Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.

[4] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In G. K. F. Glover, editor, *Handbook of Metaheuristics*, pages 251–285. Kluwer, 2002.

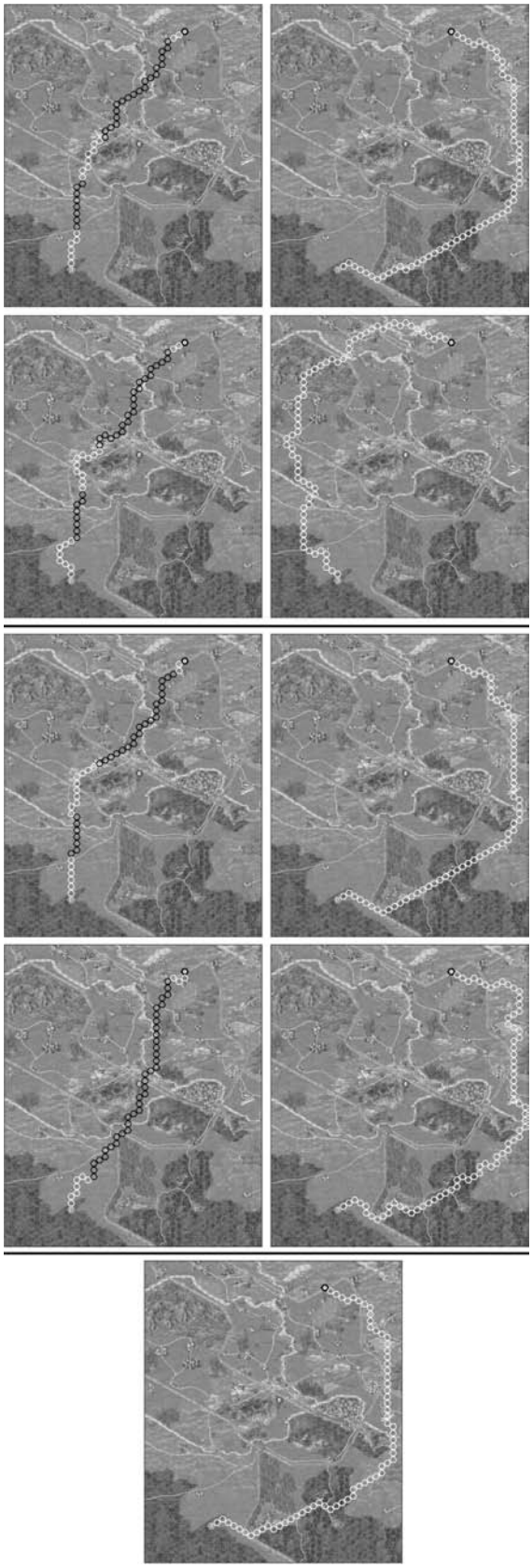

Figure 2: Map One (single enemy unit). From top to down: hCHAC-CSTR, hCHAC-DSTR, extreme hCHAC-CSTR, extreme hCHAC-DSTR, and mono-hCHAC results. Fastest (left) and safest (right).

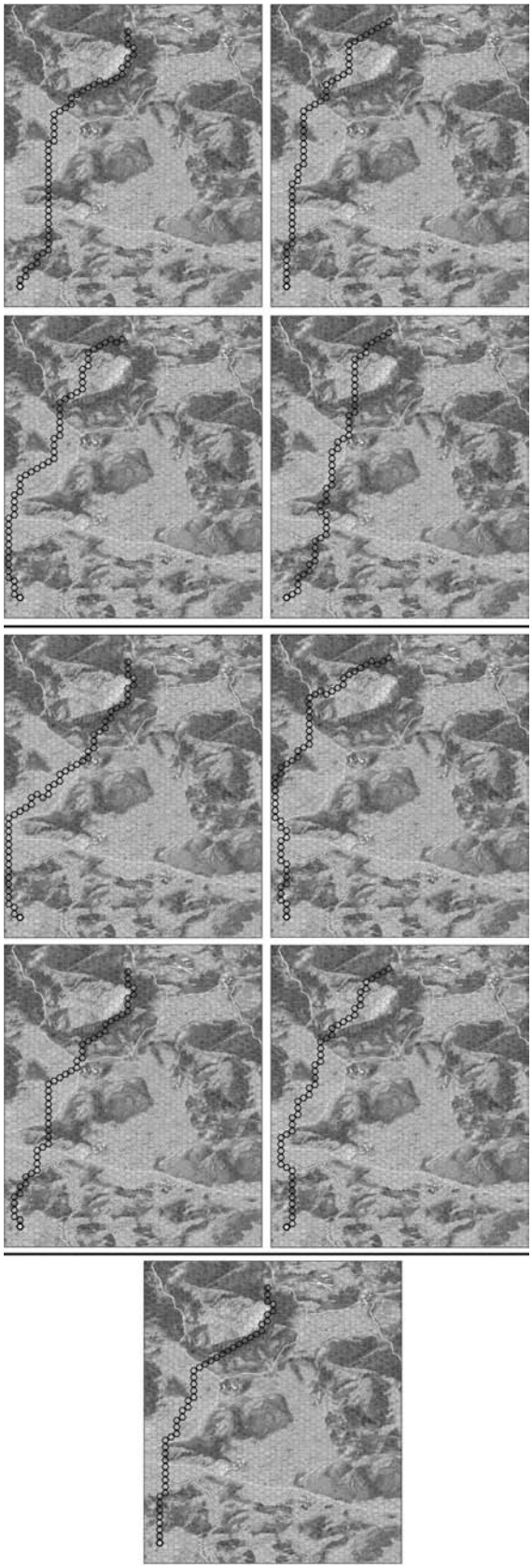

Figure 3: Map Two (No enemy unit). From top to down: hCHAC-CSTR, hCHAC-DSTR, extreme hCHAC-CSTR, extreme hCHAC-DSTR, and mono-hCHAC results. Fastest (left) and safest (right).

[5] C. García-Martínez, O. Cordón, and F.Herrera. An empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. In *ANTS 2004. Fourth International Workshop on. Ant Colony Optimization and Swarm Intelligence*, number 3172 in LNCS, pages 61–72. Springer, 2004.

[6] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, volume 1993 of *Lecture Notes in Computer Science*, pages 359–372, Berlin, 2001. Springer-Verlag.

[7] A. Mora, J. Merelo, C. Millán, J. Torrecillas, and J. Laredo. Chac. a moaco algorithm for computation of bi-criteria military unit path in the battlefield. In N. K. David A. Pelta, editor, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization. NICSO'2006*, pages 85–98, Junio 2006.

[8] A. M. Mora, J. J. Merelo, C. Millán, J. Torrecillas, J. L. J. Laredo, and P. A. Castillo. Enhancing a MOACO for solving the bi-criteria pathfinding problem for a military unit in a realistic battlefield. In M. Giacobini, editor, *EvoWorkshops 2007. Applications of Evolutionary Computing*, number 4448 in LNCS, pages 712–721. Springer, 2007.