A Decentralised Coordination Algorithm for Mobile Sensors

Ruben Stranders, Francesco Maria Delle Fave, Alex Rogers and **Nicholas R. Jennings**
{rs06r,fmdf08r,acr,nrj}@ecs.soton.ac.uk
School of Electronics and Computer Science, University of Southampton, UK

Abstract

We present an on-line decentralised algorithm for coordinating mobile sensors for a broad class of information gathering tasks. These sensors can be deployed in unknown and possibly hostile environments, where uncertainty and dynamism are endemic. Such environments are common in the areas of disaster response and military surveillance. Our coordination approach itself is based on work by Stranders et al. (2009), that uses the max-sum algorithm to coordinate mobile sensors for monitoring spatial phenomena. In particular, we generalise and extend their approach to any domain where measurements can be valued. Also, we introduce a clustering approach that allows sensors to negotiate over paths to the most relevant locations, as opposed to a set of fixed directions, which results in a significantly improved performance. We demonstrate our algorithm by applying it to two challenging and distinct information gathering tasks. In the first–pursuit-evasion (PE)–sensors need to capture a target whose movement might be unknown. In the second–patrolling (P)–sensors need to minimise loss from intrusions that occur within their environment. In doing so, we obtain the first decentralised coordination algorithms for these domains. Finally, in each domain, we empirically evaluate our approach in a simulated environment, and show that it outperforms two state of the art greedy algorithms by 30% (PE) and 44% (P), and an existing approach based on the Travelling Salesman Problem by 52% (PE) and 30% (P).

1 Introduction

Recently, information gathering with autonomous mobile sensors, such as unmanned aerial vehicles (UAVs) and ground vehicles (UGVs), has received considerable attention within the multi-agent system community. This is because gathering information in a fast and accurate fashion is of vital importance in many domains, such as military surveillance and disaster response. Now, while moving in an unknown, possibly hostile environment, the sensors measure specific features of interest. These include (but are not limited to) spatial phenomena (e.g. temperature or water contaminants) (Stranders et al. 2009; Krause et al. 2008), or targets moving within the environment (Vidal et al. 2001).

In this context, the key challenge faced by the sensors is the need to *coordinate* in order to maximise the amount of gathered information, since uncoordinated behaviour is likely to result in redundant sensor coverage of the environment and a waste of sensing resources. Also, given the dynamic and uncertain nature of the environments in which they can be deployed, it is necessary for the sensors to be able to adapt to unforeseen events. For instance, the detection of an evader or attacker requires the sensors to deviate from their previous plans and attempt to capture it. Consequently, sensors need to be able to revise their paths in an *online* fashion. Moreover, given the potentially hostile nature of their environments, it is possible that sensors fail. This, however, should not lead to the mission failure, or a severe performance degradation. Instead, the team should be *robust* to individual sensors failing. Consequently, coordination should be performed in a *decentralised* fashion, since the existence of a central controller would constitute a central point of failure.

Recent work has attempted to meet these requirements, but falls short of being fully adaptive and decentralised (Meliou et al. 2007; Singh, Krause, and Kaiser 2009) (who compute informative paths for mobile sensors in a centralised fashion), or is geared towards a specific domain (Stranders et al. 2009). The latter propose an adaptive decentralised coordination algorithm, that computes coordinated paths using the max-sum algorithm (Farinelli et al. 2008) in an online decentralised fashion. However, whilst being an improvement of earlier work in this area, this algorithm is specifically designed for monitoring spatial phenomena using Gaussian processes (GPs), and is therefore not of general use. Moreover, it performs coordination over a set of fixed directions, which we show leads to poor performance on less-structured graphs (i.e. non-lattice graphs).

Therefore, in this paper, we introduce a general model for a broad class of information gathering tasks by mobile sensors, and build upon and extend the algorithm proposed by Stranders et al. (2009) for this model. Furthermore, we significantly improve the performance of this approach by changing the set of paths over which the sensors coordinate. Specifically, instead of simply choosing the paths in a number of fixed directions, we adaptively cluster the sensors' neighbourhood and plot paths to the most relevant positions within each of these clusters. We ap-

ply our new algorithm to two widely studied problems for which no adaptive, decentralised algorithm previously existed. The first–pursuit-evasion–is characterised by the presence of an evader that the sensors need to capture as quickly as possible. Many theoretical results exist for the setting where the sensors and the evader move on a graph, showing that many realistic problem instances are NP-hard (Parsons 1976; Borie, Tovey, and Koenig 2009; Halvorson, Conitzer, and Parr 2009). Therefore, it is unsurprising that computing an optimal strategy (even for a single sensor) is intractable (Hespanha and Prandini 2002). Consequently, online adaptive, but approximate, algorithms have been developed for continuous bounded environments (Bopardikar, Bullo, and Hespanha 2008), and environments with an unknown layout defined by a graph (Hespanha, Kim, and Sastry 1999; Vidal et al. 2001). The latter work presents an attractive probabilistic approach to modelling the location of the evader, but uses two myopic greedy strategies, which, whilst being adaptive and robust, can also perform arbitrarily poorly because of the too limited look-ahead.

The second problem–patrolling–involves (possibly multiple) attackers attempting to intrude into the environment. Here, the main challenge is to detect and capture possible attackers so as to minimise loss from attacks. Again, several off-line and centralised optimal algorithms for this problem have been proposed, but they either do not scale beyond a single sensor (Paruchuri et al. 2007; Basilico, Gatti, and Amigoni 2009) or are restricted to specific types of graphs (Agmon, Kraus, and Kaminka 2008). On the other hand, approximate techniques have been proposed (Sak, Wainer, and Goldenstein 2008) using a fixed patrolling path through the environment (based on the Travelling Salesman Problem), which do not meet the requirement of adaptiveness and can therefore perform poorly in dynamic environments.

By showing that these two problems are instances of our general model, we are able to apply our algorithm to them and obtain the first online decentralised coordination approach for both. Moreover, we provide extensive empirical evidence that our algorithm outperforms the aforementioned state of the art approaches. In doing so, we make the following contributions to the state of the art:

- We extend and generalise the algorithm proposed by Stranders et al. (2009) to obtain an online decentralised coordination algorithm for mobile sensors for a broad class of information gathering tasks. Furthermore, we introduce an adaptive clustering technique for selecting the paths over which to coordinate that leads to significantly better results.
- We apply our algorithm to pursuit-evasion and patrolling, thereby obtaining the first online decentralised coordination algorithms for these domains.
- In both domains, we benchmark our algorithm against the state of the art: the two greedy algorithms proposed by Vidal et al. (2001) and a fixed path based on the TSP (Sak, Wainer, and Goldenstein 2008), and show that our algorithm outperforms these by 30% (PE) and 44% (P), and 52% (PE) and 30% (P) respectively. Furthermore, we show that our adaptive clustering technique results in a 42% performance increase compared to using a fixed set of paths as proposed by Stranders et al. (2009).

The remainder of the paper is organised as follows. We define the coordination problem in Section 2, and formalise a decentralised algorithm to solve it in Section 3. In Section 4 we define the two domains that we use to evaluate our algorithm in Section 5. We conclude in Section 6.

2 The Multi-Sensor Coordination Problem

In this section, we present a generic model for multi-sensor coordination, that subsumes many information gathering problems, including monitoring spatial phenomena (Stranders et al. 2009), patrolling and pursuit-evasion. These problems have many properties in common, such as the way the physical layout of the environment is modelled, how the sensors move, what they sense, and what the value of a measurement is in the context of the problem domain. We will now discuss each of these in turn, and introduce the notation used throughout this paper.

- The team of sensors is denoted as $\mathcal{S} = \{1, \dots, n\}$.
- The layout of the environment is given by a graph $G = (V, E)$, where V denotes the accessible locations embedded in $\mathbb{R}^2$, and E defines possible movements between locations in V.
- Time is discretised into time steps $T = \{1, 2, \dots\}$. At every step, sensors move to an adjacent location in G and take one or more measurements.[1]
- A measurement is a tuple $y = \langle y(v), y(t), y(m) \rangle$ where $y(v) \in V$ is the location and $y(t)$ the time at which y is taken, and $y(m)$ is a context dependent measurement.[2] The set of measurements taken at time t is denoted by y_t, and those taken up to and including t by Y_t. Measurements taken by sensor i are indicated by a superscript, for example y_t^i.
- The value of a set of observations y_{t+1} is given by a function $f(y_{t+1}, Y_t) \rightarrow \mathbb{R}$. The specification of this function depends on the application domain and the goals of the sensors within this application domain.[3]

Furthermore, we assume that localisation is accurate (i.e. sensors know which vertex in G they occupy), and that sensors know function f and graph G.

[1] Which measurements a sensor can make depends on the problem domain. For example, when measuring temperature, a sensor at v is only capable of measuring it at v. However, when scanning its environment for intruders, it might be able to observe locations other than v alone.

[2] For instance, in the spatial monitoring domain, m is a scalar (for example, representing temperature), while in the pursuit-evasion and patrolling domains m is a boolean (true iff an attacker was detected).

[3] For example, the coordination algorithm developed by (Stranders et al. 2009), which is aimed at predicting spatial phenomena, defines the value function $f(y_{t+1}|Y_t)$ as the entropy reduction that results from taking measurements y_{t+1}. This metric is strongly correlated with the accuracy (in terms of root mean squared error) of the predictions.

Given this problem definition, the key goal of the sensors is now to collect measurements that maximise f. In more detail, at each time step t, the sensors have to find:

$$[y^*_{t+1}, \ldots, y^*_{t+m}] = \underset{[y_{t+1}, \ldots, y_{t+m}]}{\arg\max} f(y_{t+1} \cup \ldots \cup y_{t+m}, Y_t) \quad (1)$$

subject to the movement constraints imposed by G. In other words, Equation 1 imposes an m step look-ahead,[4] and sensors need to compute paths of length m along which the most valuable measurements can be taken. At one extreme $m = 1$, sensors look ahead a single step and we obtain a greedy algorithm, which is likely to get stuck in local maxima and results in suboptimal long term behaviour. At $m = \infty$, their paths are optimal, but intractable to compute in any realistic setting. In light of this, in the next section, we present an algorithm for any value of m that computes high quality approximate solutions to Equation 1 by restricting the paths over which coordination takes place.

3 Decentralised Coordination

To develop a generalised decentralised coordination algorithm for computing solutions to Equation 1, we proceed in three steps. First, we present a decomposition of f that allows for the use of the max-sum algorithm. Second, we present a principled way of defining the action space for each sensor using a clustering approach. Third, we show how the max-sum algorithm can be applied.

To decompose f, we use the concept of the *incremental value* $\rho_y(y') = f(y \cup y') - f(y')$ of adding a set of measurements y to another set of measurements y'. Using this concept, we can decompose f into a sum of functions (as required by the max-sum algorithm):

$$f(y_{t+1}, Y_t) = \sum_{i=1}^{n} \rho_{y^i_{t+1}} \left(\bigcup_{j=1}^{i-1} y^j_{t+1} \right) \quad (2)$$

Thus, $f(y_{t+1}, Y_t)$ is a sum of the incremental values obtained by adding sensor i's future measurements y^i_t to those of sensors $j < i$. We will refer to the individual terms of the sum in Equation 2 as *sensor utility functions*, and denote these as U_i. Thus, in order to calculate its utility $U_i(y^1_t, \ldots, y^i_t)$, sensor i needs to take into account where sensors $j < i$ will take measurements, as well as its own.

Now, in our model there exists a one to one correspondence between a location v sensor i visits and the locations $y(v) \subseteq V$ at which it takes measurements. Thus, given the path $p_i = \langle v^{(1)}, \ldots, v^{(m)} \rangle$ that sensor i takes from its current position, we know $y^i_{t+1}, \ldots, y^i_{t+m}$. Thus, with slight abuse of notation, we can define the utility function U_i in terms of paths: $U_i(p_1, \ldots, p_i)$.

It is important to note that, in this formulation, sensor i's utility depends on the paths of *all* sensors $j \leq i$. However, there are two reasons for ignoring or discarding the dependency on some of these sensors, and removing the corresponding variables p_i as parameters from U_i:

1. In many domains, measurements made by certain sensors $j < i$ are independent[5] of i's and therefore do not influence sensor i's utility. Therefore, correctness is maintained by removing the corresponding variables from function U_i, which results in a much sparser coordination problem, which in turn results in less computation when using the max-sum algorithm.
2. In certain scenarios, a limited communication range might cause certain pairs of sensors to be unable to communicate directly with each other, hindering coordination between them. In such cases, the corresponding variables might have to be discarded from the sensors' utility functions, resulting in a less coordinated solution and (potentially) degraded team performance.

Note that, when the communication range is sufficient for any two sensors with dependent measurements to communicate, the performance of our approach is unaffected. For ease of exposition, in the remainder of the paper, we assume that communication between any pair of sensors is possible and that none of the sensors' measurements are independent.

Now, the key challenge is to define over which paths the sensors coordinate, or more precisely: the domain of variables p_i. The reason for this is that it is computationally infeasible to coordinate over *all* paths of length m, since this results in a combinatorial explosion for increasing m. This problem was recognised by Stranders et al. (2009), who define these domains as the paths that take the sensors in the 8 compass directions. However, while this might work well on a very structured graph, such as a lattice, we will show in Section 5 that this leads to a degradation of performance in less structured graphs, because not all locations of interest are considered (or reachable) in this fashion.

In this paper, we therefore propose a different approach to compute the domain of p_i based clustering the neighbouring area of sensor i, which proceeds in three steps:

1. Cluster the m-neighbourhood $N_G(v)$ (i.e. the subgraph of G induced by the vertices reachable in m steps from v) of the sensor's current location v into c clusters using the k-means algorithm.[6]
2. Within each cluster, identify the vertex for which f reaches a maximum.
3. The domain of p_i is now defined as the paths leading to each of these c vertices.

Clearly, the more clusters (the bigger c), the bigger the domain of p_i, and more paths need to be considered and more computation is necessary. However, by considering more (combinations of) paths the solution quality is also expected to improve. Thus, the parameter c can act as a trade-off between solution quality and computation.

[4] Since the problem over (even) a finite time horizon is NP-hard, we reduce the computational burden by limiting the look-ahead in this fashion.

[5] Two measurements y and y' are called independent if their values are additive: $f(y \cup y') = f(y) + f(y')$. As a result, $\rho_y(y') = \rho_y(\varnothing)$. Commonly, measurements taken far apart are independent, because the set of features that are observed are disjoint.

[6] This clustering algorithm was chosen for its simplicity and effectiveness. We intend to consider other algorithms for future work.

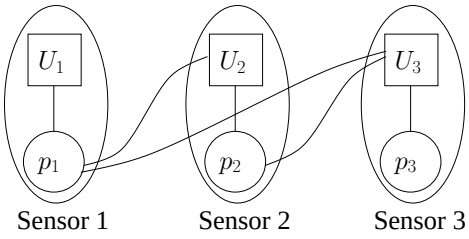


Figure 1: The factor graph for three sensors.

We have now transformed the problem so it is amenable to optimisation by the max-sum algorithm. The max-sum algorithm is a decentralised coordination algorithm based on message passing, and has been shown to compute good quality approximate solutions with acceptable computation overhead compared to various optimal decentralised algorithms (Farinelli et al. 2008). Moreover, it has been empirically shown to scale well with the number of sensors, and be robust against message loss and failure of individual sensors. In more detail, the max-sum algorithm requires the problem to be encoded as a factor graph, which is a bipartite graph in which vertices represent variables and functions, and an edges represent the 'is parameter of' relation. Hence, in our problem, an edge exists between vertices p_i and U_j iff $i \leq j$, such that sensor i's utility function U_i has i variables. Figure 1 shows the factor graph for a setting with three sensors.

Now, we can use the max-sum algorithm to find a path for each sensor $[p_1^*, \ldots, p_n^*]$ that maximises the value of the measurements collected by the team of sensors:

$$[p_1^*, \ldots, p_n^*] = \arg\max_{[p_1,\ldots,p_n]} \sum_{i=1}^{n} U_i(p_1, \ldots, p_i) \tag{3}$$

This is accomplished in a decentralised fashion by passing messages between the functions and variables as follows:

$q_{i\to j}$ **Message from variable p_i to function U_j ($i \leq j$):**

$$q_{i\to j}(p_i) = \alpha_{ij} + \sum_{k\geq i, k\neq j} r_{k\to i}(p_i) \tag{4}$$

where α_{ij} is a normalising constant chosen such that $\sum_{p_i} q_{i\to j}(p_i) = 0$, so as to prevents the messages from growing to the point of calculation error in cyclic factor graphs.

$r_{j\to i}$ **Message from function U_j to variable p_i ($i \leq j$):**

$$r_{j\to i}(p_i) = \max_{\mathbf{p}_j \setminus p_i} \left[U_j(\mathbf{p}_j) + \sum_{k\leq j, k\neq i} q_{k\to j}(p_k) \right] \tag{5}$$

Here we use $\mathbf{p}_j$ as an abbreviation for $\{p_1, \ldots, p_j\}$. Thus, $U_j(\mathbf{p}_j)$ is the utility function of sensor j.

These messages are exchanged until their contents converge, or for a predefined number of iterations if the former does not occur. Then, to obtain its part of the coordinated solution p_i^* defined in Equation 3, sensor i calculates:

$$p_i^* = \arg\max_{p_i} \sum_{k\geq i} r_{k\to i}(p_i) \tag{6}$$

The motivation here is that this sum is an approximation to the marginal function of Equation 3 with respect to p_i, where for any element e of the domain p_i, this marginal function is equal to the maximum value Equation 3 can attain if $p_i = e$. When the factor graph is acyclic, this marginal function is exact, and as a result, the solution computed in Equation 6 is guaranteed to be optimal. Otherwise, it is an approximation to Equation 3.

In Section 5 we apply our algorithm to two challenging information gathering domains, which are discussed next.

4 Two Information Gathering Domains

We now instantiate the general model and algorithm described in Sections 2 and 3 for two information gathering domains for which no online decentralised algorithm previously existed. In the first, pursuit-evasion, sensors are tasked with capturing a moving object in graph G. In the second, patrolling, the sensors' goal is to prevent intrusions on vertices of G. In both domains, observations y are binary: either a moving object or intrusion is detected, or it is not. Furthermore, we assume that sensors are capable of sensing all locations within a radius of r_s of their position. Sensing is assumed to be imperfect, and is given by a probability of a false positive p_{fp} and a false negative p_{fn} detection. In the upcoming sections, we will derive the measurement value function f we defined in Section 2 for both domains.

4.1 Pursuit Evasion

The pursuit evasion problem is characterised by the presence of a single moving object e (called an evader) that the sensors have to capture as quickly as possible. The evader e has a motion model M (e.g. random, stationary, etc.) that specifies how it moves in graph G. Figure 2 shows an example of this scenario with a randomly moving evader.

Now, to model their belief of the evader's location at time t, denoted by e_t, the sensors maintain a probabilistic map $p_e(e_t = v|Y_{t-1})$ representing the probability that the evader is at location v given measurement history Y_{t-1}. This model extends the work by Hespanha, Kim, and Sastry (1999) to a setting with an arbitrary evader (as opposed to a randomly moving one), or an unknown evader. Each sensor has a copy of this probabilistic map which is kept consistent by exchanging observations. At each time step t, the sensors take new measurements y_t and obtain $p_e(e_{t+1} = v|Y_t)$ in two steps:

1. Fuse measurements y_t with $p_e(e_t = v|Y_{t-1})$ to obtain:

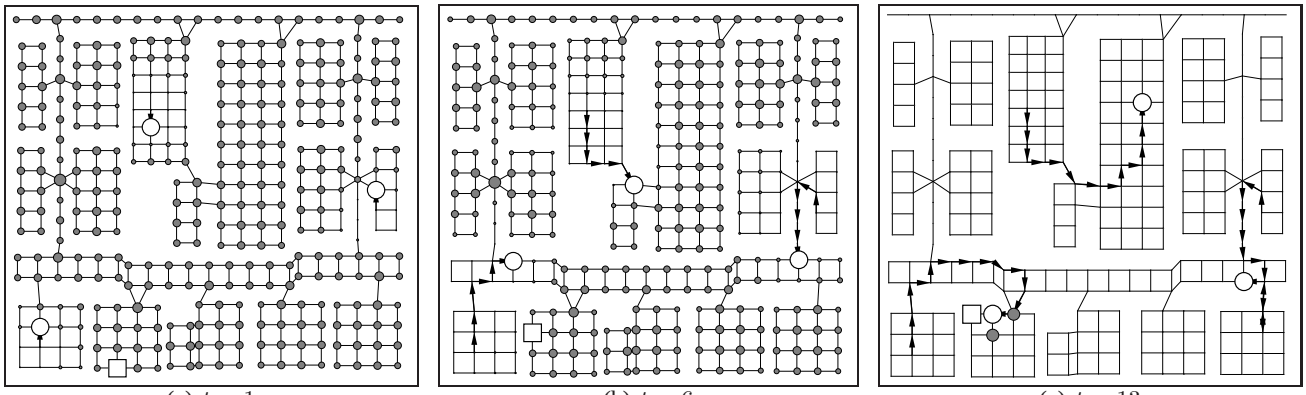
   $$p_e(e_t = v|Y_t) = \alpha p_e(e_t = v|Y_{t-1}) p(y_t|e_t = v, Y_{t-1})$$

   Here, α is a normalising constant, and $p(y_t|e_t = v, Y_{t-1})$ is computed as:

   $$p(y_t|e_t = v, Y_{t-1}) = \prod_{y\in y_t} p(y|e_t = v, Y_{t-1})$$

   and $p(y|e_t = v, Y_{t-1})$ is given by the sensing model:

   $$p(y|e_t = v, Y_{t-1}) = \begin{cases} 1 - p_{fp} & \text{if } y(m) = T \wedge y(v) = v \\ p_{fp} & \text{if } y(m) = T \wedge y(v) \neq v \\ p_{fn} & \text{if } y(m) = F \wedge y(v) = v \\ 1 - p_{fn} & \text{if } y(m) = F \wedge y(v) \neq v \end{cases}$$



(a) $t = 1$ (b) $t = 6$ (c) $t = 13$

Figure 2: An example pursuit-evasion scenario in the 'Office' environment. The big circles represent the sensors and the square represents the evader. The size of the grey circles is proportional to the value of f. The evader is captured at $t = 14$.

2. Predict the motion of the evader. If the the evader's motion model M is known, the sensors compute:

$$p_e(e_{t+1} = v|Y_t) = \sum_{v' \in V} p_m(e_{t+1} = v|e_t = v', M) p_e(e_t = v'|Y_t) \quad (7)$$

Here, $p_m(e_{t+1} = v|e_t = v', M)$ is the transition probability of evader type M from v to v'. Since the evader's movement is restricted by G, $p_m(e_{t+1} = v|e_t = v', M) = 0$ if $(v, v') \notin E$. If, however, the evader's motion model M is unknown, but it is known that $M \in \{M_1, \ldots, M_n\}$, the sensors compute a posterior over types M given observations y_t:

$$p(M_i|Y_t) = \alpha p(y_t|Y_{t-1}, M_i) P(M_i|Y_{t-1}) \quad (8)$$

where $p(y_t|Y_{t-1}, M_i) =$

$$\sum_{v \in V} p(y_t|e_t = v, Y_{t-1}, M_i) p_e(e_t = v|Y_{t-1}, M_i) \quad (9)$$

In this case, $p_e(e_{t+1} = v|Y_t)$ is computed as:

$$\sum_{i=1}^{n} p_e(e_{t+1} = v|Y_t, M_i)$$

Using the probability map obtained from this computation, measurement value function f is defined as:

$$f(y_{t+1}, Y_t) = \sum_{y \in y_{t+1}} p_e(e_{t+1} = y(v)|Y_t)$$

which is the probability of observing the evader in the next time step. To actually capture the evader, the sensors suspend their information gathering tasks as soon as a positive observation is made, and move to the location where the evader is expected to be in the next time step.

4.2 Patrolling

In the patrolling scenario, the sensors' goal is to prevent attacks on vertices of graph G. An attacker can start an attack on a vertex at any time (if no attacker is already present) and is successful if it is not captured by a sensor within k time steps. Each time an attack is successful, the sensors incur a loss of l. To model this domain, we build a Markov model for each vertex v with $k + 1$ states $S = \{\varnothing, 1, \ldots, k\}$, with $\varnothing$ indicating no attacker is present, and $1, \ldots, k$ representing that an attacker has been present for the indicated amount of time. Valid state transitions are:

- $\varnothing \to 1$: an attacker appears.
- $i \to (i + 1)$ for $1 \leq i \leq (k - 1)$: the attack progresses.
- $i \to \varnothing$ for $1 \leq i \leq k$: the attack is stopped by a sensor. No loss is incurred.
- $k \to \varnothing$: the attack succeeds and a loss of l is incurred.

Similar to the pursuit-evasion problem, the sensors maintain a probability map $p_a(s_t^v|Y_{t-1})$ representing the probability that a vertex v is in one of the states S at time t. To obtain $p_a(s_{t+1}^v|Y_t)$ from new measurements y_t the sensors follow a similar two step computation:

1. Fuse measurements y_t with $p_a(s_t^v|Y_{t-1})$ to obtain $p_a(s_t^v|Y_t)$. For ease of exposition, assume perfect sensing (i.e. $p_{fn} = 0$ and $p_{fp} = 0$). Under this assumption, we can distinguish two cases: v is currently under attack (denoted by a) or it is not (denoted by $\neg a$). In the former case, it is obvious that $p_a(s_{t+1}^v = \varnothing|a) = 0$. For $i \neq \varnothing$ we calculate $p_a(s_{t+1}^v = i|a) = \frac{p_a(s_t^v = i|Y_{t-1})}{1 - p_a(s_v^t = \varnothing|Y_{t-1})}$. In the latter case, we clearly have $p_a(s_{t+1}^v = \varnothing|\neg a) = 1$.

   Now, dropping the assumption of perfect sensing, note that in case of a negative observation an attacker might still be present with probability p_{fn}. Thus, to obtain $p_a(s_{t+1}^v|Y_t)$, we weigh vectors $p_a(s_{t+1}^v|a)$ by p_{fn} and $p_a(s_{t+1}^v|\neg a)$ by $1 - p_{fn}$. The converse holds for a positive observation.

2. Update the probability map taking into account possible new attacks. If we suppose that at every time step $t \in T$ and every $v \in V$ the probability p of an attack starting is constant and independent, this Markov model is fully

defined by the probabilistic transition function:

$$p_t(s^v_{t+1}|Y_t, s^v_t) = \begin{cases} 1-p & \text{if } s^v_{t+1} = \varnothing \text{ and } s^v_t = \varnothing \\ p & \text{if } s^v_{t+1} = 1 \text{ and } s^v_t = \varnothing \\ 1 & \text{if } s^v_{t+1} = i \text{ and } s^v_t = i-1 \\ & \qquad \text{for } i \leq k-1 \\ 1 & \text{if } s^v_{t+1} = \varnothing \text{ and } s^v_t = 1 \\ 0 & \text{otherwise} \end{cases}$$

Using this function we compute:

$$p_a(s^v_{t+1}|Y_t) = p_t(s^v_{t+1}|Y_t, s^v_t)p_a(s^v_t|Y_t)$$

After performing this computation, we can define function f for this domain in terms of the probability of an attack currently in progress at a measured location as follows:

$$f(y_{t+1}, Y_t) = \sum_{y \in y_{t+1}} \sum_{i=1}^{k} \gamma^{i-k} p_a(s^{y(v)}_{t+1}|Y_t)$$

where γ is used to discount the loss from attacks that occur in the future. Similar to the pursuit-evasion scenario, to capture an attacker, the closest sensor moves towards it.

5 Empirical Evaluation

To evaluate our algorithm, we applied it to the problem domains discussed above in two different graphs:

Office The layout of this environment is a model of our lab, which measures 67 by 47.5 metres. The sensing range $r_s = 9$m and capture range $r_s = 4$m. This environment is depicted in Figure 2. The number of sensors $n = 4$.

Lattice A 25 by 25 square lattice graph measuring 100 by 100 metres. Sensing range $r_s = 10$m, and capture range $r_s = 4$m. The number of sensors $n = 5$.

These two types of graphs were chosen to illustrate the effect of graph structure on the performance of our algorithm compared to the one proposed by Stranders et al. (2009). The setup for both problem domains is as follows:

Pursuit-Evasion The type of the evader is stationary, random (i.e. moves to a random adjacent location) or smart (i.e moves away from the closest sensor), with equal probability.

Patrolling Each instance lasts for 200 time steps. New attacks appear with $p = 0.0003$, and last 20 time steps.[7]

These values were chosen to create problem instances that distinguish between poor performing and well performing algorithms.[8] For all problem instances, the key probabilities characterising the sensing model are $p_{fp} = p_{fn} = 0.001$.[9] For pursuit-evasion we measured the time needed by the sensors to capture the evader; for patrolling, we measured the total loss incurred by successful attacks, and we benchmarked our algorithm against the state of the art, as well as algorithms that provide upper (JGreedy) and lower bounds (GRandom) on achievable performance:

(G)Greedy These are two state of the art algorithms for pursuit-evasion by Vidal et al. (2001) (which we will apply to both pursuit-evasion and patrolling). Greedy is an algorithm which moves to the adjacent location with the highest value in the next step, while GGreedy moves greedily towards the global location with highest value.

JGreedy This approach instantaneously jumps to the global location with the highest value.

(G)Random These are two random approaches. Random moves to a random location adjacent to the sensor's current position. GRandom selects a random position in the graph and then moves along the shortest path.

TSP This state of the art approach is proposed by Sak, Wainer, and Goldenstein (2008) and computes the shortest closed walk that visits all vertices (similar to the Travelling Salesman Problem[10]). To improve its adaptiveness and competitiveness, we let the sensors deviate from this path once an attacker is detected in order to capture it.

MS-8 This is our algorithm, but instead of using clustering to compute the paths over which the agents coordinate, it uses the 8 major directions on the compass rose, as suggested in Stranders et al. (2009).

MS K-M Our algorithm. For both environments, path length $m = 15$, and the number of clusters $c = 4$.[11]

We randomly generated 200 problem instances for both patrolling and pursuit-evasion, and evaluated the performance of all algorithms on each of these instances. The results are summarised in box-plots shown in Figures 3(a) and 3(b), showing that our algorithm has significantly smaller variance compared to GGreedy, MS-8, Greedy and TSP, as well as a lower median, in both patrolling and pursuit-evasion. To determine the quantitative performance gain of our algorithm, we also performed a paired Student's t-test to compute 95% confidence intervals on the difference in performance between our algorithm and the benchmark algorithms. Table 1 reports the lower bounds of these confidence intervals (with the upper bound being ∞). It is clear that the clustering approach results in a significant improvement ($> 40\%$) over MS-8 in the office environment, confirming the superiority of using clustering for defining the domain of the path variables. More importantly, our algorithm outperforms all benchmarks by at least 30% in the office environment. Finally, for a more dynamic view of our algorithm, we have made videos available of several problem instances at `www.youtube.com/user/aaai2010`.

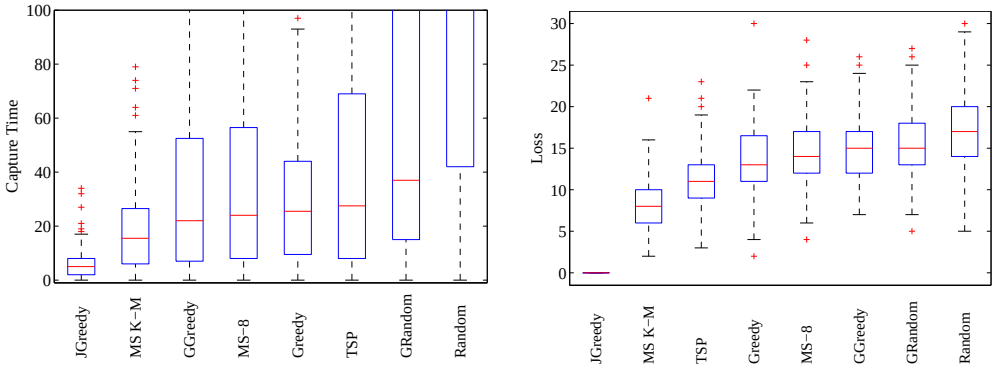
[7] Since the office environment has 350 vertices, the expected number of attacks is approximately 20.

[8] For example, by setting the attack probability too low, all algorithms will do fairly well, while by setting it too high, all algorithms will exhibit roughly equal poor performance.

[9] For a problem instance lasting for 100 time steps with 4 sensors taking approximately 15 measurements per time step, the expected number of false positives is 6.

[10] To compute the TSP cycle of the graph, we used Concorde. (`http://www.tsp.gatech.edu/concorde.html`).

[11] Since both graphs have an average branching factor roughly equal to 4, in most cases, these clusters are reached through all outgoing edges.



(a) Pursuit-Evasion (b) Patrolling

Figure 3: Empirical performance on 200 problem instances. The band near the center of the boxes is the median of the dataset; the box contains data points between the 25^{th} and 75^{th} percentile. Whiskers are drawn at 1.5 inter-quartile range (IQR).

	% Improvement (95% CI)			
Environment	**Greedy**	**GGreedy**	**MS-8**	**TSP**
PE Office	30.5%	39.5%	42.2%	52.2%
PE Lattice	26.9%	42.0%	10.0%	47.4%
P Office	43.7%	46.9%	45.2%	30.0%
P Lattice	32.4%	37.2%	21.4%	26.1%

Table 1: Lower bound of the 95% confidence intervals of performance increase of MS K-M compared to the four most competitive benchmarks.

6 Conclusions

In this paper, we developed a decentralised coordination algorithm for mobile sensors for a wide range of information gathering tasks. Our algorithm generalises the one proposed by Stranders et al. (2009) to a much richer set of domains (including monitoring spatial phenomena for which it was originally designed) and extends it by introducing an adaptive clustering technique for defining the paths over which coordination takes place. We applied our algorithm to pursuit-evasion (PE) and patrolling (P), and thereby obtained the first online decentralised coordination algorithms for these domains. Furthermore, we benchmarked our algorithm against the state of the art and showed a significant performance gain. Finally, we show that our adaptive clustering technique results in a $> 40\%$ performance increase compared to using a fixed set of paths as proposed by Stranders et al. (2009) on a non-lattice graph. For future work, we intend to extend our algorithm to compute solutions with a guaranteed approximation ratio for any planning horizon in order to guarantee a level of optimality of the sensors' actions.

Acknowledgments

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) Project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership (EP/C 548051/1).

References

Agmon, N.; Kraus, S.; and Kaminka, G. A. 2008. Multi-robot perimeter patrol in adversarial settings. In *Proc. of ICRA*, 2339–2345.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proc. of AAMAS*, 57–64.

Bopardikar, S. D.; Bullo, F.; and Hespanha, J. P. 2008. A pursuit game with range-only measurements. In *Proc. of the 47th Conf. on Decision and Contr.*, 4233–4238.

Borie, R.; Tovey, C.; and Koenig, S. 2009. Algorithms and complexity results for pursuit-evasion problems. In *Proc. of IJCAI*, 59–66.

Farinelli, A.; Rogers, A.; Petcu, A.; and Jennings, N. R. 2008. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proc. of AAMAS-08*, 639–646.

Halvorson, E.; Conitzer, V.; and Parr, R. 2009. Multi-step multi-sensor hider-seeker games. In *Proc. of IJCAI*, 159–166.

Hespanha, J. P., and Prandini, M. 2002. Optimal pursuit under partial information. In *Proc. of the 10th Mediterranean Conf. on Contr. and Automat.*, 469–479.

Hespanha, J. P.; Kim, H. J.; and Sastry, S. 1999. Multiple-agent probabilistic pursuit-evasion games. In *Proc. of the 38th Conf. on Decision and Contr.*, volume 3, 2432–2437.

Krause, A.; Leskovec, J.; Guestrin, C.; VanBriesen, J.; and Faloutsos, C. 2008. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management* 134(6):516–526.

Meliou, A.; Krause, A.; Guestrin, C.; and Hellerstein, J. M. 2007. Nonmyopic informative path planning in spatio-temporal models. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, 602–607.

Parsons, T. 1976. Pursuit-evasion in a graph. *Theory and Application of Graphs* Y. Alani and D. R. Lick, eds.:426–441.

Paruchuri, P.; Pearce, J.; Tambe, M.; Ordonez, F.; and Kraus, S. 2007. An efficient heuristic approach for security against multiple adversaries. In *Proc. of AAMAS*, 1–8.

Sak, T.; Wainer, J.; and Goldenstein, S. K. 2008. Probabilistic multiagent patrolling. In *Proc. of SBIA*, 124–133.

Singh, A.; Krause, A.; and Kaiser, W. J. 2009. Nonmyopic adaptive informative path planning for multiple robots. In Boutilier, C., ed., *IJCAI*, 1843–1850.

Stranders, R.; Farinelli, A.; Rogers, A.; and Jennings, N. 2009. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proc 21st Int. Joint Conf on AI (IJCAI)*, 299–304.

Vidal, R.; Rashid, S.; Sharp, C.; Jin, S.; and Sastry, S. 2001. Pursuit-evasion games with unmanned ground and aerial vehicles. In *in Proc. of IEEE ICRA*, 2948–2955.