The Ant Search Algorithm: An Ant Colony Optimization Algorithm for the Optimal Searcher Path Problem with Visibility

Michael Morin[1], Luc Lamontagne[2], Irène Abi-Zeid[3], and Patrick Maupin[4]

[1,2] Department of Computer Science and Software Engineering
[3] Department of Operations and Decision Systems
Université Laval, Québec, QC, Canada
Michael.Morin.3@ulaval.ca, Luc.Lamontagne@ift.ulaval.ca,
Irene.Abi-Zeid@osd.ulaval.ca
[4] Defence Research and Development Canada
Valcartier, QC, Canada
Patrick.Maupin@drdc-rddc.gc.ca

Abstract. In the first part of this paper, we present the Optimal Searcher Path problem with Visibility, a novel path planning approach that models inter-region visibility and that uses concepts from search theory to model uncertainty on the goal's (*i.e.*, the search object) detectability and location. In the second part, we introduce the Ant Search algorithm, a solving technique based on ant colony optimization. Our results, when compared with a general mixed-integer programming model and solver, show that Ant Search is a promising technique for handling this particular complex problem.

Keywords: Ant search, optimal searcher path problem, path planning, search theory, ant colony optimization.

1 Introduction

The *optimal searcher path* (OSP) problem is a well-known detection search problem in classical search theory. In short, an OSP problem consists of computing a sequence of regions that maximizes the probability of finding an object of unknown location using available resources and limited capabilities. Search theory and the OSP problem provide a framework to take into account uncertainty on the search object's (*i.e.*, the goal) detectability and location in path planning problems dealing with a physical travelling process: a robot trying to detect an object, a manned patrol searching for the survivors of an aircraft accident, etc. Recently, the *optimal searcher path* problem *with visibility* (OSPV) has been proposed as a variant of the classical OSP problem [12]. The OSPV formulation introduces the notion of inter-region visibility to take into account the fact that a search unit (*i.e.*, a searcher) may search (or scan) visible regions from a distant location.

In this paper, we extend the detection model used in the original OSPV problem formulation [12]. We also describe *ant search* (AS), an application of *ant colony* *optimization* (ACO) to solve the extended problem and to manage its complexity. We then present the results and compare them to a generic problem solving scheme: a *mixed-integer linear programming* (MILP) model and a general purpose solver. The purpose of the experiment is not to prove the superiority of ACO in comparison to MIP techniques, but to give insights on the AS algorithm performance and applicability to the OSPV problem.

2 The OSP Problem – Background and History

The OSP problem was initially defined in the search theory literature, one of the earliest Operations Research discipline in the United States. B.O. Koopman and the U.S. Navy's operations research group were the first to formalize the detection search problem. Their objective was to enhance the efficiency of naval operations during World War II [10]. Since then, search theory has been applied to surveillance, oil exploration, medicine and search and rescue operations. Recently, its application area has been extended to include unmanned aerial vehicles [7] and robotized search in structured environments [8] [11]. A definition of classical search theory can be found in [14] and [5] and a survey of the classical problems can be found in [1].

For one-sided search problems where the search object's motion model does not depend on the search unit's actions, we may identify two general problem classes in search theory: the *optimal search density* (OSD) problems and the *optimal searcher path* (OSP) problems. The former deals with an unconstrained search unit's motion while the latter, as described in the introduction, involves constrained search unit's motion. As shown in [15], the decision formulation of the OSP problem in discrete time and space with a stationary search object is NP-Complete.

Among the various OSP formulations, Stewart [13] considered a moving search object with a continuous search effort. The problem was solved using a network flow formulation under the assumption of an exponential *probability of detection* (*pod*) function. In the same paper, Stewart considered a discrete and unitary (indivisible) search effort and proposed a depth-first Branch and Bound algorithm. In the discrete (arbitrarily divisible) search effort case, Stewart suggested a sequential allocation or a relaxation of the indivisibility constraint for sufficiently large available search effort amounts. Eagle [4] proposed dynamic programming as a solving technique in the discrete and unitary search effort case. A review of Branch and Bound procedures and heuristics for OSPs before 1998 can be found in [16]. Among the recent developments related to the OSP problem, Lau [11] introduced the OSP problem with non-uniform travel times (OSPT) to use in structured environments.

3 A Formal Definition of the OSPV Problem

The OSPV problem may be characterized as a one-sided detection search problem in discrete time and space that involves one constrained search unit and one search object (moving or not). In the OSPV context, the environment where the search object is hidden is discretized by a set R of N regions numbered from 0 to $N - 1$. The time allocated to the search operation is discretized by a set I of T time steps numbered from 1 to T. For convenience purposes, all regions and time steps will be referred to by using their integer identification and $t = 0$ will represent the initialization step of the search operation.

As in OSP-like problems, the search unit's motion is constrained by the environment. For a given search unit, these inter-region accessibility constraints are defined by an *accessibility graph*. Each node of the graph corresponds to a region of the environment and the presence of an edge from region s to region r implies that it is possible for the search unit to travel from s to r within one time step. The *accessibility graph* is represented by an accessibility map A from the set of regions to the power set of regions (1). At each time step t, the search unit can move to an accessible region from its current position noted y_t (2) and y_0 is the initial search unit's position.

$$A : R \rightarrow 2^R . \tag{1}$$

$$\forall t \in I : y_t \in A(y_{t-1}) . \tag{2}$$

Inter-region visibility is defined as a *visibility graph* of regions where an edge from region s to region r implies that r is visible from s. Again, the graph is represented by a map (V) from the set of regions to the power set of regions (3). At each time step t, the search unit allocates a total amount of discrete search effort Q to one or many visible regions where $e_t(r)$ represents the amount of discrete search effort (between 0 and Q) allocated to region r at time step t (4). At a given time step t, the available search effort can only be allocated to regions that are visible from y_t (5) and the total amount of allocated effort must be less than or equal to Q (6). Moreover, the amount of effort allocated to any region r at time step 0 is equal to 0 (7).

$$V : R \rightarrow 2^R . \tag{3}$$

$$\forall t \in I : \forall r \in R : e_t(r) \in \{0,...,Q\} . \tag{4}$$

$$\forall t \in I : \forall r \in R : e_t(r) > 0 \Rightarrow r \in V(y_t) . \tag{5}$$

$$\forall t \in I : \sum_{r \in R} e_t(r) \leq Q . \tag{6}$$

$$\forall r \in R : e_0(r) = 0 . \tag{7}$$

The accessibility and visibility graphs share the same set of nodes but have different sets of edges. They can be seen as roadmaps of the original continuous environment (see [2] for roadmaps examples).

The *probability of containment* (*poc*) distribution over all the regions defines our knowledge of the search object's position before and during the search operation. The updated local *poc* value in each region and at each time step is conditional to the fact that the search object has not been detected before. If we assume that the search object is located somewhere in the environment, then the initial *poc* distribution (poc_0) will total 1.0 over the environment (8).

$$\sum_{r \in R} poc_0(r) = 1.0 \ . \tag{8}$$

At each time step, this distribution evolves according to the motion model of the search object and to the search unit's effort allocations (9).

$$\forall t \in I : \forall r \in R : poc_t(r) = \sum\nolimits_{s \in R} d(s,r)\left[poc_{t-1}(s) - pos_{t-1}(s)\right], \tag{9}$$

where $d(s,r)$ is the probability that the search object moves from region s to region r within one time step and where $pos_{t\text{-}1}(s)$ is the local probability of success in region s at the previous time step (defined in (11)). Recalling that $poc_t(r)$ is the probability that the search object is located in region r at time step t and given that it has not been detected before time step t, the resulting sum over all the regions s may be interpreted as the remaining "mass of containment" in region r at time step t. The motion model is such that d may be represented as a matrix where each row is a source region s and where each column is a destination region r. The sum of each row is 1.0 (10) since we assume that the search object cannot escape from the environment.

$$\forall s \in R : \sum\nolimits_{r \in R} d(s,r) = 1.0 . \tag{10}$$

The probability of success $pos_t(r)$ (local to a given region r at a given time step t) is conditional to the fact that the search object has not been detected earlier. This probability is the product of the local probability of containment in r at time step t and of the conditional local probability of detection in r at time step t (11). Note that $pos_0(r)$ is equal to 0 for all regions r since the search operation begins at time step 1.

$$\forall t \in I : \forall r \in R : pos_t(r) = poc_t(r) \times pod_t(y_t, r, e_t(r)) , \tag{11}$$

where $pod_t(y_t,r,e_t(r))$ is the probability of detection at time t in region r conditional to the fact that the search object is in region r at time t. Moreover, the *pod* function varies according to y_t due to inter-region visibility and as a function of $e_t(r)$, the amount of effort allocated in r at time step t.

In this paper, we assume that the detection model follows the exponential detection law of equation (12). The detectability index ($W_t(s,r)$) varies as a function of the source region s, of the destination region r and of time due to several factors (*e.g.*, weather).

$$\forall t \in I : \forall s, r \in R : \forall e \in \{0,...,Q\} : pod_t(s,r,e) = 1 - \exp(-W_t(s,r) \times e) . \tag{12}$$

Our objective is to obtain a search plan P (a path and a sequence of effort allocations) (13) that maximizes the cumulative overall probability of success (*COS*) defined as the sum of the local probabilities of success across regions and time steps (14).

$$P = \langle [y_1, y_2, ..., y_T] \quad [e_1, e_2, ..., e_T] \rangle . \tag{13}$$

$$COS(P) = \sum\nolimits_{t \in I} \sum\nolimits_{r \in R} pos_t(r) . \tag{14}$$

4 Searching with Ants

Ant colony optimization (ACO) is a general stochastic local search technique. As defined in [9], it constructs, at each iteration of the algorithm, a population of candidate solutions according to a common "memory" (the *pheromone trails*) to iteratively find better incumbent solutions. In view of the ACO metaheuristic metaphor, each candidate solution is built by an ant that stochastically chooses the solution's components based on their associated *pheromone value* and on a heuristic function that corresponds to the greedy "appeal" of this local decision. A typical definition for the probability of choosing a given action (or component) is shown in equation (15) where v_{act} is the pheromone value of the action *act* weighted by α , h_{act} is the heuristic value of the action *act* weighted by β and p_{act} is the resulting probability of choosing the action *act*.

$$p_{act} = \frac{(v_{act})^{\alpha}(h_{act})^{\beta}}{\sum (v_{act})^{\alpha}(h_{act})^{\beta}} . \tag{15}$$

Usually, the pheromone value of a given decision is updated proportionally to its quality and an evaporation factor decreases the pheromone values at each iteration in order to avoid getting stuck in a local optimum. In [3], Dorigo and Blum present a survey of ACO techniques.

4.1 The Ant Search Algorithm

Our ant search (AS) algorithm applies the idea of the ACO metaheuristic to the OSPV problem. This section defines the general algorithm and the *pheromone model* used to store and to update the trails. The outline of AS is presented on Figure 1.

```
Algorithm 1: AntSearch(Ospv, $C$, $p_{rdm}$ )
Input: An Ospv problem Ospv, the size of the colony
       $C$ and the probability of exploration $p_{rdm}$.
Output: The incumbent search plan $P_i$.
begin
    Initialize Trails with small values;
    while stopping criterion is not met do
        Colony ← Generate(Ospv,$C$,$p_{rdm}$,Trails);
        $P_i$,Trails ← Update(Colony,$P_i$,Trails);
    end
end
```

Fig. 1. The outline of the Ant Search algorithm

Given an OSPV problem, the total number of candidate search plans (solutions) to generate at each iteration, and a small probability of exploration, the algorithm first sets the pheromone trails to small values. Then it iteratively generates a set of candidate solutions and updates the trails and the incumbent according to its pheromone model (as defined in sections 4.1.1 and 4.1.2).

4.1.1 Generating the Colony

The *Generate* function constructs each solution of the colony by choosing the search unit's actions for each time step. There are two search unit's action types: moving from one region to another, and allocating one unit of effort in a visible region from its new position. Each ant can be seen as a search unit that tries to construct the best possible search plan using the colony's experience (defined as the pheromone trails). First, an ant chooses a region that is accessible from its current position. Then it sequentially allocates the Q units of available effort to visible regions. The process is repeated for each time step. In our implementation, the pheromone values correspond to the local probability of success (11) and to the overall probability of success at time t (16).

$$\forall t \in I : os_t = \sum_{r \in R} pos_t(r) . \tag{16}$$

The local probability of success is used to select the effort allocation and the overall probability of success is used to select the path (see section 4.1.2).

In classical ACO algorithms, the ant's choices are guided by the pheromone values and by a heuristic function. Our first implementation of AS involved a greedy heuristic. However, the time required for computing the value of a greedy move based on the overall probability of success was quite high: for each accessible region, each possible combination with repetitions of visible regions must be evaluated. Thus we chose not use a heuristic in the implemented pheromone model and we replaced the usual equation (15) with equation (17). The main benefit of such an approach is to reduce the time consumed by an ant during its stochastic choices. Another benefit to our choice is the reduction of the number of parameters since there is no need for α and β nor for any other heuristic parameters.

$$p_{act} = v_{act} \Big/ \sum v_{act} . \tag{17}$$

In order to avoid stagnation, at each time step, the ant has a small probability p_{rdm} of choosing a random move and a random search effort allocation. In the AS context, this random choice can help diversify the search since the ants rely entirely on the pheromone trails. At the end of the *Generate* function, a colony of C candidate solutions has evolved.

4.1.2 Updating the Trails and the Incumbent Solution

At each time step, each ant performs two action types: it moves and it allocates the search effort. The first challenge was to store the pheromone values in relatively small data structures while keeping enough information. The solution we retained was to create 2 $T \times N$ tables (one per action type) and to store the pheromone by pairs of time steps and regions. The second challenge was to choose the values to store in the tables. As mentioned in section 4.4.1, we used the overall probability of success for the *path table* and the local probability of success for the *allocation table*.

Two versions of the algorithm were tested. In the first variant (called *Egalitarian AS*), the pheromone is updated for all candidate solutions of the colony. In the second variant (called *Elitist AS*), the pheromone is updated only when the candidate solution is better than the incumbent (many updates may occur for the same colony of search plans depending on their evaluation order). In all cases, a small evaporation factor is applied at each iteration to decrease the pheromone values. The most promising approach is Elitist AS. As a result, we present the results of the Elitist AS algorithm only.

5 Experimentation

The goal of our experiment was to compare the results of Elitist AS over a small set of instances to the results of a well known and well-founded solution scheme that guarantees the optimality of its solutions (given that it has enough time and resources to do so): ILOG CPLEX and a MILP model. The experiment was run on the following hardware: an Intel Core2 Quad Q6600 processor with 3 GB of RAM. All the code was developed using C++. The following assumptions hold for each instance:

- All environments are grids containing cells of 5 × 5 distance units.
- The search object's motion model (*d*) is random on rows and each row sums up to 1.0.
- The initial *poc* distribution and the initial search unit's position are randomly generated (using a uniform distribution).
- The accessibility graph is defined using a maximum accessibility range a_{rng} equal to 5.01 distance units.
- The visibility graph is defined using a maximum visibility range v_{rng} equal to 8.0 distance units.
- A region *r* is accessible (resp. visible) from another region *s* if the distance between the centers of *s* and *r* is less than a_{rng} (resp. v_{rng}).
- The *pod* function corresponds to equation (12) but is kept constant over time and $W_t(s,r)$ is such that

$$\forall t \in I : \forall s, r \in R : W_t(s,r) = \begin{cases} \dfrac{(v_{rng} - dist(s,r))}{area(r)}, & \forall s, r \in R : r \in V(s) \\ 0, & \forall s, r \in R : r \notin V(s) \end{cases}, \qquad (18)$$

where *dist*(*s*,*r*) is the distance between the center of region *s* and the center of region *r* (in distance units) and *area*(*r*) is the area of region *r* (in square distance units).

Table 1 presents the 7 problems used for the tests. While these environments are relatively small, the overall complexity of the problem is high since a search plan is both a path of *T* regions and a sequence of *T* combinations with repetitions of *Q* visible regions. The evaluation metrics are the *COS* measure of the final incumbent solution and the approximate time (in seconds) needed to obtain the resulting search plan which corresponds to the last incumbent update time. By using these metrics we avoid including the time used by CPLEX to prove the optimality of its incumbent solution. The maximum allowed resolution time is 7200 seconds.

Table 1. The environments

No	Height	Width	*T*	*Q*
0	2	2	4	5
1	3	3	9	5
2	4	4	16	5
3	5	5	25	5
4	6	6	36	5
5	7	7	49	5
6	10	10	100	5

5.1 A Generic Solver

By looking at the formal OSPV model we note that the *pod* function, the initial *poc* distribution, the search object's motion model and the search unit's initial position are a priori known data. Moreover, the detection model implements a *pod* function that varies as a function of a discrete search effort. Thus the formal model can be reformulated as a MILP. The advantage of such a reformulation is that algorithms for solving MILP to global optimality are in the public domain (*e.g.*, [6]) and that MILP solvers implementing programmable libraries are available (*e.g.*, ILOG CPLEX 11.2 and Concert Technology 2.7). In the version used for the tests (*i.e.*, 11.2), CPLEX uses Branch and Cut algorithms variants such as dynamic search to solve MILP models. Branch and Cut is a cutting plane algorithm and its solving process is not too far from the usual Branch and Bound procedure used for OSP problems (*e.g.*, [13]).

We present the results of 5 different configurations of CPLEX. The first one (*Default*) involves the default parameters. The second (*ScaleUp*) involves a MILP model where all the probabilities are multiplied by a large factor (100000). The third (*Feasibility*) puts the emphasis on finding feasible solutions. The fourth (*BestEst*) implements a best-first search approach based on the estimate of the *COS* value given by the heuristics of CPLEX. The fifth (*DepthFirst*) implements a depth-first search approach.

5.2 Ant Search Configurations

In order to estimate the impact of the parameters on the performance of Elitist AS, the algorithm is evaluated on all the 7 problems using the following configurations: a colony size C in $\{1, 100, 250, 500, 1000, 2500, 5000\}$ with an exploration probability p_{rdm} of 0.1 and p_{rdm} in $\{0.0, 0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1.0\}$ with C equal to 100. Then, to derive the statistics, the algorithm is evaluated on the 7 problems for 10 runs with $C = 1000$ and $p_{rdm} = 0.001$. For all tests, the evaporation factor is set to 0.1 and the initial pheromone values are set to 0.01. Moreover, Elitist AS is configured to stop if no improvement occurs within 900 seconds of the last incumbent's update.

6 Results and Discussion

Figure 2 presents the last incumbent's *COS* as a function of various configurations for problems 4 to 6. Other problems (while having the same general tendency) are not shown on the figure to avoid overcrowding it. The first graph shows the *COS* as a function of increasing colony size (C) with an exploration probability (p_{rdm}) of 0.1, the second graph shows the *COS* as a function of an increasing exploration probability (p_{rdm}) with a colony size (C) of 100. For the evaluated OSPV instances and the evaluated configurations, the general tendency shows an increase in the solution's quality among larger colonies. We see (in the second graph) that a small exploration probability (p_{rdm}) has a positive impact on the incumbent's quality of problems 5 and 6 while the highest p_{rdm} values have a negative impact on the last incumbent's *COS*. When p_{rdm} tends toward 1.0, the algorithm roughly corresponds to a random search of the environment and this is why the performance usually decreases for the larger values of p_{rdm}. This tendency shows the positive impact of using our pheromone model instead of a random search as it guides the exploration of the solutions' space.

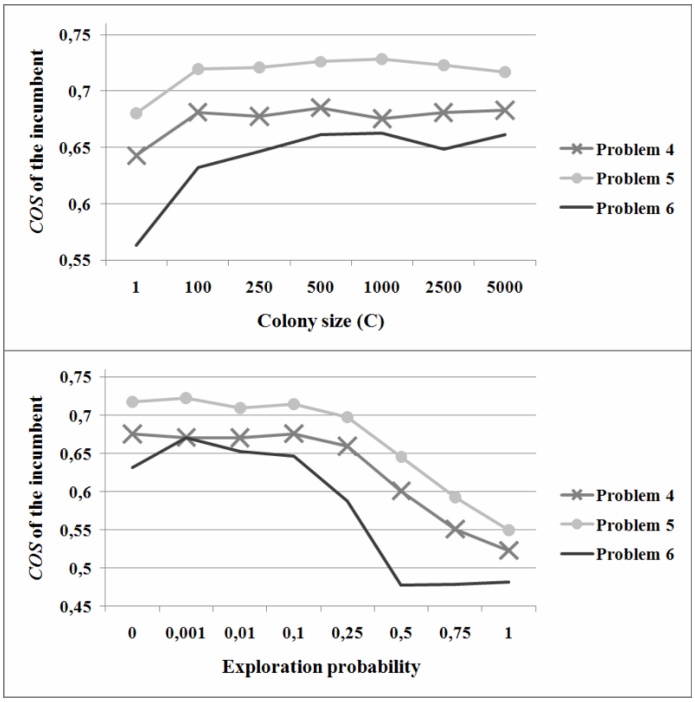

Fig. 2. For problems 4 to 6, the *COS* value of the incumbent for *C* in {1, 100, 250, 500, 1000, 2500, 5000} with $p_{rdm} = 0.1$ (above) and for p_{rdm} in {0.0, 0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1.0} with *C* = 100 (below)

Figure 3 presents the time (in seconds) spent to obtain the last incumbent's search plan as well as the corresponding *COS* value for problems 0 to 6. The time to last incumbent differs from the stopping time. This metric is preferred over the stopping time since it does not consider the time used by CPLEX to prove the optimality of its solution. The values displayed for Elitist AS are the average of 10 runs with *C* = 1000 and $p_{rdm} = 0.001$ and the values displayed for CPLEX are the best of the 5 tested configurations. A lower time value and a higher *COS* value imply a better performance. In all cases, CPLEX failed to prove the optimality of its solution within 2 hours. The MILP model of problem 6 is too large to fit in memory (considering the current hardware and software configuration of CPLEX). As a result CPLEX has failed to find a feasible solution. Moreover, the average *COS* of the last incumbent found by Elitist AS is equal or higher than the best *COS* of the last incumbent found by CPLEX for problems 0, 2, 4, 5 and 6. The exceptions are problem 1 and 3 where the average *COS* obtained by Elitist AS is slightly below the best one of CPLEX. Finally, we see that the average times to the last incumbent of Elitist AS are lower than the ones of

CPLEX. The exceptions are problem 4 where CPLEX found its first and last incumbent after approximately 550 seconds and problem 6 where CPLEX failed to start its solving process due to a lack of memory. The time to the last incumbent obtained by CPLEX on problems 4, 5 and 6 are irrelevant due to the poor solution's quality: the best *COS* values are approximately equal to 0.187, 0.053 and 0.0. These results are positive when we consider the fact that the Elitist AS algorithm was stopped 900 seconds after the last incumbent's update (if no improvement occurred). Introducing a mechanism to perturb the pheromone values after a specific delay of non improvement may yield to better results.

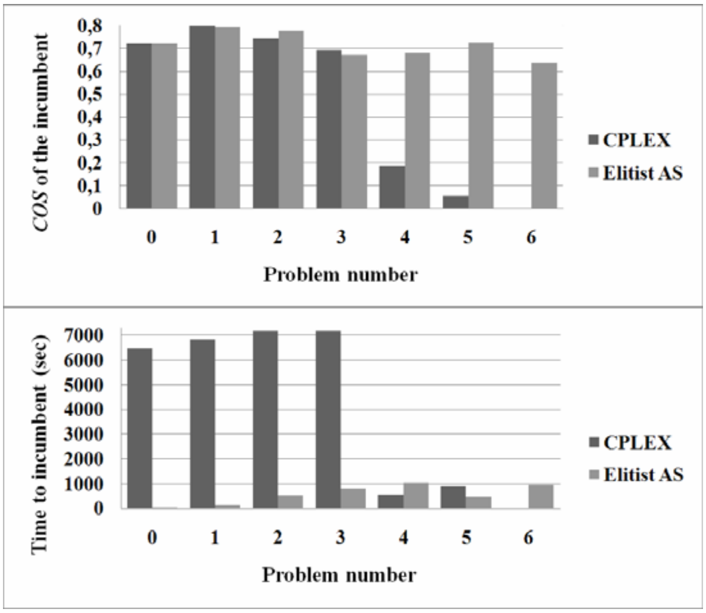

Fig. 3. The best *COS* obtained by CPLEX and the average *COS* obtained by Elitist AS with C = 1000 and p_{rdm} = 0.001 over 10 runs (above). The best time to the last incumbent obtained by CPLEX and the average time obtained by Elitist AS over the same 10 runs (below).

Figure 4 shows the distribution of the last incumbents' *COS* value obtained by Elitist AS for 10 runs on problems 0 to 6. For all problems, the *COS* values of the 10 runs are similar. There are few outliers considering an inter-quartile range (IQR) of 1.5. While providing *COS* values that are superior in average to the results obtained with our MILP model, Elitist AS is relatively constant in its incumbent quality (in view of the evaluated problems). Considering our 10 runs on problem 1 to 5, the *COS* values are within 0.005 of the mean in 95% of the cases. For problem 6, the confidence interval is of 0.03 with all the values and of 0.01 when removing the extreme *COS* value of 0.494. For problem 0, it is equal to 0.0. Considering the outlier obtained for problem 6 (*COS* = 0.494), further tests using the same random numbers with p_{rdm} = 0.1 suggest that this is a local optimum that can be avoided using a higher probability of exploration.

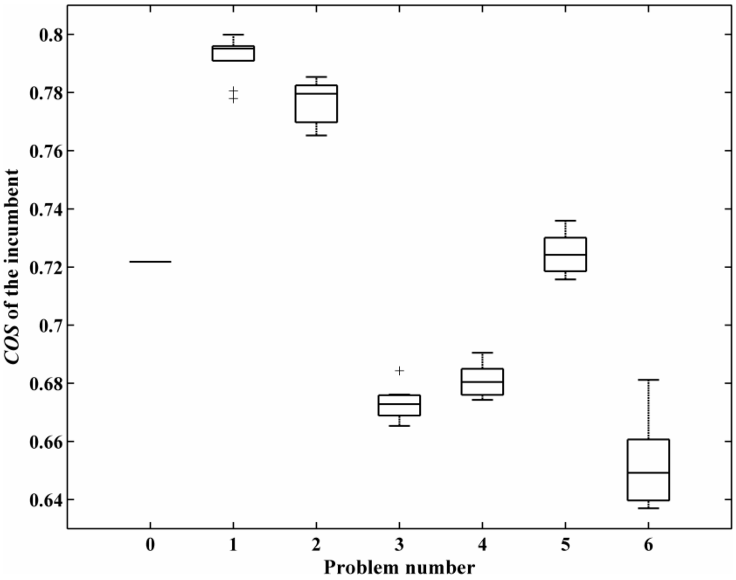

Fig. 4. For problem 0 to 6, a box plot (IQR = 1.5) illustrating the distribution of the last incumbent *COS* on 10 runs of Elitist AS. All the outliers are shown except for the value of approximately 0.494 obtained on problem 6 that was removed for clearness purpose.

7 Conclusion and Further Research

In this paper, we introduced the novel OSPV problem that models inter-region visibility and that uses concepts from search theory to model uncertainty on the search object's detectability and location. Moreover, we showed that our implementation of the ACO metaheuristic in the context of the OSPV problem (AS) is a promising technique to obtain search plans. Even if AS shares the main disadvantage of other local search techniques (it cannot guarantee the optimality of its last incumbent), it performed well in finding high quality solutions (in terms of *COS* when compared to our MILP formulation) to each tested instance. In all the experimental cases, the ratios of the last incumbent's *COS* to the time to the last incumbent for Elitist AS stayed high in comparison to the ones obtained by our general solving scheme involving CPLEX, a widely used MILP solver. Since practical OSP-like problems usually involve larger environments than those presented in our experiment (*e.g.*, a thousand of regions in the case of search and rescue operations), the AS methodology still needs to be enhanced. Further work and experiments include the development of fast heuristic functions, the development of better pheromone models to handle the environment's size, the reevaluation of Egalitarian AS and the development of algorithms based on other metaheuristics and on constrained programming.

References

1. Benkoski, S., Weisinger, J.R., Monticino, M.G.: A Survey of the Search Theory Literature. Naval Research Logistics 38, 469–494 (1991)
2. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry, Algorithms and Applications, 3rd edn. Springer, Berlin (2008)
3. Dorigo, M., Blum, C.: Ant Colony Optimization Theory: a Survey. Theoretical Computer Science 344, 243–278 (2005)
4. Eagle, J.N.: The Optimal Search for a Moving Target when the Search Path is Constrained. Operations Research 32(5), 1107–1115 (1984)
5. Frost, J.R.: Principles of Search Theory, part I-IV (2000)
6. Garfinkel, R.S., Nemhauser, G.L.: Integer Programming. John Wiley & Sons, New York (1972)
7. Hansen, S.R.: Applications of Search Theory to Coordinated Searching by Unmanned Aerial Vehicles. Master's thesis. Dept. of Mechanical Engineering, Brigham Young Univ., Provo, Utah, USA (2007)
8. Hollinger, G.A.: Search in the Physical World. Proposal for Ph.D. Thesis, Robotics Institute, Carnegie Mellon Univ., Pittsburgh, PA, USA (2008)
9. Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Elsevier, The Netherlands (2004)
10. Koopman, B.O.: Search and Screening: General Principles with Historical Applications. Pergamon Press, New York (1980)
11. Lau, H.: Optimal Search in Structured Environments. Ph.D. Thesis. The University of Technology, Sydney, Australia (2007)
12. Morin, M., Lamontagne, L., Abi-Zeid, I., Lang, P., Maupin, P.: The Optimal Searcher Path Problem with a Visibility Criterion in Discrete Time and Space. In: Proceedings of the 12th International Conference on Information Fusion, pp. 2217–2224. ISIF IEEE (2009)
13. Stewart, T.J.: Search for a Moving Target when the Searcher Motion Is Restricted. Computers and Operations Research 6, 129–140 (1979)
14. Stone, L.D.: Theory of Optimal Search. Topics in Operations Research, INFORMS (2004)
15. Trummel, K.E., Weisinger, J.R.: The Complexity of the Optimal Searcher Path Problem. Operations Research 34(2), 324–327 (1986)
16. Washburn, A.R.: Branch and Bound Methods for a Search Problem. Naval Research Logistics 45, 243–257 (1998)