Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

A HYBRID APPROACH BASED ON MULTI-AGENT GEOSIMULATION AND REINFORCEMENT LEARNING TO SOLVE A UAV PATROLLING PROBLEM

Jimmy Perron
Jimmy Hogan

NSim Technology
4715 Ave des replats #225
Quebec, G2J 1B8, Canada

Bernard Moulin

Laval University
Computer Science Department
Québec, QC G1V0A6, Canada

Jean Berger
Micheline Bélanger

Decision Support Systems Section
DRDC Valcartier, 2459 Pie-XI Blvd Nord
Quebec, QC G3J 1X5, Canada

ABSTRACT

In this paper we address a dynamic distributed patrolling problem where a team of autonomous unmanned aerial vehicles (UAVs) patrolling moving targets over a large area must coordinate. We propose a hybrid approach combining multi-agent geosimulation and reinforcement learning enabling a group of agents to find near optimal solutions in realistic geo-referenced virtual environments. We present the COLMAS System which implements the proposed approach and show how a set of UAV can automatically find patrolling patterns in a dynamic environment characterized by unknown obstacles and moving targets. We also comment the value of the approach based on limited computational results.

1 INTRODUCTION

Over the past few years the use of unmanned aerial vehicles (UAVs) for public safety and military operations has grown continuously. Typical applications include reconnaissance sorties, surveillance of known targets, search-and-destroy missions and search-and-rescue operations. The COLMAS (COordination Learning in Multi-Agent System) Project aims at developing a framework, algorithms, and automated advisory decision support capabilities for dynamic distributed resource management in which a heterogeneous team of agents drawn from distinct classes (static and moving airborne/land vehicles, unmanned/ manned vehicles) are engaged in a surveillance mission (reconnaissance, target search including detection/ recognition, information gathering, exploration, etc.) evolving in a dynamic uncertain environment with both known and unknown targets and threats (a mix of moving/static, evading/non-evading behaviors). In this context, UAVs present some special interest for their abilities to coordinate the simultaneous coverage of large surveillance areas and to cooperate in order to achieve various tasks such as target monitoring.

In this work we primarily focus on learning team coordination to achieve target allocation and navigation tasks for a distributed patrolling problem instantiated as a team of UAVs continuously monitoring possibly moving targets over a given region. The challenge is to solve such a problem taking into account the geographic characteristics of the environment and the fact that targets may move in an unpredictable manner.

In order to address such a problem, it seems natural to use a multi-agent approach to represent UAVs that may carry out their tasks autonomously, while trying to coordinate their collective action. In recent years several researchers (Machado et al. 2002; Chevaleyre 2004; Santana et al. 2004) proposed multi-agent solutions to a simpler version of the patrolling problem, considering that: 1) the territory can be represented by an undirected graph; 2) targets are fixed; 3) the patrolling task consists of continuously visiting all graph nodes in order to minimize the time lag between two visits.

Different techniques have been proposed to coordinate patrolling agents such as 1) Formulating the patrolling problem as a combinatorial optimization problem and extending solutions of the *Traveling Salesman Problem* (TSP) to multiple agents (Chevaleyere 2004); 2) Market-based multi-agent negotiation techniques (Almeida et al. 2004) where a traveling agent that cannot visit a target in a reasonable time, uses an auction mechanism inviting other team members to bid; 3) Techniques based on *swarm intelligence* (Charrier et al. 2007; Nowak et al. 2007); 4) Techniques based on *ant colony optimization* (Lauri and Charpillet 2006); 5) Multi-agent learning techniques based on *Reinforcement Learning* (RL) which allows agents to continuously adapt their patrolling strategies (Santana et al. 2004).

In the approaches presented above, relative target positions are represented by an undirected graph. However, in the case of a real geographic environment in which targets behave dynamically and obstacles (e.g. dangerous no-flight zones) may show up unpredictably, a graph representation is not a suitable representation and graph-based solutions to the patrolling problem do not apply.

As an alternative representation, we propose to use a multi-agent geosimulation approach (Moulin et al. 2003) in order to simulate the movements and interactions of autonomous agents (in our case UAVs) in a virtual geographic environment (VGE) generated from geo-referenced data contained in a Geographic Information System (GIS) (Benenson and Torrens 2004). To this end, we need to use a multi-agent geosimulation (MAGS) platform in which agents possess different capabilities that enable them to: 1) perceive other agents as well as the geographic characteristics of the VGE; 2) autonomously navigate; 3) avoid obstacles and 4) make decisions in order to achieve their goals (Moulin et al. 2003). Since targets and dynamic obstacles can also be represented by simple agents (or at least objects with reactive behaviors), we can use such an approach to plausibly simulate a dynamic environment in which UAVs will track moving land targets. In addition, we propose a hybrid approach based on multi-agent geosimulation and reinforcement learning that extends previous works (Santana et al. 2004) in order to enable a group of agents to autonomously coordinate their movements in order to patrol a large area in which targets may change locations and obstacles must be avoided.

The paper is outlined as follows. Section 2 introduces the patrolling problem while Section 3 presents the proposed solution Then, the COLMAS architecture and its components are described in Section 4. Section 5 shows how a user can define scenarios as an input to the COLMAS system. Section 6 illustrates some experiments demonstrating how the system can determine patrolling patterns. Section 7 presents some observations and a brief discussion. Finally, a conclusion is given in Section 8.

2 PROBLEM STATEMENT

In a surveillance scenario, a team of n agents (UAVs) patrolling an area continuously monitors a set of M land targets (shown by symbol "T" in Figure 1) moving freely and non-deterministically. The team objective is to determine UAV paths which minimize time lags between consecutive target visits over a given time horizon. A suitable UAV path solution would ideally correspond to a so-called *patrolling pattern*, defined as a semi-periodic sequence of agent movements among a subset of targets (pictured as a sequence of arrows in Figure 1). Our goal is to develop a system which automatically finds a combination of such individual patterns in order to find a global and optimal surveillance pattern.

There is no efficient solution known to such a problem since the number of possible combinations of UAVs' elementary actions in any realistic situation is very large, and there is no straightforward mathematical way of representing the spatial information (such as moving target locations and areas changing in size and locations). The proposed idea to take advantage of the analogical representation of space allowed by a geosimulation taking place in a geo-referenced virtual world (Moulin et al. 2003; Benenson et Torrens 2004).

As a first approximation to solve this problem, we consider that the agents' displacements in the VGE are defined in a simple way by actions of the type: *Go from current position to target m*. Thus, a patrolling pattern becomes a spatial trajectory. At every time step of the simulation, each agent must autonomously decide about its next move. The problem consists of computing the next action to be executed by each UAV. The combination of UAVs' patrolling patterns must result in a course of actions (plan) defining a global surveillance pattern of the area which minimizes certain criteria. The main criterion used to evaluate the quality of a solution of the patrolling problem relies on the notion of *Idlenes*s. Target idleness represents the time elapsed since it was last visited or similarly the time lag between two successive target visits.

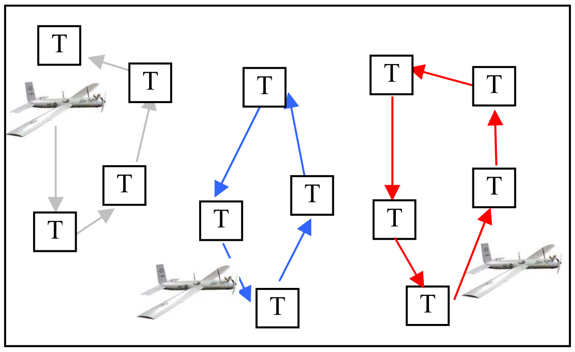


Figure 1: UAV's patrolling patterns

Let $M(t)$ be the number of targets at time t, and $V_m(t)$ the time at which target m was last visited.

Idleness of target m at time $\boldsymbol{t}$ represents the time elapsed between the last target 'm' visit and the current time: $\boldsymbol{I_m(t) = t - V_m(t)}$.

The Global Idleness of the system represents the sum of idlenesses over all targets at a given time $\boldsymbol{t}$: $\boldsymbol{I(t) = \sum_{m=0}^{M} I_m(t)}$.

The Average Global Idleness represents global time-weighted idleness over a time interval/horizon $\boldsymbol{T}$:

$$\bar{I}_T = \frac{1}{T}\int_T I(t)dt \approx \frac{1}{T}\sum_{k=1}^{nb\ of\ visits} I(t_k)\Delta t_k; \quad \sum_{k=1}^{nb\ of\ visits} \Delta t_k = T$$

t_k: k^{th} visit event (out of *nb of visits*) time

$\Delta t_k = t_k - t_{k-1}$: elapsed time between two visit events

The team objective is to determine UAVs' path (e.g. patrolling patterns) minimizing average global idleness.

In the graph-based approaches mentioned in Section 1, solving the patrolling problem consists of elaborating a multi-agent graph coverage strategy. Such a strategy must optimize a given quality criterion involving the instantaneous node idleness (Machado et al. 2002). Simulation is used in order to allow agents explore the problem state space and the different possible solutions are evaluated in order to keep the best ones. A multi-agent patrol strategy can be evaluated after T cycles of simulation using either the *average idleness* or *the worst idleness* criteria.

Santana et al. (2004) proposed an approach in which the agents are able to learn to patrol using Reinforcement Learning (RL). Each agent integrates a *Markov Decision Process* (MDP) that is used to determine which action to perform in every situation state. An action allows an agent to go to the adjacent nodes in the graph. A situation state corresponds to the minimal information required by an agent to decide what to do. An agent may immediately reach the adjacent target in a single cycle of execution. The distance it must travel to reach the target is interpreted as a cost rather than a distance. This cost is used to compute the reward associated with the state. Since the environment is represented as a graph where nodes correspond to targets and links to transition costs, the model of the environment is assumed to be known prior to creating the graph. This is not possible in the case of a complex dynamic geographic environment in which targets may change positions and obstacles may appear and move unpredictably.

3 A SOLUTION BASED ON GEOSIMULATION AND DISTRIBUTED REINFORCEMENT LEARNING

We propose a hybrid approach combining distributed reinforcement learning and geosimulation to handle task allocation (high-level planning) and navigation/routing (low-level planning) respectively. The hierarchical decomposition scheme can be characterized as follows. The distributed RL approach uses a simple problem state representation in computing an approximate solution to the learning problem, significantly reducing computational complexity. It promotes emergent behavior through suitable tradeoffs between solution space exploration and exploitation. Partial state observability and action execution are modeled through geosimulation allowing to capture realistic environments using agent visibility models and kinematic/collision avoidance/geographical constraints.

The internal state representation is an important implementation issue that needs to be solved when considering reinforcement learning algorithms. Indeed, the representation of a learning problem may have an important impact on performance, both of the learning algorithm and of its results. Due to the simplicity of its implementation, the classic *total representation* where all internal states must be known was selected. To use the total representation, the reinforcement learning problem definition must be modeled at a high-level of abstraction, limiting the number of internal states as well as the number of actions. This high level of abstraction can be reached using a hybrid architecture where the RL module is used to make high-level decisions and where a geosimulator executes low level actions. These low level actions must include all spatial actions like navigation, path finding, and obstacle avoidance.

Our solution takes advantage of a state-space representation similar to the one used by Santana while adding constraints from a real-time geosimulation system. The general workflow of the proposed system is presented in Figure 2. The perception module receives and processes relevant data that is obtained from the environment and that will be used to make a decision. The memory module takes into consideration the simulator's past state, if necessary, to create the RL state entry. This means that the model does not need complex learning algorithms to take past states into account. It is possible, for the decision module, to learn from past information using a simple Q-Learning algorithm.

Considering that a patrolling pattern consists of a sequence of targets to visit, it is possible to significantly reduce the dimension of the state space sent to the RL algorithm. The agent only needs to know the last target it has visited in order to decide which one it will visit next. For *m* targets, there will be *m* states. Once the action is chosen by the agent, the simulator executes this action.

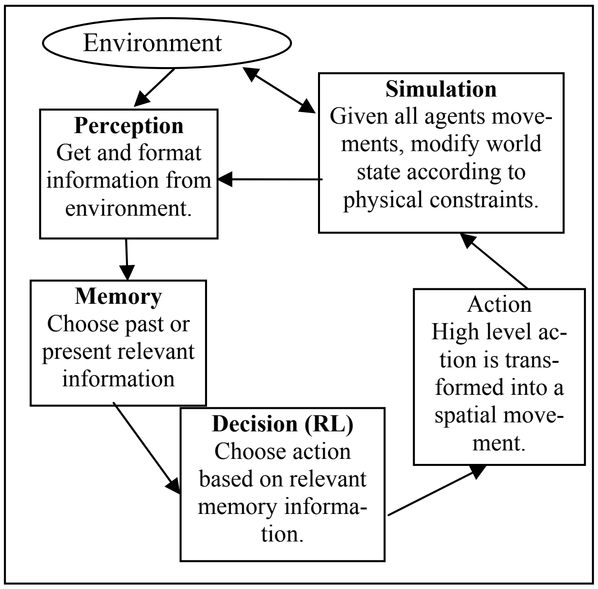

Environment
Simulation
Given all agents movements, modify world state according to physical constraints.
Perception
Get and format information from environment.
Memory
Choose past or present relevant information
Action
High level action is transformed into a spatial movement.
Decision (RL)
Choose action based on relevant memory information.

Figure 2: COLMAS general workflow

The RL module plays the following three roles in the system: 1) Map abstract system states to abstract actions; 2) Learn to interact with an unknown environment model; 3) Control an agent according to a learned behavior.

We define an agent behavior as its responses to external stimuli. An agent's responses include actions carried out to modify the state of the surrounding world and actions that modify its own state (i.e. deciding to remember or forget an element of the world).

The simulation module provides the following functions to the system: 1) Acts as a state transition function; 2) Process and formats simulation data for the decision module (RL); 3) Executes action planned by the decision module; 4) Acts as a realistic model of the spatial environment.

Hence, the simulation allows RL to process high-level plans. In the proposed solution, plans are in turn expanded to low level navigation actions, being processed by the simulation engine. Abstraction of the simulation's spatial dimension through simple state and action representation greatly simplifies the learning process of high level plans.

4 THE COLMAS ARCHITECTURE

COLMAS (COordination Learning in Multi-Agent System) is a hybrid system which couples a reinforcement learning approach (namely Q-Learning) for planning, with geospatial reasoning (using an agent-based geosimulator) for plan execution in order to enable a team of UAVs to autonomously navigate in a VGE and to coordinate their actions. Each agent is performing a plan (likely to evolve toward a stable patrolling pattern after a number of simulation steps). Key problem and environment characteristics include:

- Decentralization - the system is inherently decentralized, meaning that each agent only observes part of the global state of the system (local observation) and makes its own decision. Team reward is considered.
- Autonomy - the agents make their own individual decisions based on the available information, requiring a local decision process for each agent.
- Uncertainty - the system must be able to represent uncertainty in the system and also in each agent's behavior. Uncertainty in the environment will come from the simulation and from the decentralized multi-agent approach (each agent doesn't know which actions other agents' intent or current action being executed).
- Communication and Coordination - agents may not communicate explicitly and manage interdependency constraints. A supervisory central node mediates global (team) reward information-sharing.
- Spatial environment – the problem includes a dynamic spatial environment managed through a geosimulator.

Spatial information is recorded in a raster mode which enables agents to access the information contained in a number of bitmaps which encode different kinds of information about the terrain characteristics and the objects contained in the VGE. The *HeightMap* is a 2D matrix (or grid) which represents the space of the VGE. It is generated from data contained in a digital elevation model and different layers of the GIS data base. Every cell contains a single value indicating the height of the corresponding point relative to the point of lowest elevation in the VGE. The *ObstacleMap* is a 2D matrix based on a bitmap representation generated from GIS data in which each obstacle is identified by a set of cells associated with the same color characterizing the nature of the obstacle.

Each agent has a perception capability which enables it to collect data about its surroundings in the VGE. The agent also reasons on the collected data and chooses its next action (or goal) accordingly. Hence, the agent autonomously navigates and avoids collisions and obstacles in the VGE. A full radial perception capability enables an agent to perceive the three closest visible targets in the VGE in its parameterized range of perception. The navigation algorithm processes the perceived information and enables each agent to choose its next move (in the direction of the chosen target) while taking into account obstacles. For more details on perception and navigation algorithms see (Perron and Moulin 2004).

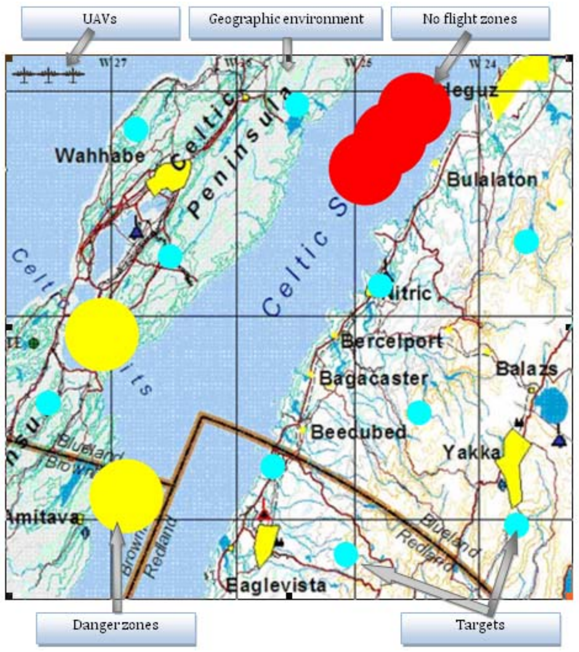


Figure 3: Specifying a scenario

Two navigation algorithms are used for plan execution:

- *Simple obstacle avoidance*: This algorithm tries to avoid no-flight zones in an efficient way (reactive algorithm). However, it may happen, with a particular no-flight zones layout, that the algorithm is unable to find a valid path between two targets (hence, the second algorithm).
- *A* planner*: This algorithm plans a path between two targets and finds an optimal solution to avoid obstacles. It is computationally more expensive.

A Q-learning algorithm inspired from (Watkins, 1989) and (Schneider et al. 1999) is used for planning (target visit selection). It learns an optimal policy through a function

Q(***s,a***) describing the quality of each state-action pairs in order to prescribe behavior (what action ***a*** to select in a state ***s***) to maximize team expected cumulative rewards (objective function). Agent Q(***s,a***) update occurs over a state transition (***s,s'***) and is based on reward ***r*** obtained by executing action ***a***. A separate control policy is used for solution space exploration to update the current optimal policy being learned through the Q values. The resulting learned policy consists in selecting the action ***a*** maximizing Q over a state ***s.*** Using state space abstraction, value function approximators and limited action sets to reduce computational complexity and focus on a given class of solution, the COLMAS learning process iterates as follows:

1. The agent's perception module processes environment sensor data to extract the current local state ***s***, then required to run the Reinforcement Learning algorithm. In the patrolling task, the needed information involves to the 5 next potential targets to visit, coupled with the last visited target stored in the agent's MemoryState.
2. The agent's exploration (control) policy (e.g. *SoftMax* (Sutton 2004)) selects next action ***a*** to execute (based on current Q values information).
3. The action ***a*** (high-level plan) is then executed (low-level navigation/plan execution) by the geo-simulator operating state transition (***s,s'***). Agent reward ***r*** can then be computed via a central node.
4. The learner's generic RL algorithm updates Q:
   (a) Get current state ***s***
   (b) Get the last executed action ***a***
   (c) Perceive next state ***s'***
   (d) Get reward ***r*** = RewardFunction(***s, s', a***)
   $\boldsymbol{r} \propto (-\boldsymbol{I}(\boldsymbol{t}_k)\,\Delta\boldsymbol{t}_k / \boldsymbol{T})$ over k^{th} transition (***s,s'***)
   (e) Update Q values. Q = Q-Function(***s, s', a, r***):
   $$\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a}) \leftarrow (1-\alpha)\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a}) + \alpha(\boldsymbol{r} + \gamma \underset{a'}{\boldsymbol{Max}}\,\boldsymbol{Q}(\boldsymbol{s}',\boldsymbol{a}'))$$

In the COLMAS system, the simulation is running continuously, progressively learning and improving solution policy. An interesting characteristic of the approach is a technique to detect and extract patterns from the last solution computed by the simulator. Considering that a greedy exploration policy is used in the RL, the system searches for a stable pattern executed by each UAV which minimizes idleness. Navigation patterns are identified using a history of the 100 latest executed actions for each UAV. A pattern is defined by a recurrent trajectory.

5 SCENARIOS AND PROBLEM SPECIFICATION

Using COLMAS, a user specifies a situation to be solved as a scenario which contains the following objects as pictured in Figure 3:

- *Geographic environment*: a map of the area of interest.
- *Targets*: these objects that are visited by UAVs are represented by a colored circle positioned in the geographic environment.
- *UAV*: patrolling agent represented by a colored aircraft positioned in the geographic environment.
- *No flight zone*: area characterized by UAV flight interdiction (red circle). The UAV navigation algorithm imposes travel constraints on these areas which are considered as physical obstacles in the environment.
- *Danger zones*: A danger zone (yellow circle) is only informative and represents potential threats (as opposed to pure physical obstacles) to be avoided by UAVs. These zones are used by a metric in the simulation (*danger zone metric*) to compute how many times UAVs cross the danger zone for each solution.

In addition to scenario elements, the user can specify specific constraints. As illustrated in Figure 4, an editor allows the user to visually specify three types of constraints for each UAV, namely:

- *Desired paths*: represented by a colored arrow (a different color is used for each UAV) starting from an initial target to a final target. The user can specify a set of desired paths for each UAV in the scenario.
- *No-flight paths*: represented by a black arrow starting from an initial target to a final target. The user can specify a set of no-flight paths for each UAV in the scenario.
- *No-flight targets*: represented by a black target and specifies a target that must not be visited by a particular UAV. The user can specify a set of no-flight targets for each UAV in the scenario.

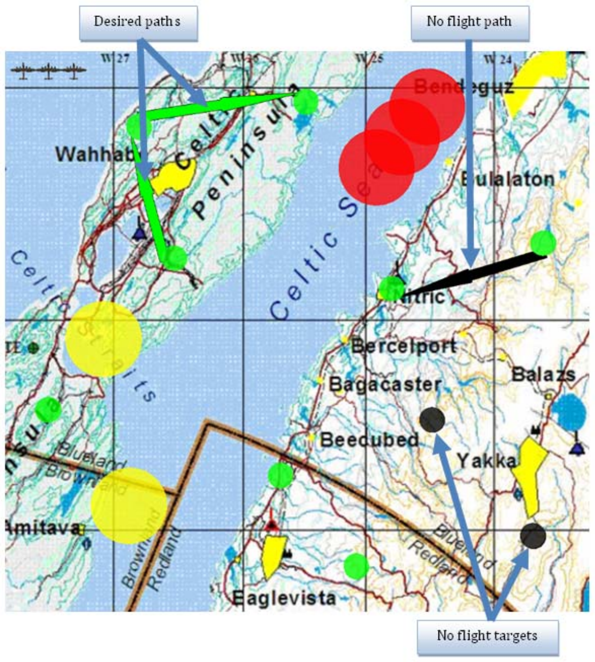


Figure 4: Specifying constraints

Once the problem and constraints are defined, COLMAS computes a solution to the patrolling problem as shown in Figure 5. When a solution is completed, it is evaluated by the evaluation module.

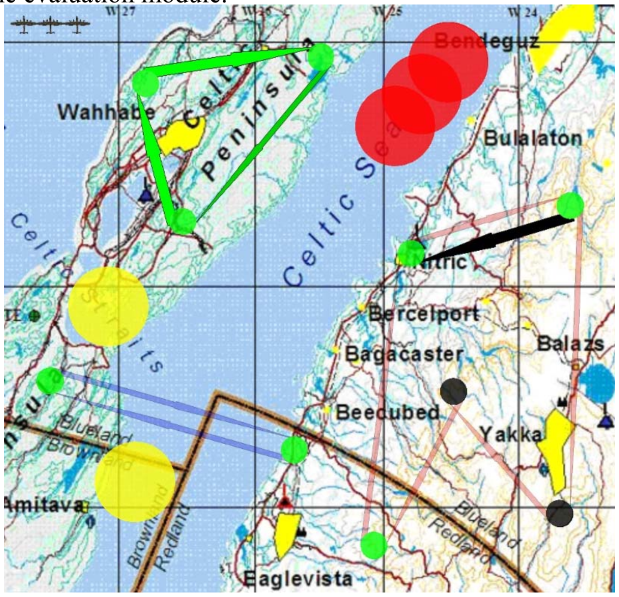


Figure 5: A solution with UAV patrolling patterns

The learning module uses the evaluation module as well as a background process to find solutions (using it as the reward function). The resulting solutions are then presented to the user as cyclic patterns using different colors, one for each UAV (Figure 5).

6 EXPERIMENTS AND RESULTS

Experiments have been carried out to test and illustrate the system capacity in converging toward good solutions (patrolling patterns) over various combinations of UAVs and numbers of targets.

Here is the experimentation methodology that we used:

- A scenario and a problem are specified (positions and number of targets, positions and number of obstacles in the environment, the number of UAVs used for the experiment)
- Initial parameters are set for the COLMAS system:
  - (a) Input state is the last visited target.
  - (b) The exploration policy used is the Greedy policy.
  - (c) The available agent action set corresponds to respective moves toward the 3 nearest targets.
  - (d) Reward is the average global idleness computed over 40 simulation cycles or alternatively weighted idleness over time intervals separating two successive visits (events).
  - (e) Each UAV uses a radial perception which gives a complete perception of the environment in the given radius (1km)
  - (f) Each UAV uses the obstacle avoidance navigation algorithm
- The COLMAS system is started and runs until it converges.
- The patterns are extracted and evaluated

Figure 6 presents the two patrolling patterns that one UAV has found to visit 4 targets (Experiment 1) as well as the pattern found by 2 UAVs patrolling 4 targets (Experiment 2), and how the patterns are modified when an obstacle is dynamically introduced in the environment (Experiment 3). This experiment shows how the UAVs are able to dynamically adapt the patrolling patterns to avoid obstacle and maximize the average global idleness.

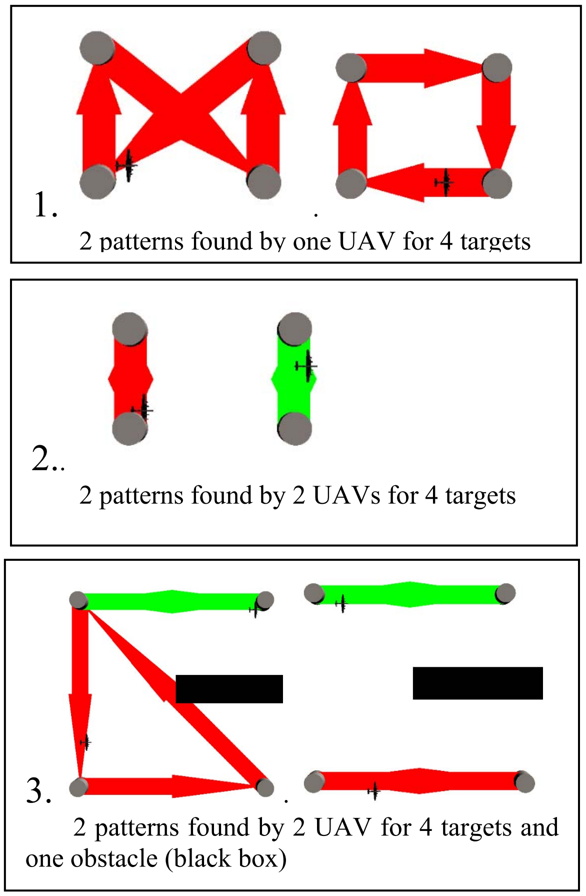

1.
2 patterns found by one UAV for 4 targets
2..
2 patterns found by 2 UAVs for 4 targets
3.
2 patterns found by 2 UAV for 4 targets and one obstacle (black box)

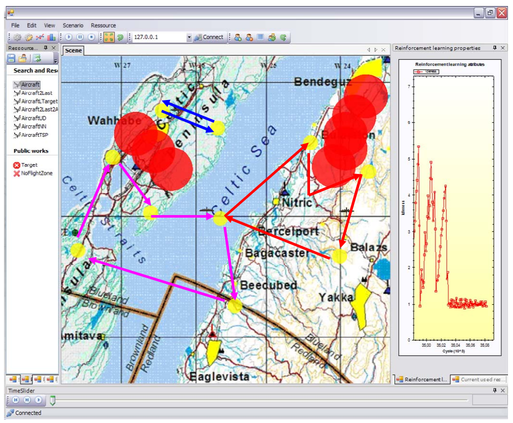
Figure 6: Experiments 1, 2 and 3

Figure 7 presents the pattern found by 3 UAVs patrolling 10 targets (Experiment 4), and the pattern found by 4 UAVs patrolling 16 targets (Experiment 5).

Further experiments are needed, but the results obtained so far look very promising. We observed that in Experiment 4, agents are able to find patterns which consider obstacles in the spatial environment layout. In Experiment 5, targets are located in the environment in a particular way to form clusters. The experiment shows that agents are able to find a good solution based on the geographic constraints formed by the cluster.

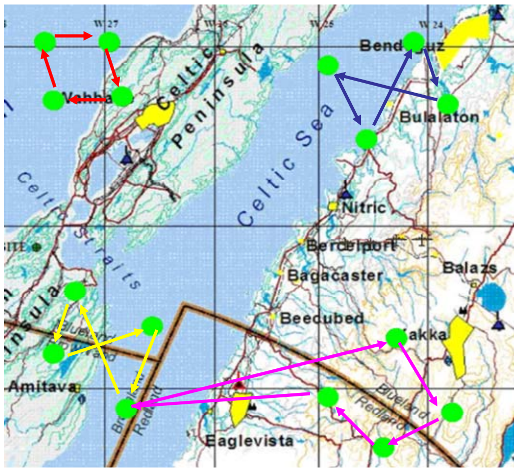
Agents initial positions have no impact on solution quality and convergence rate. This may be explained by the fact that it takes only a few iterations to the RL algorithm to cover all the targets when the exploration begins.

4.



1 pattern found by 3 UAVs for 10 targets

5.



1 pattern found by 4 UAVs for 16 targets

Figure 7: Experiments 4 and 5

7 DISCUSSION

In order to provide further insight on the value of our hybrid procedure, early results comparing performance with alternate techniques is discussed. Accordingly, limited scenarios involving static and dynamic (moving) targets have been considered. for the following algorithms, namely, the TSP-based procedure proposed by (Chevaleyre, 2004), a nearest neighbor strategy in which an agent patrolling policy consists in visiting the closest target, and a RL-based suite of methods (Schneider et al. 1999) incorporated to the COLMAS learning module for exploration purposes. It is worth mentioning that COLMAS' current implementation provided promising results, in terms of average global idleness of targets, providing the best solution for the *25 targets and 5 UAVs* scenario, while showing competitive results to the best computed solution in the remaining four scenarios.

Although additional experiments and related improvements to the COLMAS System might be further required, computational results conducted so far clearly show that an hybrid approach combining reinforcement learning and multi-agent geosimulation can efficiently solve the multi-agent patrolling problem in virtual geo-referenced environments involving moving targets and obstacles. This pushes a step further previous work relying on graph-based environment representation considering small number of fixed targets only.

It is believed that multi-agent solution convergence rate might be slightly improved using a biased policy exploiting some prior heuristics knowledge. These heuristics could help agents to preferentially explore actions leading to good patrolling patterns. Accordingly, a naïve distance-based heuristic has already been used in which the set of available actions only considers a subset of target visits, namely the closest ones to an agent. A more sophisticated heuristic might consist to select actions most likely to generate specific patrolling patterns (e.g. circle).

The main strengths of the proposed approach can be summarized as follows:

- Emergent behavior - agent patterns. Agents learn good patrolling solution patterns without prior knowledge of the environment model. That means that low level algorithms responsible for step by step navigation might be changed depending on the application.
- Abstract state space representation. The reduced state space resulting from state aggregation is polynomial in terms of the number of agents and targets due to state approximation/abstraction. Accordingly, despite uncertainty on solution convergence, agents learn their patterns very quickly even when there is a change in target configuration. Only few trials are needed to find new patterns.
- Decentralized solution. Agents do not explicitly communicate with one another, they learn patterns from their local perception.
- Online patterns discovery. RL can adaptively finds new patterns online when the number of targets evolves dynamically. Shown solution convergence for reasonable/practical problem size.

- Adaptive method. Despite suboptimal emerging patterns, computation is fast.
- Realistic environment. Real-world problem can be captured using geosimulation.

The current system weaknesses include:

- Data translation. The designer needs to code formatting algorithms for data which are inputs and outputs for the RL algorithm
- Predetermined behavior patterns. The solution cannot explicitly learn desirable patterns in advance; it must start from scratch with each target configuration change.
- Credit assignment. Proper reward definition reflecting a meaningful team objective contribution when selecting action ***a*** in state ***s***,
- Scaling - curse of dimensionality. The approach has difficulties scaling up due to large state spaces, and non-determinism associated with concurrent multiple agents' policies learning. Hierarchical problem decomposition and partitioning may nonetheless mitigate that problem.
- Q-learning convergence. No known proof of Q-learning convergence for our function approximation to the distributed decision patrolling problem investigated. Multi-agent solution convergence rate may ultimately depend on multiple parameter settings for increasing problem complexity.

8 CONCLUSION

A hybrid approach combining distributed reinforcement learning and geosimulation to handle target allocation (high-level planning) and navigation/routing (low-level planning) tasks for the dynamic distributed patrolling problem has been proposed. The hierarchical decomposition scheme can be characterized as follows. The distributed reinforcement learning approach uses an abstract problem state representation in computing an approximate solution to the learning problem, significantly reducing computational complexity. It promotes emergent behavior through suitable tradeoffs between solution space exploration and exploitation. Local state observability and action execution are modeled through geosimulation allowing to capture realistic environments using agents' visibility models and kinematic/collision avoidance/geographical constraints. A computational experiment has shown convergence of the proposed hybrid procedure for real-world problem instances providing emergent patrolling behavior patterns.

As an extension to this work, a separate effort aimed at capturing user's preferences inspired from our hybrid framework has been initiated. The proposed approach integrates user's preferences into the reinforcement learning solving process proposed in the COLMAS system. User's feedback allows the generation of realistic solutions for the patrolling problem. The extended approach enables the system to extract user's preferences and to calibrate the system. contributing to significantly reduce computational complexity, while providing additional guidance in exploring the solution space. The proposed approach exploits an inverse reinforcement learning technique which only relies on a single expert solution to automatically extract coefficients of relative importance which characterize user's preferences. More details on early results may be found in (Bélanger et al. 2008).

ACKNOWLEDGEMENTS

This research has been funded by the Defence Research and Development Canada – Valcartier (DRDC Valcartier, Quebec, Canada).

REFERENCES

Almeida, G., G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre. 2004. Recent advances on multi-Agent patrolling. Ed. A.L.C. Bazzan and S. Labidi, *SBIA 2004*, Springer Verlag Lecture Notes in AI 3171, 474–483.

Bélanger, M., J. Berger, J. Perron, J. Hogan, B. Moulin. 2008. Exploitation of user's preferences in reinforcement learning decision support systems Multidisciplinary Workshop on Advances in Preference Handling (W9). Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08). Chicago, 2008.

Benenson, I., and P. M. Torrens. 2004. *Geosimulation: Automata-Based Modeling of Urban Phenomena*. London: John Wiley & Sons.

Charrier, R., C. Bourjot and F. Charpillet. 2007. In *Proceedings of the IEEE International Conference on Self-Adaptive and Self-Organizing Systems - SASO 2007*, 32-44. IEEE.

Chevaleyre, Y. 2004. Theoretical analysis of the Multi-agent patrolling problem. *In Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 302-308. Beijing, China.

Lauri, F., and F. Charpillet. 2006. Ant Colony Optimization applied to the Multi-Agent Patrolling Problem, In the *Swarm Intelligence Symposium*, Indianapolis, Indiana, USA, IEEE.

Machado, A., G. Ramalho, J-D. Zucker, A. Drogoul. 2002. An empirical analysis of alternative Architectures. In Proceedings of the *3rd International Workshop on Multi-Agent Based Simulation*, 155-170.

Moulin, B, W. Chaker, J. Perron, P. Pelletier and J. Hogan. 2003. The MAGS Project: Multi-agent geosimulation and crowd simulation. In Kuhn, Worboys and Timpf eds., *Spatial Information Theory*, Springer Verlag LNCS 2825, 151-168.

Nowak, D.J., I. Price and G. B. Lamont. 2007. In *Proceeding of the 2007 Winter Simulation Conference*, ed. S.G. Henderson S. G. Henderson, B. Biller, M.-H Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1315-

1323. New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Perron, J., and J. Hogan. 2006. COLMAS Project - Technical survey and COLMAS architecture. Research Report CR 2006-601, DRDC Valcartier.

Perron, and B. Moulin. 2004. Un modèle de mémoire dans un système multi-agent de géo-simulation, *Revue d'Intelligence Artificielle*, 18(5-6) : 647-678.

Santana, H., G. Ramalho, V. Corruble, and B. Ratitch. 2004. Multi-agent patrolling with reinforcement learning. In *Proc. of the Third International Conference on Autonomous Agents and Multi-agent Systems*, 1122-1129.

Schneider, J., W.-K. Wong, A. Moore, and M. Riedmiller. 1999. Distributed value functions. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning*, 371-378. San Francisco, CA.

Sutton, R., and A. Barto. 2004. *Reinforcement Learning: An Introduction*. The MIT press.

Watkins, C. J. C. H. 1989. Learning from Delayed Rewards. *PhD thesis*. Cambridge University, Cambridge, England.

AUTHOR BIOGRAPHIES

JIMMY PERRON is a scientist with the research department of NSim Technology. He received his BS and MS degrees in computer science from Laval University, Québec, Canada. His research interests include artificial intelligence, multi-agent system and geosimulation applied in the field of decision support. jimmy.perron@nsimtech.com

JIMMY HOGAN is a scientist with the research department of NSim Technology. He received his BS degree in computer engineering and and MBA degree from Laval University, Québec, Canada. His research interests include artificial intelligence and simulation to help decision support system. jimmy.hogan@nsimtech.com

BERNARD MOULIN is a full professor at Laval University, teaching in the Computer Science and Software Engineering Department. He is also a member of the Research Center in Geomatics at Laval University and an active researcher of GEOIDE, the Canadian Network of Centers of Excellence in Geomatics. He leads several research projects supported by different Canadian institutions (Geoide, NSERC, FQRNT) in various fields: Multi-agent geosimulation, Design methods for multiagent systems and software-agent environments; representation of temporal and spatial knowledge in discourse; modeling and simulation of conversations between artificial agents; modeling and design approaches for knowledge-based systems and multiagent systems, as well as several projects at the intersection of geomatics and artificial intelligence. Bernard.moulin@ift.ulaval.ca

JEAN BERGER is a defense scientist with the Decision Support Systems Section of Defense Research and Development Canada - Valcartier, working in the field of information technology. He received BS and MS degrees in engineering physics from Ecole Polytechnique de Montréal, Canada. His research interests include artificial intelligence and operations research applied to intelligent control, simulation, planning, routing and scheduling problems. jean.berger@drdc-rddc.gc.ca

MICHELINE BÉLANGER is a defence scientist with Defence R&D Canada – Valcartier since 1990. As a member of the Decision Support Systems Section, she is investigating artificial intelligence and multicriteria decision aid concepts to develop decision support tools for command and control applications. Her current research interest is the integration of feedback approaches into decision support systems. She received a master's degree in computer science in 1990 from Université de Montréal and a bachelor's degree in mathematics and computer science in 1988 from Université du Québec à Rimouski. micheline.belanger@drdc-rddc.gc.ca