A Markov Model for Multiagent Patrolling in Continuous Time

Jean-Samuel Marier, Camille Besse, and Brahim Chaib-draa
{marier,besse,chaib}@damas.ift.ulaval.ca

Department of Computer Science and Software Engineering, Laval University, Quebec, Canada

Abstract. We present a model for the multiagent patrolling problem with continuous-time. An anytime and online algorithm is then described and extended to asynchronous multiagent decision processes. An online algorithm is also proposed for coordinating the agents. We finally compared our approach empirically to existing methods.

Keywords: Multiagent, Unmanned Autonomous Vehicle, Online, Patrol.

1 Introduction

Unmanned Aerial Vehicles (UAVs) are a promising technology for many information gathering applications. As these devices become cheaper, more robust and have increased autonomy, we can expect to see them used in many different new applications such as surveillance and patrol missions. In such missions, the status of some sites must be monitored for events. If an UAV must be close from a location to monitor it correctly and the number of UAV does not allow covering each site simultaneously, a path planning problem arises: how should the agents visit the locations in order to make sure that the information about all locations is as accurate as possible?

In the last decade, several patrolling algorithms have been developed. For instance, Santana *et al.* [1] proposed a graph patrolling formulation on which agents use reinforcement learning on a particular Markov Decision Process (MDP). They defined this MDP over a countably infinite state space, leading to long and costly computations. Their approach also assumes that agents communicate by leaving messages on the nodes of the graph, leading to unrealistic communication models. On the other hand, reactive algorithms such as the ant-colony approach from [2] have been shown to perform well in theory as well as empirically. However such approach also rely on simplistic communication models relying on so-called pheromones.

In the case where all locations are equally important, Chevaleyre [3] proved that the shortest Hamiltonian circuit is an optimal solution for a single agent and a complete empirical study [4] shows that multiagent strategies using a unique cycle are the best whatever the graph is. However, as some locations may be more important than others, not visiting the less important ones from time to time may be advantageous.

In this paper, we propose a new graphical formulation of the patrolling problem featuring uncertainty on sensors and on information retrieval located on nodes of the graph and importance of certain locations over others. Contrary to previous approaches

that tried to minimize the idleness of the vertices of the graph, our model outlines the information retrieval aspect of the patrolling problem and the decay of the information value as the age of the information retrieved increases, forcing the agent to update its knowledge about nodes as frequently as possible. Our approach has a continuous-time formulation, allowing real durations and asynchronicity and it is readily usable for scenarios where durations are uncertain. We also present a planning algorithm that is suitable for the proposed formulation.

The remainder of this paper is structured as follows: section 2 presents the proposed model for patrol problems, section 3 presents the multiagent MDP framework and a mapping of the problem at hand as a continuous-time MDP, an online algorithm for solving the problem is presented in section 4, and section 5 presents experiments. Finally sections 6 and 7 discusses results and future work.

2 Problem Formulation

The patrolling problem has a graphical structure. Let us use V for the vertex set of that graph and E for its edge set. Let L be an $|V| \times |V|$ matrix, in which L_{ij} is a real number that represents the time required to go travel from i to j if $[i, j] \in E$ and is infinite otherwise. Each vertex i has a real *non-negative* importance weight, noted w_i. We note $\mathbf{w}$ the vector of all such weights.

In patrolling literature such as [2], idleness is used as a performance measure. The idleness of vertex i, noted τ_i represents the time since the last visit of an agent to that vertex. The idleness is 0 if and only if an agent is currently at vertex i and $\tau_i^{t+\Delta t} = \tau_i^t + \Delta t$ if there are no visits to i in the time interval $(t, t+\Delta t)$. Because idleness is an unbounded quantity, a more suitable representation is to use $k_i^t = b^{\tau_i^t}$, with $0 < b < 1$. We call this exponential idleness. Since k_i^t is always in $[0, 1]$, it can be seen as the expected value of a Bernoulli random variable which has value 1 if vertex i is observed correctly and 0 otherwise. Thus, k_i^t is the probability that this random variable is 1 at time t.

The probability evolves as $k_i^{t+\Delta t} = k_i^t b^{\Delta t}$ if there are no visits to i during time interval $(t, t+\Delta t)$. If an agent with noisy observations visits i at time t, idleness becomes 0 with probability $b < (1 - a) \leq 1$. If n agents visit vertex i at time $t + \Delta t$ and that there were no visits since time t:

$$k_i^{t+\Delta t} = k_i^t a^n b^{\Delta t} + 1 - a^n. \tag{1}$$

To sum up, an instance of the patrolling problem is a tuple $\langle L, \mathbf{w}, a, b \rangle$, consisting respectively of the matrix L of edge lengths, the vector $\mathbf{w}$ of importance weights and parameters a (the probability that the idleness does not become 0 when an agent visits a vertex) and b (the rate at which k_i decays over time).

3 Multiagent Markov Decision Processes (MMDPs)

This section casts the previous problem as a Multiagent MDP (MMDP). We assume that the problem state is fully observable, i.e. every agent has the same complete information

to make its decision. Such problems are called MMDPs. In the patrolling problem however, the actions of each agent have a concurrent effect on the the environment and they are of different durations. Concurrency in decision processes is conveniently modeled with a Generalized Semi-Markov Decision Process (GSMDP), introduced by Younes *et al.* [5]. Such decision processes also generalize MMDPs to continuous-time with asynchronous events. We use a restricted GSMDP and hence do not present that framework thoroughly.

The state variables for this problem describe the position of each agent and the idleness of each vertex (as per equation (1)). If the total number of agents is N, the state space is

$$\mathcal{S} = V^N \times [0,1]^{|V|}.$$

Given some state $s = (\mathbf{v}, \mathbf{k}) \in \mathcal{S}$, v_i is the position of the i-th agent and k_i the idleness of the i-th vertex. We use $s^t = (\mathbf{v}^t, \mathbf{k}^t)$ for the state and its components at time t.

At various time points, called decision epochs, the agents must choose an action. The actions from which an agent can choose from depend on the structure of the graph and on its position: if an agent is at vertex v, it can choose its action from $\mathcal{A}_v = \{u : [v,u] \in E\}$. If an agent chooses action u from vertex v at time t^i, the next decision epoch for that agent occurs at time $t^{i+1} = t^i + L_{vu}$, and $v^t = v$ while $t \in [t^i, t^{i+1})$ and $v^t = u$ as soon as $t = t^{i+1}$.

The problem is concurrent because the decision epochs of all agents can be interleaved arbitrarily. Each component k_i of $\mathbf{k}$ evolves independently. Equation (1) was defined in terms of two time epochs (t and $t + \Delta t$) and the number of agents (n). Let $\{t^j\}_j$ be the non-decreasing sequence of decision epochs and write n_i^j for the number of agents arriving at vertex i at time t^j. To simplify notation, let $\Delta t^j = t^{j+1} - t^j$. We thus have that

$$k_i^{t^{j+1}} = k_i^{t^j} a^{n_i^{j+1}} b^{\Delta t^j} + 1 - a^{n_i^{j+1}}.$$

The reward process $\mathcal{R}$ is defined in terms of $\mathbf{k}$. Specifically, the rate at which reward is gained is given by

$$dR = \mathbf{w}^\top \mathbf{k}^t \, dt. \tag{2}$$

The discounted value function for a GSMDP is defined by Younes *et al.* [5]. In our problem, it becomes:

$$\begin{aligned} V^\pi(s) &\underset{(a)}{=} \mathbb{E}\left[\int_0^\infty \gamma^t dR\right] \underset{(b)}{=} \mathbb{E}\left[\sum_{j=0}^{\infty} \gamma^{t^j} \int_0^{\Delta t^j} \gamma^t \mathbf{w}^\top \mathbf{k}^t \, dt\right] \\ &\underset{(c)}{=} \mathbb{E}\left[\sum_{j=0}^{\infty} \gamma^{t^j} \mathbf{w}^\top \mathbf{k}^{t^j} \frac{(b\gamma)^{\Delta t^j} - 1}{\ln(b\gamma)}\right], \end{aligned} \tag{3}$$

where $\gamma \in (0,1]$ is the discount factor. Equality (a) is the definition of the continuous-time discounted value function, (b) is obtained by noticing that (2) only exists between decision epochs and (c) is obtained by evaluating the integral. The expectation is taken over action durations. In this paper we consider deterministic durations. The sequence of states and decision epochs encountered depends on the actions chosen by the agents,

which are denoted by π. The problem is to chose actions for all agents and decision epochs that will maximize this expectation.

4 Solving the Patrolling Problem

Online planning has the advantage that it solves (3) only for the current state. This is in contrast with offline algorithms that do so for all states. Online algorithms are more appealing for systems that need to make a decision for the situation at hand. The problem described in section 3 is simpler to solve online than offline. We use an anytime online planning algorithm, which is described in section 4.1. We describe how it can be applied to the patrol problem in sections 4.2 and 4.3. An algorithm to coordinate multiple agents while retaining the online and anytime properties in section 4.4. An algorithm is anytime if firstly it can be stopped at any time and provide a useful result and secondly running it any longer does not degrade solution quality.

4.1 Anytime Error Minimization Search

Anytime Error Minimization Search (AEMS) is an online algorithm introduced originally by Ross *et al.* [6] for Partially Observable Markov Decision Processes (POMDPs). It performs a heuristic search in the state space. The search proceeds using a typical branch and bound scheme. Since the exact long term expected value of any state is not exactly known, it is approximated using upper and lower bounds. AEMS guides the expansion of the search tree by greedily reducing the error on the estimated value of the root node. While we are not in a strict POMDP setting, the greedy error reduction is useful. In our problem, actions have the same interpretation as in a partially observable setting, whereas observations are the travel durations. Using many such "observations", our model is extendable to stochastic durations. Let us recall briefly how AEMS works.

In AEMS, the error is defined using the upper bound and the lower bound on the value of some state. Let $s \in \mathcal{S}$ be a state. We have $L(s) \leq V(s) \leq U(s)$ where $V(s)$ is the actual value of s, and $L(s)$ and $U(s)$ are the lower and upper bounds respectively. Given some search tree $\mathbb{T}$, whose set of leaf nodes is noted $\mathcal{F}(\mathbb{T})$, the bounds for the root node are estimated recursively according to

$$L(s) = \begin{cases} \hat{L}(s) & \text{if } s \in \mathcal{F}(\mathbb{T}) \\ L(s, \boldsymbol{a}) = \max\limits_{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}} R(s, \boldsymbol{a}) + \gamma L(\tau(s, \boldsymbol{a})) & \text{otherwise.} \end{cases} \tag{4}$$

and

$$U(s) = \begin{cases} \hat{U}(s) & \text{if } s \in \mathcal{F}(\mathbb{T}) \\ U(s, a) = \max\limits_{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}} R(s, \boldsymbol{a}) + \gamma U(\tau(s, \boldsymbol{a})) & \text{otherwise,} \end{cases} \tag{5}$$

where $\tau(s, \mathbf{a})$ is the next state if action $\mathbf{a}$ is taken in state s. In equations (4) and (5), $\hat{L}(s)$ and $\hat{U}(s)$ are problem-dependent heuristics such as those proposed in section 4.2.

An estimation of the error on the value of s is given by $\hat{e}(s) = U(s) - L(s)$. Let s^0 be the state at the root of search tree $\mathbb{T}$. We are interested in expanding the state at the fringe of the search tree whose contribution to the error on s^0 is maximal. Since all states

are not reachable with equal probability (depending on the policy), the contribution of any state s to the error on s^0 is approximated by:

$$\hat{E}(s^0, s^t, \mathbb{T}) = \gamma^t \Pr(h_0^t | s^0, \hat{\pi}) \hat{e}(s^t),$$

where t is the depth of s in $\mathbb{T}$, and $\Pr(h_0^t | s^0, \hat{\pi})$ denotes the probability of having history h_0^t (the sequence of joint-actions that lead from s^0 to s^t), while following policy $\hat{\pi}$. Note that the above term describes exactly the error whenever $\hat{\pi} = \pi^*$. The value of $\Pr(h_0^t | s^0, \hat{\pi})$ is what we are now interested in.

If $h_0^t = \boldsymbol{a}^0, \boldsymbol{o}^0, \boldsymbol{a}^1, \boldsymbol{o}^1, \ldots, \boldsymbol{a}^t, \boldsymbol{o}^t$, is the joint-action history for some sequence of states $s^0, s^1, \ldots s^t$, then we have

$$\Pr(h_0^t | s^0, \hat{\pi}) = \prod_{i=0}^{t} \Pr(\boldsymbol{a}^i = \hat{\pi}(s^i) | s^i) \Pr(\boldsymbol{o}^i | s^i, \boldsymbol{a}^i).$$

Since we do not know the optimal policy (this is what we are searching for), a good approximation is to use:

$$\Pr(\boldsymbol{a}|s) = \begin{cases} 1 & \text{if } U(s, \boldsymbol{a}) = \max_{\boldsymbol{a}' \in \boldsymbol{\mathcal{A}}} U(s, \boldsymbol{a}') \\ 0 & \text{otherwise.} \end{cases}$$

Given a search tree $\mathbb{T}$, rooted at s^0, AEMS tells us that the next state to expand is

$$\tilde{s}(\mathbb{T}) = \arg\max_{s \in \mathcal{F}(\mathbb{T})} \hat{E}(s, s^0, \mathbb{T}).$$

Each time a node $\tilde{s}$ is expanded, it is removed from $\mathcal{F}(\mathbb{T})$, its children are added to $\mathcal{F}(\mathbb{T})$ and the bounds of $\tilde{s}$ and its parents are updated. When an agent must choose an action, the action of maximum lower bound is chosen.

4.2 Bounds for the Patrolling Problem

In order to use AEMS, we must specify the lower ($\hat{L}(\cdot)$) and upper ($\hat{U}(\cdot)$) bounds for the value of states. The alternate representation of idleness introduced in section 2 makes the reward and value functions bounded. It is thus possible to provide upper and lower bounds for the value function.

A lower bound for the value of any state is the value of following any policy from that state. A greedy policy is arguably a good "simple" policy. It is defined to always choose the action with arrival state for which $\mathbf{w}^\top \mathbf{k}$ is maximal. Equation (3) defines the value of such a policy.

An upper bound is usually obtained by relaxing problem constraints. We can thus upper-bound the value of a policy by assuming that agents are ubiquitous: they can be in more than one locations at the same time. Whenever an agent reaches a vertex, it instantaneously multiplies itself and starts heading to adjacent unvisited locations. This bound estimates the shortest time that a swarm of agents would take to cover the entire graph and estimates through the discount factor an upper bound on the maximum reward obtainable. This bound implicitly assumes that the optimal policy does not require having more than one agent at any vertex.

4.3 Extension to Asynchronous Multiagent Setting

Extending AEMS to a asynchronous multiagent is simple: whenever a node is expanded, there is a branch for every joint action (and observation). Asynchronicity is handled with state augmentation. The state is now $(s, \boldsymbol{\eta})$, where η_i is the time remaining before the next decision epoch of agent i. At any time t, the next decision epoch happens at time $t + \min_i \{\eta_i\}$. The expand operation adds actions and observations for any agent for which $\eta = 0$. Whenever agent i performs an action of duration Δt, η_i is assigned Δt. Otherwise, η_i is updated according to its depth in the search tree.

4.4 Coordinating Agents

AEMS can be used to perform online planning for any subset of agents. However, it is unlikely that any agent has the computational capacity to compute a joint policy, because the complexity is exponential in the number of agents. We thus coordinate agents *locally*. We define an partial order amongst agents. (Think of a directed acyclic graph where agents are vertices and there is an edge between two agents whenever they are close.) We say that an agent is greater than (resp. less than) another agent if it must choose its policy before (resp. after).

The agents compute their policy according to that order. Once an agent knows the policies of all greater agents, it proceeds to compute its policy, and then communicates it to the lesser agents. Whenever an agent selects its policy, it chooses the best policy given the policy of greater agents. This approach can be improved by using *altruistic* agents. Loosely speaking, such an agent does not consider only the value of its own policy. Instead it chooses the best sub-policy from a *small* set of joint policies: i.e. the joint policies for himself and a small number (i.e. 1) of lesser neighboring agents.

A useful property of this coordination algorithm is that if the agents use an online anytime planner, then it is also anytime and online. A fallback strategy is to ignore the presence of the greater agents until their policy has been received. For the remainder of this paper, we use C-AEMS to refer to this coordinated AEMS.

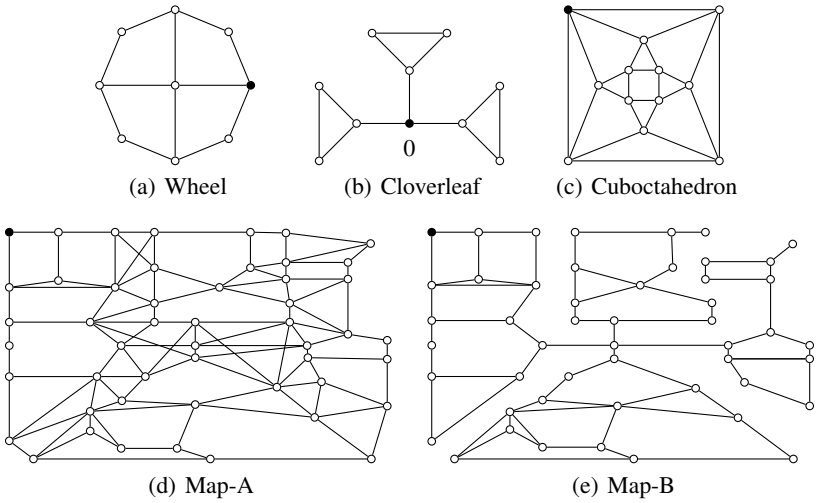
5 Experiments

Comparisons are made against a reactive algorithm proposed by Machado *et al.* [4] since it handles uncertainties and weights on locations. This algorithm is actually equivalent to the lower bound we used. We also compare our algorithm to the optimal policy value for some specific graphs that have an Hamiltonian cycle.

While our algorithm maximizes expected reward, idleness is often used a performance measure for patrol planning algorithms. Idleness is a useful measure because it is more readily interpretable than the average reward rate. The instantaneous idleness is the maximum idleness among all vertices at some given time. The maximum idleness is the maximum instantaneous idleness throughout an episode whereas the mean idleness is the mean instantaneous idleness. For the idleness measure to be meaningful, we have chosen $\mathbf{w} = \mathbf{1}$ and $a = 0$. (It is easier to verify that a patrol route is optimal in that case.) We have used $\gamma = 0.95$ throughout. The choice of b is discussed in section 6. The

initial state is $\mathbf{k} = \mathbf{1}$ for all experiments and the start vertex is shown as filled circles in Figure 1.

Results presented in Table 1 show the performance of our algorithm, for two agents, on various graphs instances presented in Figure 1. The small problem instances were chosen for their simplicity and also because the optimal policy is somewhat obvious. The wheel and cuboctahedron instances are interesting because of their symmetry. The two large instances, Map-A and Map-B, are seen frequently in the literature, for instance in Santana *et al.* [1]. The AEMS algorithm was allowed to expand 50 states on the three small instances and 100 on the larger ones.



(a) Wheel (b) Cloverleaf (c) Cuboctahedron

(d) Map-A (e) Map-B

Fig. 1. Patrolling instances. Start vertex is filled, edges have unit length and vertices have unit weight unless otherwise noted.

6 Discussion

The parameters of the model, (γ, $\mathbf{w}$, a and b) influence greatly the policy found by our algorithm. If b is too small, locations that have not been visited for a long time can remain unvisited without significant value loss. In such cases, the route of maximum value does not correspond to a low idleness policy. In practice, the representation of $\mathbf{k}$ is also subject to underflow if b is too small. If there is a circuit of length L, a value of b that would incite agents to visit all vertices is one such that $0.5 \approx b^L$. Note that while the policy generated by the reactive algorithm does not depend on a or b, the rewards it receives depends on b. Since b represents the rate at which the vertices change, it is typically not a tunable parameter. Interesting problems about b include finding how many agents are required to attain a given performance level and what is b given an actual problem instance. Although we did not explicitly state it in the definition of the

Table 1. Results for 2 agents on different graphs

	b	Max Idleness	Mean Idleness	Total Reward
Wheel (50 time units)				
C-AEMS	0.9	**5**	**3.90**	**144.512**
Greedy		6	4.24	142.957
Clover (50 time units)				
C-AEMS	0.9	**8**	**5.76**	**153.255**
Greedy		11	6.19	150.364
Cuboctahedron (50 time units)				
C-AEMS	0.9	**5**	**4.29**	**183.317**
Greedy		6	4.33	182.873
Map-A (200 time units)				
C-AEMS	0.95	**37**	**26.34**	**5718.02**
Greedy		51	32.58	5567.42
Map-B (200 time units)				
C-AEMS	0.99	**59**	**38.62**	**8708.29**
Greedy		67	41.34	8477.87

model, b could be different from one vertex to another and a could be different for every agent-vertex pair.

Results suggest that for given instances, and some values of b, the model allows finding good solutions with regard to the max-idleness metric. This is not obvious, because our reward is based on the exponential idleness of section 2. Our approach slightly outperforms the baseline approach on the instances, whereas the improvement is more significant on the large ones. We attribute this to non-myopia and to more efficient coordination between agents. On the three small instances, our algorithm found the optimal patrol route.

A problem with C-AEMS is that it does not perform well on highly symmetric instances such as the wheel or the cuboctahedron. In such cases, lot of computational effort is spent in discriminating patrol routes that have exactly the same value which is not possible. AEMS will then behave like breadth-first search. Such problems happen especially at the beginning of a simulation, because almost all vertices have never been visited and have equal value.

Our C-AEMS spends a significant portion of its time computing the value of the lower bound. The algorithm performs better when the bound is computed accurately. However, to get a lower error, the value must be evaluated over a longer time span. Computing the value of the lower bound offline would alleviate this problem.

7 CONCLUSION

We defined a model for the stochastic multiagent patrolling problem using Markov models. The proposed model allows specifying time uncertainty and concurrent asyn-

chronous actions by the agents. We propose an online algorithm to solve the problem for a subset of agents and a method for coordinating many such solvers.

The algorithm can be implemented on physical robots that must perform patrol under the assumption that they can communicate each other's policies reliably. The robots must be provided with a way to estimate the time until they reach the patrol location they are currently heading to. The proposed framework supports distributions on travel durations. Future work includes improving the performance of the lower bound with supervised learning and performing experiments in a setting where the agents evolve in a simulated world and to eventually allow actual UAVs to patrol a set of locations.

References

1. Santana, H., Ramalho, G., Corruble, V., Ratitch, B.: Multi-agent patrolling with reinforcement learning. In: Proc. of AAMAS'04. (2004)
2. Arnaud Glad, Olivier Simonin, Olivier Buffet, François Charpillet: Theoretical study of ant-based algorithms for multi-agent patrolling. In: Proc. of ECAI'08. (2008) 626–630
3. Chevaleyre, Y., Sempé, F., Ramalho, G.: A theoretical analysis of multi-agent patrolling strategies. In: Proc. of AAMAS'04. (2004)
4. Machado, A., Ramalho, G., Zucker, J.D., Drogoul, A.: Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures. In: Proc. of the Multi-Agent Based Simulation Conference. (2002)
5. Younes, H.L.S., Simmons, R.G.: A formalism for stochastic decision processes with asynchronous events. In: Proc. of AAAI Workshop on Learning and Planning in Markov Processes, AAAI Press (2004) 107–110
6. Ross, S., Chaib-draa, B.: AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs. In: Proc. of IJCAI'07. (2007) 2592–2598