Robotic Cops: The Intruder Problem*

Vincenzo Gervasi
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
gervasi@di.unipi.it

Giuseppe Prencipe
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
prencipe@di.unipi.it

Abstract – *In this paper we present a self-stabilizing algorithm for the* intruder problem. *The problem can be formulated as follows: an enemy unit, or* intruder, *is trying to sneak through a field patrolled by an arbitrary number of friendly autonomous (i.e. robotic) units. These units must reach the intruder and block it by surrounding it.*

Our solution to this problem, provided as an algorithm for the autonomous patrolling units, makes minimal assumptions on their capabilities. In particular, we assume they are completely asynchronous, and moreover that they have no observable identities, no memory, and no means to explicitly communicate with each other. Each unit needs only to be capable of observing the current position of its fellows and of the intruder.

All these features, while making the task harder, give to the algorithm the nice property of self-stabilization, thus improving its robustness. For example, if any unit is knocked out, all the others automatically adjust their behavior, in order to still complete the task. By concentrating on extremely simple units, we are also able to investigate which capabilities are really *needed to solve this problem, with obvious cost benefits (especially if the units are deployed in a hostile environment).*

In the paper, we first present a computational model for our robotic "cops", followed by the description of the algorithm we propose. We also show results of computer simulations, providing quantitative measures on the efficiency of the algorithm.

1 Introduction

Consider a restricted area patrolled by a set of robotic cops. Their task is to capture any hostile unit, or *intruder*, that could possibly enter the zone under surveillance: the cops consider their task done when they surround the intruder, so that it has no means to escape. This problem can arise in a number of real-world situations. For example, sensible areas where little or no traffic is expected, like airfields runways and aprons, or logistic compounds, could be effectively patrolled by robotic units, thus leaving humans free to concentrate on more high-level tasks.

Another possible setting is the automated patrolling of hostile territory, for example in military operations. In this case, the robots must be truly autonomous — the problem must be solved without relying on any kind of on-site infrastructure. Also, robots could conceivably be knocked off by opponents, and radio communications among them could be intercepted (thus revealing their presence) or disrupted (thus making them useless). Hence, a good solution to the intruder problem must do away with explicit communication, relying instead only on the intrinsic capabilities of the robots; it must assume no external help, and should be able to adapt to a varying (i.e., decreasing) number of robots.

This problem has been extensively explored in a graph-oriented setting [1, 5]: the robots have to patrol an area that is described as a graph; they can move only from node to node, following the edges connecting them. In the graph there is also an intruder robot, and the patrolling units must surround him: in particular, they have to occupy all the neighborhood of the node where the intruder is. In contrast to this kind of study, in our approach the patrolled area is modeled as a two dimensional plane where our agents, as well as the intruder, can freely move.

A related problem to ours has been analyzed in [11, 12], where the robots and the intruder could move strictly inside a polygonal area (including its border): each searcher robot could hold a flashlight that emits rays of light whose direction can be changed continuously. In their model, each cop can only see points lying on one of the rays. The goal of the robots is to detect the presence of the intruder. This is different from our problem, in that we assume the cops can always see the intruder, but we ask them to surround him rather than just detecting him.

Following the motivations that prompted previous studies ([4, 8, 9]), in this paper we adopt extremely simple units to study the problem: the cops are completely anonymous, identical (no identities are used during the computation), asynchronous, memoryless, and with no

*0-7803-7952-7/03/$17.00 © 2003 IEEE.

means of direct communication. We describe an algorithm, the same for all the cops, that allows them to surround the intruder, limiting his movement ability, and to keep him surrounded until some external event concludes the pursuit. Moreover, we present results of computer simulations that show the effectiveness of the proposed solution.

In Section 2 we introduce the computational model we adopt for our cops. In Section 3 we provide a formal definition of the intruder problem, and describe our algorithm to solve it. Section 4 presents a discussion on the evaluation of the algorithm, and provides the results of numerical simulations of the same. Some conclusions complete the paper.

2 The Computational Model

We consider[1] a system of autonomous mobile robots (*cops*) that have to patrol a given area, modeled as an infinite plane. A distinguished kind of agent, the *intruder*, is also on the plane. The intruder acts independently from the cops. The goal of the cops is to surround the intruder, while keeping at a certain distance from him, in order to reduce the leeway of the intruder. In particular, the cops must place themselves as to minimize the maximum distance that the intruder can place between himself and the nearest cop along any escape route.

Each cop is capable of observing its surrounding, computing a destination based on what it observed, and moving towards the computed destination; hence it performs an (endless) cycle of observing, computing, and moving.

Each cop has its own *local view* of the world. This view includes a local Cartesian coordinate system having an origin (that without losing generality we can assume to be the position of the robot), a unit of length, and the *directions* of two coordinate axes, together with their *orientations*, identified as the positive and negative sides of the axes. Notice that there is no agreement among the cops on the chirality of the respective coordinate systems (i.e., the robots do not share the same concept of where North, East, South, and West are).

The cops are modeled as units with computational capabilities, which are able to freely move in the plane. They are equipped with sensors that let each robot observe the positions of the others with respect to their local coordinate system. Each cop is viewed as a point, and can see all the other fellow cops in the patrolled area, as well as the intruder.

The cops act totally *independently* and *asynchronously* from each other, and do not rely on any centralized directives, nor on any common notion of time. Furthermore, they are *oblivious*, meaning that they do not remember any previous observation nor computations performed in the previous steps. Note that this feature, while making the capture task harder, gives to the algorithms designed in this model the nice property of self-stabilization [4]: in fact, every decision taken by a cop cannot depend on what happened in the system previously, and hence cannot be based on corrupted data stored in its local memory.

The cops are *anonymous*, meaning that they are a priori indistinguishable by their appearances, and they do not have any kind of identifiers that can be used during the computation. They can only distinguish the intruder from a fellow cop. Moreover, there are no explicit direct means of communication; hence the only way they have to acquire information from the fellow cops is by observing their positions. Note that the obliviousness of the cops also renders the observations weaker. In fact, nothing observed in the past can be remembered, hence used in order to let the cops organize themselves to accomplish their task.

They execute the same algorithm, which takes as input the observed positions, and returns a destination point towards which the executing cop moves. A cop, asynchronously and independently from the other robots, (i) *observes* the environment (*Look*), by taking a snapshot of the positions of all other cops and of the intruder with respect to its local coordinate system (since each cop is viewed as a point, its position in the plane is given by its coordinates); (ii) It *computes* a destination point p according to its oblivious algorithm (*Compute*); the local computation is based only on the current (i.e., at the time of the previous *Look*) locations observed by the robot. (iii) Finally, the cop *moves* an unpredictable amount of space towards p (*Move*), which is however assumed to be neither infinite, nor infinitesimally small (see Assumption A2 below), and goes back to the *Look* state.

The life of a cop consists in repeating an endless cycle of *states* (i)–(iii). Moreover, the only assumptions made in the model are the following:

(A1) The time for a robot to complete a *Look-Compute-Move* cycle is neither infinite nor infinitesimally small (i.e., is finite and bounded from below).

(A2) For each cop f, there exists an arbitrary (small) constant $\delta_f > 0$, representing the minimum distance it travels in the *Move* state; if the computed destination point is closer than δ_f, f will reach it.

(A3) Since we need to model robots that "continuously" move, we assume that the time spent in looking and computing is negligible compared to the time spent in moving.

Summarizing, each cop moves totally independently and asynchronously from the others, not having any

[1]The model we adopt is based on the CORDA model described in [10]. It has been adapted by taking into account the existence of a distinguished agent, the intruder.

bound on the time it needs to perform a *Move*, hence a cycle (it has to be, however, finite by Assumption A1); therefore, a robot can be seen *while* it is moving; in addition, they are oblivious, and anonymous. Moreover, no one of the cops knows in advance the path that the intruder will follow, nor can it derive it at run-time (e.g., by observing the position of the intruder at different times or his heading in order to estimate the current direction). Their only task is to observe where the intruder and the other cops are, reach an agreement — without communicating — on how to surround the intruder, and move to positions such that the intruder is kept in a confined area.

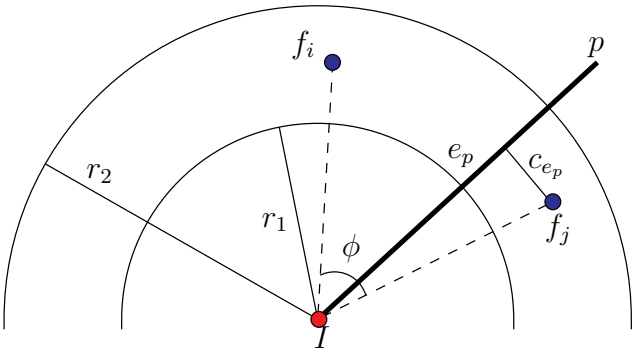


Figure 1: The geometric layout of the problem.

3 The Problem and the Algorithm

The intruder problem. The problem is formally defined as follows. Given $f_1, \ldots, f_n$ cops and the intruder I, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the two circles centered in I and having radius r_1 and r_2, respectively, where r_1 and r_2 are given constants of the problem, with $r_2 > r_1$. The cops must place themselves in the *capture area* $\mathcal{K}$, defined as follows:

$$\mathcal{K} = \mathcal{C}_2 \setminus \mathcal{C}_1.$$

In other words, the cops have to reach positions in the plane such that

$$\forall\ 1 \leq i \leq n,\ r_1 \leq \text{dist}(f_i, I) \leq r_2, \tag{1}$$

where $\text{dist}(a, b)$ denotes the Euclidean distance between a and b.

Moreover, the cops must evenly foreclose any escape route to the intruder. In formal terms, let p be a point on the plane such that $\text{dist}(I, p) > r_2$ (i.e., p lies outside of $\mathcal{C}_2$, and hence outside of $\mathcal{K}$). An *escape route* e_p for I to p is the segment $[I, p]$. The *capture distance* c_{e_p} for e_p is

$$c_{e_p} = \min_{1 \leq i \leq n} \text{dist}(e_p, f_i),$$

where we have extended dist() to segments in the usual way. Intuitively, the capture distance c_{e_p} is the minimum distance that the intruder would place between himself and the nearest cop, if he tried to escape $\mathcal{K}$ along e_p. Hence, the goal of the cops is to place themselves such that

$$\max_p c_{e_p} \text{ is minimal.} \tag{2}$$

It is easy to see that Condition (2) is satisfied when the cops place themselves spaced evenly on the inner border of the region allowed by Condition (1), thus forming a regular polygon of characteristic angle $\phi = 2\pi/n$ and radius r_1 (refer to Figure 1).

Limitations. Since the intruder keeps moving, it is impossible for the cops to maintain a perfect solution. In the following we will consider sub-optimal solution acceptable, as long as they are indefinitely maintained once first reached at time t_0. In this context a sub-optimal solution is defined as having, at each time $t > t_0$,

$$\forall\ 1 \leq i \leq n,\ r_1 - \epsilon_1 \leq \text{dist}(f_i, I) \leq r_2 + \epsilon_2,$$

and

$$\frac{\max_p c_{e_p}}{\tilde{c}} \leq 1 + \epsilon_3,$$

where $\tilde{c}$ is the minimal value from Condition (2) above.

The various constants $\epsilon_{1,2,3}$ are also tied to the temporal features of the asynchronous behavior of the cops. In fact, the longer the time between two consecutive *Look*s of a cop, the more outdated the snapshot taken of the other agents' positions becomes. Hence, computationally slow cops will only be able to guarantee a sub-optimal solution for relatively large values of $\epsilon_{1,2,3}$, while faster cops will be able to better approximate the optimal solution [2].

Finally, it is worthwhile to observe that the cops have no hope of reliably capturing an intruder faster than themselves. Therefore, a necessary condition for the solvability of the problem is that the intruder is slower than the slowest of the cops, i.e.

$$v_I < \min_i v_{f_i},$$

where v_k denotes the linear velocity of k.

The algorithm. As discussed above, the life cycle of each robot consists in an endless loop of *Look*, *Compute*, and *Move* phases. Since *Look* and *Move* are fixed, and do not depend on the particular problem that is being considered, the behavior of the cops is completely specified by defining an algorithm for the *Compute* phase.

[2]Space constraints prevent us from providing here a complete formal treatment of this problem. See [6] for a full treatment of a related problem.

In our case, the algorithm has in input the positions of all the other cops at the time of the last *Look*, and the position of the intruder, expressed as set of points in the local coordinate system of the cop. In particular, $(I.x, I.y)$ will denote the coordinates of I, and *Me* the current position of the cop executing the algorithm, that is $(0, 0)$ in its local coordinate system.

The algorithm must return as output the point q towards which the robots should move, also expressed in the local coordinate system. This is denoted by `moveTo(`q`)` in the code below.

Algorithm 1 The Intruder Capture Algorithm

```
    Chief := Closest Cop to I;
    If I Am Chief Then
      If dist(Me, I) > r1 Then
        moveTo(I).
5:    Else
        moveTo(Me).
    Else
      r := max(r1, dist(I, Chief));
      sortByAngle(Cops, I, Chief);
10:   k :=myRank();
      φ = 2π/n;
      θ :=angle([I, Chief), MyX);
      α := k · φ + θ;
      r' := r · (1 + ε);
15:   target := (I.x + r' · cos(α),
                 I.y + r' · sin(α));
      C := Circle Centered in I,
           With Radius r';
      If [Me, target] ∩ C ≠ ∅ Then
20:     target :=NonIntersTarget(I, target, r');
    moveTo(target).
```

The idea of the algorithm is as follows. First, the closest cop to the intruder is located (call it *chief*). The chief simply moves towards the intruder, trying to maintain a distance r_1 from him (Lines 2–6). All the other cops aim to reach the vertices of the regular n-gon inscribed in the circle $\mathcal{C}_r$ of radius $r = \max(r_1, \text{dist}(I, \textit{Chief}))$ and centered in the observed intruder's position. Once they reach such vertices, and $r_1 \leq r \leq r_2$, the cops' task is achieved. In order to reach an agreement on which vertex is assigned to each cop, the cops are sorted by routine `sortByAngle()`: in particular, the chief is considered to be the first cop in the order; the other cops are sorted, in increasing order, according to the angle each of them forms with the intruder and the chief (Line 9). At this point, the targets (i.e. the positions they have to reach in order to complete the task) of the cops are computed (Line 15): these are the vertices of the regular polygon having characteristic angle $\phi = 2\pi/n$, with the first vertex being on the chief's position, and inscribed in the circle $\mathcal{C}$ centered in I and having radius $r' = r(1 + \epsilon)$ (Line 14). The target of the i-th cop in the ordering is the i-th vertex of the polygon (the rank of the executing cop is returned by routine `myRank()` in Line 10). Routine `angle()` in Line 12 returns the angle between the half-line $[I, \textit{Chief})$ and the x axis in the local coordinate system of the executing cop: this angle is used to rotate the polygon to be formed so that the first vertex coincides with the *Chief* (Line 13). The reason for the targets being computed with respect to $\mathcal{C}$ and not with respect to $\mathcal{C}_r$, is to reduce cases where another cop becomes chief, displacing the previous chief: in fact, such displacements would introduce some instability in the algorithm, slowing down convergence.

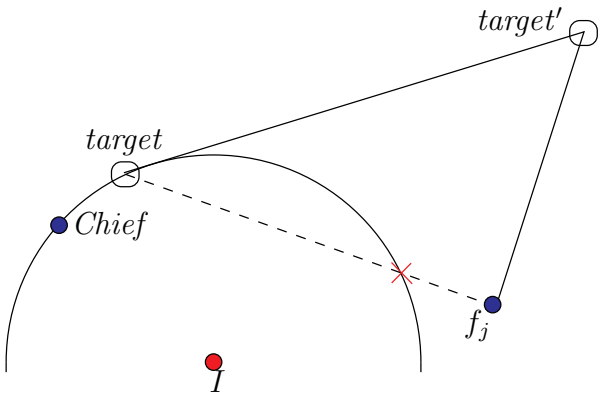


Figure 2: Sideway stepping in `NonIntersTarget()` routine.

Also, it is possible that a cop f, to reach its target, crosses $\mathcal{C}$. This too would introduce instability in the algorithm, since in so doing f could come closer to I than the current chief, thus becoming chief itself. To avoid this effect, Line 20 of the algorithm invokes routine `NonIntersTarget()`, that forces f to take a route outside $\mathcal{C}$, so that no crossing is possible: f moves sideways until a straight path from its current position to its assigned target does not cross $\mathcal{C}$ (see the example depicted in Figure 2). In this routine, the constant δ represents the length of the sideway step, and $(\textit{target}.x, \textit{target}.y)$ the coordinates of the *target* given in input. The cop will keep stepping sideway until necessary to reach its real target without crossing $\mathcal{C}$.

```
Routine NonIntersTarget(I, target, r')
  β := arctan(target.y/target.x);
  γ :=angle(Me, I, target);
  If γ > π Then
    β := beta + π/2;
  Else
    β := beta − π/2;
  target := (δ · cos(β), δ · sin(β));
  Return target.
```

As a final remark, note that a requirement of the intruder algorithm is that the cops must have common

knowledge [7] of the unit of measure. This is needed to allow them to have a common understanding of constants r_1 and r_2, and to agree on the distance they have to be to surround the intruder: in fact, the algorithm does not specify any strategy for deriving a shared unit of measure that can be maintained invariant at each cycle (remember that in our model the cops are completely oblivious).

4 Evaluation of the Algorithm

Experimental setting. To assess the effectiveness of the algorithm, we ran a number of tests using numerical simulations. Each run included a random[3] number of cops between 2 and 50; the intruder and the cops were initially placed at random in a 256×256 units square. The cops had their axes orientation and direction assigned randomly, and linear speed v_f between 0.5 and 5 space units per time units.

The intruder's course was determined as follows: at all times, the intruder would move forward according to its linear velocity. At each move, with a probability of 1/10, the intruder could start turning to its left or right, with random angular velocity less than its maximum angular velocity. If already turning, with probability 1/100 the intruder could stop and continue its course as a straight line (these parameters ensured curved, irregular trajectories).

Measures. To measure the convergence features of the algorithm, we measured three parameters. The first one, ν_r, measures how many cops have reached the capture area, as a ratio of the total number of cops:

$$\nu_r = \frac{|\{f_i | r_1 \leq \mathrm{dist}(f_i, I) \leq r_2\}|}{n}.$$

The second one, ϕ_a, measures the largest angle between two angularly adjacent cops in the capture area, i.e.

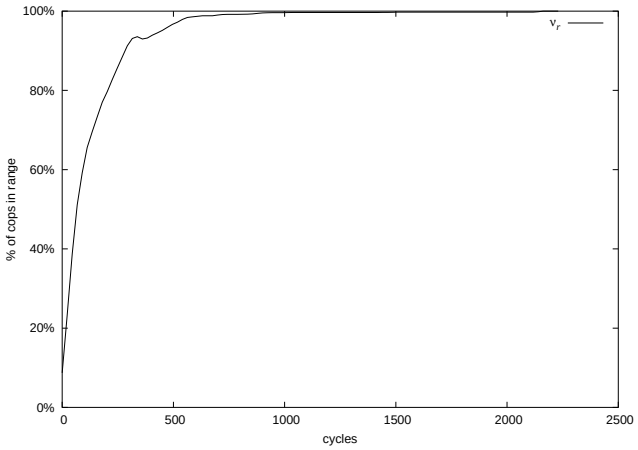
$$\phi_a = \max_{i,j}\{f_i \widehat{I} f_j\},$$

with $i \neq j$, such that there is no f_k, $i \neq k \neq j$, in the region of the plane delimited by the half-lines $[I, f_i)$ and $[I, f_j)$ intersected with the capture area. Of course, a given value of ϕ_a can be considered "good" or "bad", depending on the number of available cops n. In fact, the optimum ϕ_a for n cops is $2\pi/n$. To express this relative degree of optimality, we introduce a third measure ϕ_r, as a ratio between ϕ_a and its optimal value, that is

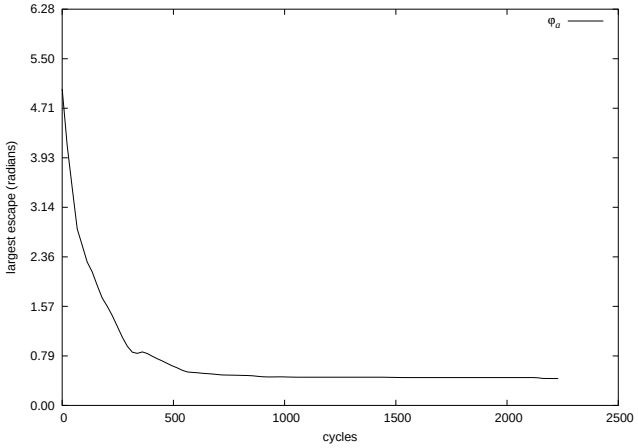
$$\phi_r = \frac{n \cdot \phi_a}{2\pi}.$$

Values of ϕ_r close to 1 indicate that the cops are close to the optimal capture configuration.

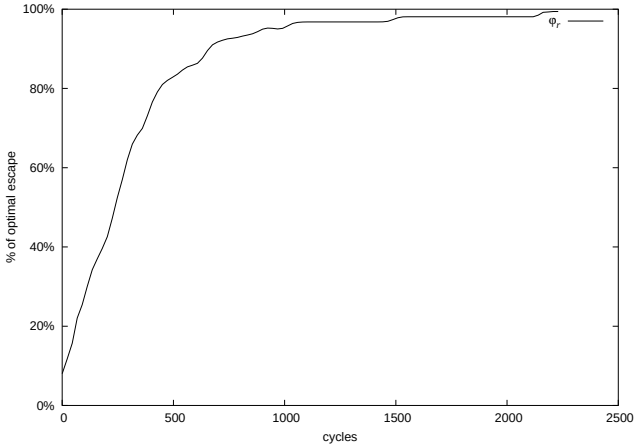
[3]In all cases, random values were obtained from a linear distribution.



(a)



(b)



(c)

Figure 3: (a) Average number of cops in the capture area (ν_r) in the simulations. (b) Average width of the largest angle between two angularly adjacent cops in the capture area (ϕ_a) in the simulations. (c) Average relative largest angle between two angularly adjacent cops in the capture area (ϕ_r) in the simulations.

Results. In all cases the cops were able to surround the intruder, correctly solving the problem (although with sub-optimal solutions, as described earlier). The results obtained by averaging the measures above over fifty random runs of our algorithm, with each run comprising 2300 *Look–Compute–Move* cycles, are shown in Figures 3.(a)–(c).

As can be observed in the figures, the algorithm exhibits reasonably fast and stable convergence to a good solution.

5 Conclusions

In this paper we studied the intruder problem: a number of robotic cops that patrol a restricted area have to capture an intruder that sneaked inside the area. The cops are non-communicating, asynchronous, anonymous and memoryless vehicles that can freely moves on a plane; the intruder is an external agent whose behavior is not known to the cops in advance.

We have provided an algorithm to solve the intruder problem, that only assumes that the cops share a common unit of distance, but need not to have a common sense of direction (i.e., a common coordinate system).

Indeed, the algorithm we proposed exhibits remarkable robustness, and numeric simulations indicate that the intruder is efficiently captured in a relatively short time and kept surrounded after that, as desired. The solution we proposed is self-stabilizing [2, 3]. In particular, any external intervention (e.g., if one or more of the cops are stopped, slowed down, knocked out, or simply faulty) does not prevent the completion of the task.

Real-world applications can usually count on a richer equipment: more powerful sensors to provide more information about the environment, on-board memory to store past observations and plans for the future, more sophisticated actuators, communication devices. We have proved that certain tasks can be accomplished without such a rich equipment. From a theoretical point of view, this clarifies the relationship between computability and solvability, and establishes some fundamental limit to what can be achieved. From a practical point of view, this allows simpler, more robust and economically advantageous units to be used instead of costly, complex and less fault-tolerant units for these tasks. Furthermore, their very simplicity ensures that the cops can be deployed also in hostile territories, since they do not rely on any infrastructure.

References

[1] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an Intruder by Mobile Agents. In 14^{th} *ACM Symposium on Parallel Algorithms and Architectures (SPAA 2002)*, 2002.

[2] E. W. Dijkstra. Self-stabilizing Systems in Spite of Distributed Control. *Communication of the ACM*, 17(11):643–644, November 1974.

[3] S. Dolev. *Self-Stabilization*. The MIT Press, 2000.

[4] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Distributed Coordination of a Set of Autonomous Mobile Robots. In *IEEE Intelligent Veichle Symposium (IV 2000)*, pages 480–485, 2000.

[5] N. Foukia, J. G. Hulaas, and J. Harms. Intrusion Detection with Mobile Agents. In 11^{th} *Annual Conference of the Internet Society (INET 2001)*, 2001. http://www.isoc.org.

[6] V. Gervasi and G. Prencipe. Coordination without Communication: The Case of the Flocking Problem. *Discrete Applied Mathemathics*, 2003. to appear.

[7] J. Halpern and Y. Moses. Knowledge and Common Knowledge in a Distributed Environment. In *Proceedings of the* 3^{rd} *ACM Symposium on Principles of Distributed Computing (PODC 1984)*, pages 50–61, 1984.

[8] Y. Oasa, I. Suzuki, and M. Yamashita. A Robust Distributed Convergence Algorithm for Autonomous Mobile Robots. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 287–292, October 1997.

[9] G. Prencipe. CORDA: Distributed Coordination of a Set of Autonomous Mobile Robots. In *Proceedings Fourth European Research Seminar on Advances in Distributed Systems (ERSADS 2001)*, pages 185–190, May 2001.

[10] G. Prencipe. Instantaneous Actions vs. Full Asynchronicity: Controlling and Coordinating a Set of Autonomous Mobile Robots. In *VII Italian Conference on Theoretical Computer Science (ICTCS 2001)*, pages 185–190, 2001.

[11] I. Suzuki and M. Yamashita. Searching for a Mobile Intruder in a Polygonal Region. *Siam Journal on Computing*, 21(5):868–888, 1992.

[12] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for Mobile Intruders in a Polygonal Region by a Group of Mobile Searchers. *Algorithmica*, 31:208–236, 2001.