Multiagent Meta-level Control for a Network of Weather Radars

Shanjun Cheng, Anita Raja
Department of Software and Information Systems
The University of North Carolina at Charlotte
Charlotte, NC 28223
{scheng6, anraja}@uncc.edu

Victor Lesser
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
lesser@cs.umass.edu

Abstract

It is crucial for embedded systems to adapt to the dynamics of open environments. This adaptation process becomes especially challenging in the context of multiagent systems. In this paper, we argue that multiagent meta-level control is an effective way to determine when this adaptation process should be done and how much effort should be invested in adaptation as opposed to continuing with the current action plan. We develop a reinforcement learning based mechanism for multiagent meta-level control that facilitates the meta-level control component of each agent to learn policies in a decentralized fashion that (a) it can efficiently support agent interactions with other agents and (b) reorganize the underlying network when needed. We evaluate this mechanism in the context of a multiagent tornado tracking application called NetRads. Empirical results show that adaptive multiagent meta-level control significantly improves the performance of the tornado tracking network for a variety of weather scenarios.

1. Introduction

Embedded systems consisting of collaborating agents capable of interacting with their environment are becoming ubiquitous. These agents operate in an iterative three-step closed loop [9]: receiving sensory data from the environment, performing internal computations on the data, and responding by performing actions that affect the environment either using effectors or via communication with other agents. Two levels of control are associated with this loop: deliberative and meta-level control [6]. The lower control level is deliberative control (also called object level), which involves the agent making decisions about what domain-level problem solving to perform in the current context and how to coordinate with other agents to complete tasks requiring joint effort. At the higher control level is meta-level control, which involves the agent making decisions about whether to deliberate, how many resources to dedicate to this deliberation, and what specific deliberative control to perform in the current context.

Meta-level control in complex agent-based settings was explored in previous work [8] where a sophisticated architecture that could reason about alternative methods for computation was developed. We build on this earlier work and open a new vein of inquiry by addressing issues of scalability, partial information, and complex interactions across agent boundaries. Consider for instance a scenario where two agents A_1 and A_2 are negotiating about when A_1 can complete task T_1 that enables A_2's task T_2. This negotiation involves an iterative process of proposals and counter-proposals where at each stage A_2 generates a commitment request to A_1, A_1 performs local optimization computations (scheduling) to evaluate commitment requests; this process repeats until A_1 and A_2 arrive at a mutually acceptable commitment. The multiagent meta-level control decision would be to ensure that A_1 completes its local optimization in an acceptable amount of time so that A_2 can choose alternate methods in case the commitment is not possible. In setting up a negotiation, the meta-level control should establish when negotiation results will be available. This involves defining important parameters of the negotiation including the negotiation context and the earliest time the target method will be enabled. Meta-level control will ensure that the negotiation phase of two agents overlaps to guarantee efficiency. Multiagent meta-level control (MMLC) facilitates agents to have a decentralized meta-level multiagent policy, where the progression of what deliberations the agents should do, and when, is choreographed carefully and includes branches to account for what could happen as deliberation plays out. Our hypothesis in this paper is that MMLC leads to improved performance in the context of a multiagent tornado tracking application.

NetRads [7] is a network of adaptive radars controlled by a collection of Meteorological Command and Control (MCC) agents that determine for the local radars where to scan based on emerging weather conditions. The NetRads radar is designed to quickly detect low-lying meteorological phenomena such as tornadoes, and each radar belongs to exactly one MCC. The MCC agent can manage multiple radars simultaneously. The time allotted to the radar and its control systems for data gathering and analysis of tasks is known as a *heartbeat*. In [7], a system is implemented with three phases containing only deliberative-level actions in a heartbeat. The phases are: *Data Processing*, *Local Optimization (LO)* and *Negotiation (Neg)*. In *Data Processing*, each MCC analyzes weather moment data from the radars collected during the previous heartbeat. The results of this analysis lead to a set of weather-scanning tasks of interest for the next radar scanning cycle. In *LO*, the MCC determines the best set of scans for the available radars that will maximize the sum of the utilities associated with the chosen tasks according to a utility function based on the end-user priorities. In *Neg*, the MCC negotiates with its neighboring MCCs to adjust their local optimization so as to accomplish joint tasks and to avoid redundant scanning of the same area. The authors [7] applied this negotiation protocol to an abstract simulation of NetRads radars to show its usefulness.

In our work, we add a fourth phase, implemented as the *MMLC Module*. Each heartbeat is now split up into four phases containing both deliberative-level actions (Phase 1: *Data Processing* , Phase 3: *LO* and Phase 4: *Neg* are exactly the same as in [7]) and meta-level actions (Phase 2: *MMLC Module*) that is the research proposed in this paper. *MMLC Module* contains meta-level actions that handle the coordination of MCC agents and guide the deliberative-level actions in *LO* and *Neg*. We augment the MCC agents with meta-level control capabilities (Phase 2) to address two problems in NetRads: 1) How to adjust the system heartbeat so as to adapt to changing weather conditions? 2) How to re-organize the sub-nets of radars under each MCC?. We describe a multiagent meta-level control approach that involves coordination of decentralized Markov Decision Processes (DEC-MDPs) [3] using Weighted Policy Learner (WPL) [2], a reinforcement learning (RL) algorithm. WPL is used to learn the policies for the meta-level DEC-MDPs belonging to individual agents. We empirically show that distributed meta-level control gives a performance advantage in NetRads for a number of scenarios.

The rest of the paper is structured as follows: We first identify the meta-level research issues within the context of NetRads, a real-world tornado-tracking application. We then describe the formalization of MMLC based on coordinating DEC-MDPs [3] using WPL algorithm followed by an empirical evaluation of this approach on NetRads. We then present the conclusions and future work directions.

2. Motivating Example

At the highest level, the question we plan to address in NetRads is the following: How does the meta-level control component of each agent learn policies so that it can efficiently support agent interactions with other agents and reorganize the underlying network when needed? Specifically in NetRads, reorganizing the network involves addressing the following questions:

1) What weather conditions trigger a radar to be handed off to another MCC and how do we determine which MCC to hand off the radar to?
2) How to assign different heartbeats to sub networks of agents in order to adapt to changing weather conditions?

The intuition behind identifying these meta-level issues is that it is preferable that radars with large *data correlation* be allocated to the same MCC. *Data Correlation* occurs when radars belonging to different MCCs have overlapping sensing areas and there are weather phenomena in these overlapping areas. MCCs cooperatively avoid redundant scans of the same area by sharing data with each other. Allocating such radars to the same MCC potentially reduces the amount of communication and the time for negotiation among MCCs. Moreover, adjusting the system heartbeat allows MCCs to adapt to changing weather conditions. For example, if rapidly changing weather phenomena occur in a certain region, meta-control may decide to use a shorter heartbeat to allow the system to respond more rapidly. In our work, a single heartbeat of MCC is set to be 30 seconds (shorter) or 60 seconds (longer). This decision would also involve reorganizing the MCC neighborhoods so that there are clusters of MCCs with each cluster having a different heartbeat depending on the type and frequency of tasks that the cluster has to handle.

Fig 1 shows an example NetRads topology of 4 MCCs. Each MCC controls 3 radars (A radar is connected with its supervised MCC via a solid line in Fig 1, e.g., MCC_2 supervises radars $\{R_4, R_5, R_6\}$). Data correlation between two radars is represented by dashed arrows. R_3 has high data correlation with R_4 and R_5, and reallocating it from MCC_1 to MCC_2 will improve performance. In Fig 1, suppose rapidly changing weather phenomena occur in the common boundary between MCC_2 and MCC_3, it is preferable for these two MCCs to use a shorter heartbeat (30 seconds) so as to respond rapidly to the changing environments. Also, suppose MCC_1, MCC_2 and MCC_3 execute the specific actions respectively: " Move R_3 to MCC_2", "Move R_5 to MCC_3" and "Move R_9 to MCC_4" . Fig 2 is the resulting NetRads topology. By making such changes in heartbeat and radar associations, the system reduces the time needed for negotiation among MCCs as well as enhancing the average quality of radar scanning tasks.

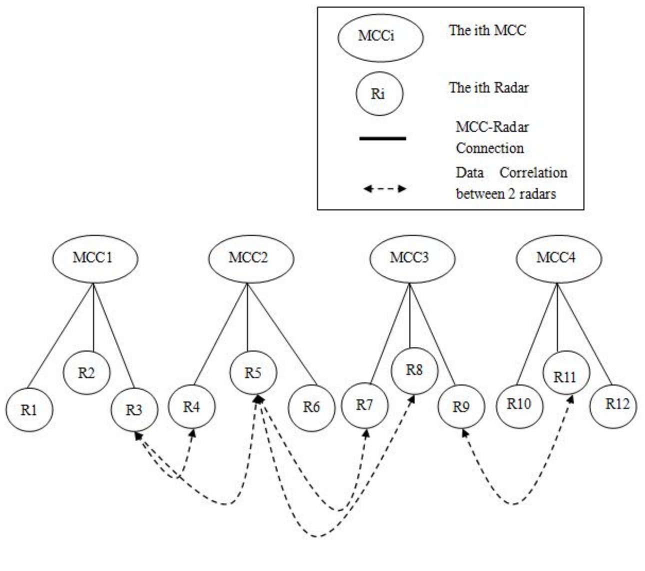


Fig. 1. An Example NetRads Topology

In the next section we describe the details of Phase 2: *MMLC Module* which implements the coordination of meta-level control parameters across agents. This includes discussing the RL based approach to learn meta-level policies and how the MCC network handles the non-stationary environment caused by changing patterns of weather-scanning tasks by

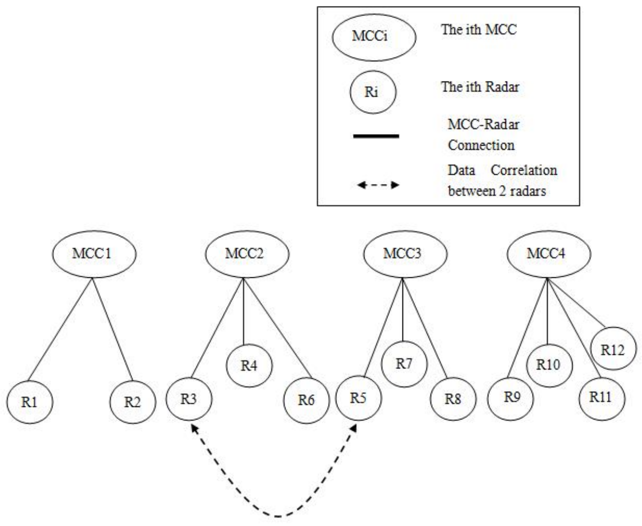


Fig. 2. Resulting NetRads Topology of Fig 1.

switching among policies.

3. Formalizing MMLC

Prior to describing our MMLC framework, we define several key terms used in the rest of this paper:

Task: In NetRads application, each scanning *task* in the system has a position, a velocity, a radius, a priority, a preferred scanning mode, and a type [7]. *Tasks* may be one of a few different types: *storm*, *rotation*, *reflectivity* or *velocity*. Each of these types has its own distributions for the characteristics described above. Tasks may be either *pinpointing* or *non-pinpointing*.

Pinpointing and non-pinpointing Task: *Pinpointing* tasks are those tasks that can not be completed effectively without simultaneous scanning by multiple radars belonging to the same or different MCCs [7]. The utility gained from scanning a pinpointing task increases with the number of radars scanning the task; whereas, the utility for a *non-pinpointing* task is the maximum of the utilities from the individual radars.

Degree of Data Correlation: *Degree of data correlation* reflects the interdependence of the tasks that MCC_i has with those of its neighbor(s). It is defined as $\langle Q_1, Q_2, ..., Q_j \rangle$, in which j is the total number of MCC_i's neighbors and $Q_j \in \{High, Low\}$. When radars belonging to different MCCs share data (especially data about the pinpointing tasks between them), the communication between these two MCCs during negotiation would increase and thus there is more interdependency . Tasks may be either pinpointing or non-pinpointing. We assume the value to be $High$ if the percentage of pinpointing tasks between two MCCs is equal or more than 50%; otherwise it is set to Low.

Neighborhood Scenario: In NetRads application, two MCCs are defined as *neighbors* if they share overlapping scanning regions (In Fig 1, MCC_2 has two neighbors $\{MCC_1$, $MCC_3\}$ while MCC_1 has only MCC_2 as its neighbor.). In other words, if radars belonging to two MCCs are expected to scan some part of the same physical space, then the MCCs are neighbors. Each *neighborhood scenario* is a qualitative abstraction that captures the characteristics of a class of real scenarios that are similar in structure and policy. We define a set of NS_i which consists of the *neighborhood scenarios* of MCC_i might encounter based on the data correlation degrees it has with its neighbors. $NS_i = \{V_0, V_1, .., V_j\}$, where j denotes the number of neighbors of MCC_i. $V_j (j \neq 0)$ denotes the number of radars involved in the data correlation between MCC_i and its jth neighbor (V_0 is the number of radars of MCC_i involved in the data correlation.). $V_j \in \{0, 1, many\}$. In Fig 1, from the view of MCC_2, it is in NS_2= $\{many, 1, many\}$.

3.1. Meta-level Control flow

Fig 3 captures the control flow in the *MMLC Module* of each MCC. The *Scenario Library Module* stores the MDPs of the abstract meta-level scenarios and their policies which is available to each MCC agent. We group sets of MCC scenarios into abstract meta-level scenarios based on types of *tasks* and *neighborhood scenarios* and learn the policies for each abstract scenario offline which is the role of the *Offline RL Module* (We will discuss this module later.). The *Optimal Policy Generation Module* generates the optimal abstract policy from an abstract MDP. The *Specific Action Mapping Module* maps the abstract action policies to specific actions in NetRads domain which includes radar/MCC reconfiguration and heartbeat adaptation. At runtime, each MCC agent adopts the scenario-appropriate policy, executes the appropriate actions and switches to a new policy with changes in scenario in the next heartbeat.

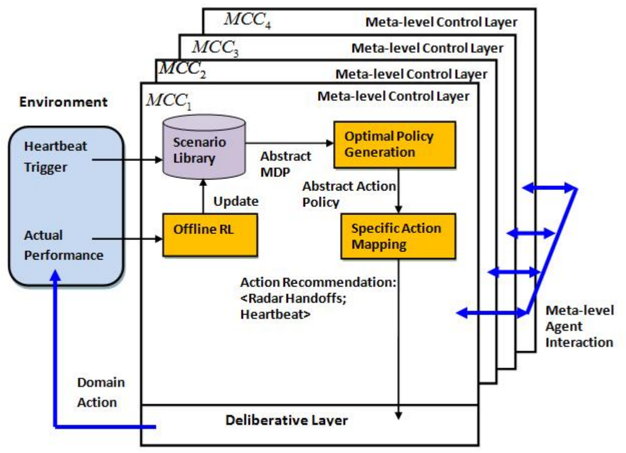


Fig. 3. Control flow in *MMLC Module* of each MCC involving 4 MCCs.

3.2. DEC-MDP formalization

A Markov Decision Process (MDP) is a probabilistic model of a sequential decision problem, where states can be perceived exactly, and the current state and action selected determine a probability distribution on future states [10]. Specifically, the outcome of applying an action to a state depends only on the current action and state (and not on preceding actions or states). We map the NetRads meta-level control problem to a DEC-MDP model in the following way. The model is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where

- $\mathcal{S}$ is a finite set of world states, with a distinguished initial state s^0 which is scenario dependent. s^0 of MCC_i is different in different environments. It depends on the situation of MCC_i, MCC_i's neighbors and the *degree of data correlation* between them. In NetRads domain, the state of each MCC agent is the *meta-level state* (defined below).
- $\mathcal{A}$ is a finite set of actions. In NetRads domain, The actions for the MCC agents are the combinations of the *abstract actions* (defined below) or the changing of the heartbeat.
- $\mathcal{P}$ is a transition function. $\mathcal{P}(s' \mid s, a_i)$ is the probability of the outcome state s' when the action a_i is taken in state s. In NetRads domain, the transition function is based on the time/quality distribution for the actions MCC_i chooses to execute.
- $\mathcal{R}$ is a reward function. $\mathcal{R}(s, a_i, s')$ is the reward obtained from taking action a_i in state s and transitioning to state s'. In NetRads domain, the reward is only received in a terminal state, and it represents the average of *qualities* of all tasks collected by MCC_i in Phase 1 (*Data Processing*) from last heartbeat. The *quality* of a task from a single radar is the priority of the task multiplied by a factor meant to represent the quality of the data that would result from the scan (specified by experts in the field e.g. meteorologists) [7].

The real state of the agent has the detailed information related to the agent's decision making and execution [8]. It accounts for every task which has to be reasoned about by the agent, the execution characteristics of each of these tasks, and information about the environment such as types of tasks (*storm*, *rotation*, *velocity* or *reflectivity* in NetRads application) arriving at the agent and frequency of arrival of tasks. The real state is continuous and complex. This leads to a combinatorial explosion in the real state space for meta-level control even for simple scenarios. The complexity of the real state is handled by defining an abstract representation of the state which captures the important qualitative state information relevant to the meta-level control decision making process. We call this the *meta-level state* of the agent.

We define three features of the *meta-level state* F_0, F_1 and F_2 as follows:

Feature F_0 contains *Information about Self*. Specifically it consists of the MCC's own heartbeat (V_{hb}) and the number of MCC's own radars (V_{radar}) involved in the data correlation with its neighboring MCCs. It is defined as (V_{hb}, V_{radar}), in which $V_{hb} \in \{30seconds, 60seconds\}$ and $V_{radar} \in \{0, 1, many\}$. $many$ means there are more than one radar involved in the data correlation. We use the qualitative value $many$ to simplify the description of MCC's relation with its neighbors so as to reduce the number of different feature sets. As discussed later, this helps determine abstractions of the states and actions of MDPs. In Fig 1, suppose MCC_2 has a 30 seconds heartbeat and it has two radars (R_4 and R_5) involved in the data correlation with its neighboring MCCs. MCC_2 has the feature $F_0 = (30seconds, many)$ in its meta-level state.

Feature F_1 contains *Information about Neighbor(s)*. This feature is expressed as a tuple $\langle f_1, f_2, ..., f_i \rangle$, in which i is the total number of neighbors of the MCC, f_i denotes the ith neighbor's information and is as defined in F_0. In Fig 1, suppose MCC_1 has a 30 seconds heartbeat and MCC_3 has a 60 seconds heartbeat. MCC_2 has the feature $F_1 = \langle (30seconds, 1), (60seconds, many) \rangle$ in its meta-level state.

Feature F_2 has the same definition as *Degree of Data Correlation* defined before.

In Fig 1, MCC_2 has the initial state: s^0, in which $F_0 = (30seconds, many)$, $F_1 = \langle (30seconds, 1), (60seconds, many) \rangle$ and $F_2 = \langle High, High \rangle$.

We abstract the actions in the MDP in two qualitative modes. The two modes are: *Heavy Move* and *Light Move*. Suppose MCC_i has high data correlation with its neighbors, *Heavy Move* of MCC_i , is defined as "Move more than 70% of MCC_i's radars to its neighbors until data correlation degree between MCC_i and its neighbors changes to *Low*" ; *Light Move* of MCC_i is defined as "Move less than 20% radars of MCC_i's radars to its neighbors until data correlation degree between MCC_i and its neighbors changes to *Low*" . *Abstract action* is defined as: $Mode(MCC_i\ to\ MCC_j)$, which means "move radars from MCC_i to MCC_j using the qualitative mode $Mode$. In Fig 1, one action for MCC_2 could be "$LightMove(MCC_1\ to\ MCC_2)\ \&\ LightMove\ (MCC_3\ to\ MCC_2)$". Using abstract actions substantially reduces the number of explored states in the MDP. For example, suppose MCC_i supervises x radars and has y neighbors. Without defining abstract actions, each radar of MCC_i has $y + 1$ possible handoff choices (to be handed off to one of MCC_i's neighbors or stay under MCC_i). The total number of possible action sets for the x radars is $(y+1)^x$ which leads to $(y+1)^x$ exploring states in the MDP tree. Using abstract actions, for each neighbor of MCC_i, MCC_i has 3 possible choices $\{\phi$, *Heavy Move*, *Light Move*$\}$. The total number of possible action sets in this case is 3^y. In NetRads domain, the number of radars each MCC supervises can be large. 3^y is substantially smaller than $(y + 1)^x$ in most cases, especially in the case that x is huge.

3.3. Applying WPL to Learn Policy

Multiagent Reinforcement Learning (*MARL*) is a common approach for solving multiagent decision making problems.

It allows agents to dynamically adapt to changes in the environment, while requiring minimum domain knowledge.

We will use WPL [1], which is a variant of the WoLF [5] algorithm, for learning off-line each agent's meta-level control policy. WPL achieves convergence using an intuitive idea: slow down learning when moving away from a stable policy and speed up learning when moving towards the stable policy. The main idea in Algorithm 1 is to compute an approximate gradient of Q_i, defined as $\Delta(a)$, and use it to update π_i, with small step η. We determine the computation of $\Delta(a)$ by comparing the value of total average reward $\widehat{r}$ to the value of $Q_i(s,a)$. A learner is doing better than expected, if

$$\Sigma_{a\in A}\pi_i(s,a)Q_i(s,a) > Q_i(s,a) \qquad (1)$$

When it is doing better, we update π_i using $\Delta(a)$ calculated in line 9, Algorithm 1, otherwise using $\Delta(a)$ calculated in line 10, Algorithm 1.

In Algorithm 1, $Q_i(s,a)$ stores the reward MCC_i expects if it executes action a at state s. $\pi_i(s,a)$ stores the probability that MCC_i will execute action a at state s. The actions here are abstracted actions and the states are *meta-level states* as defined in Section 3.2. As in WPL, Q and π together capture what a MCC has learned so far. The reward value in our RL algorithm is the average quality of scanning tasks executed by MCC_i in *Data Processing* phase. In the $(i+1)th$ heartbeat period, the radars of MCC_j would do the scanning tasks based on the optimization of ith heartbeat period. At the beginning of the $(i+2)th$ heartbeat period, the *Average Quality* is collected by MCC_j which reflects the effect of the meta-level control policies of MCC_j in the ith heartbeat period. The horizon of the MMLC policies for the NetRads application is two heartbeat periods. We defined this horizon manually after examining the behavior of the NetRads domain in various scenarios. If the horizon of the MMLC policies is too short, it triggers meta-level control too frequently which increases the cost of decision making and affects performance. On the other hand, a long horizon makes the meta-level control policy obsolete due to the dynamic nature of the environment.

Since a heartbeat period consists of four phases, it is important that the *MMLC Module* phase takes negligible amount of time so that there is enough time for the complex operations of *LO* and *Neg* phases. The NetRads system is designed to quickly detect low-lying meteorological phenomena, so time is a critical concern. Online learning on a very large MDP that captures all possible weather scenarios (learning $Q_i(s,a)$ and $\pi_i(s,a)$ for each possible specific weather scenario) during the *MMLC Module* phase can be very time expensive. To overcome this challenge, we construct a library of small MDPs (the *Scenario Library Module*) for different types of *neighborhood scenarios* at the meta-level where there is no requirement for the transfer of learned knowledge between agents. Each *neighborhood scenario* is a qualitative abstraction that captures the characteristics of a class of real scenarios that are similar in structure and policy. We perform the learning offline and constrain the runtime costs by limiting Phase 2 activity to just looking up the scenario-appropriate policy to determine the best action (the *Optimal Policy Generation Module*).

The *Specific Action Mapping Module* maps the abstract action policies to specific actions in NetRads domain which includes radar/MCC reconfiguration and heartbeat adaptation.

Algorithm 1 Abdallah & Lesser's WPL (state s, action a)

```
1:  begin
2:      $r \leftarrow$ Average Quality
3:      update $Q_i(s',a')$ using $r$
4:      $s' \leftarrow s$
5:      $a' \leftarrow a$
6:      $\widehat{r} \leftarrow$ total average reward $= \Sigma_{a\in A}\pi_i(s,a)Q_i(s,a)$.
7:      foreach action $a \in A$ do
8:          $\Delta(a) \leftarrow Q_i(s,a) - \widehat{r}$
9:          if $\Delta(a) > 0$ then $\Delta(a) \leftarrow \Delta(a)(1-\pi_i(a))$
10:         else $\Delta(a) \leftarrow \Delta(a)(\pi_i(a))$
11:     end
12:     $\pi_i \leftarrow \pi_i + \eta\Delta$
13: end
```

In the next section, we will evaluate the role of MMLC in NetRads performance. We first generate meta-level heuristics manually to show meta-level control is useful and then show that our learning algorithm allows the NetRads network to dynamically adjust to changing weather phenomena.

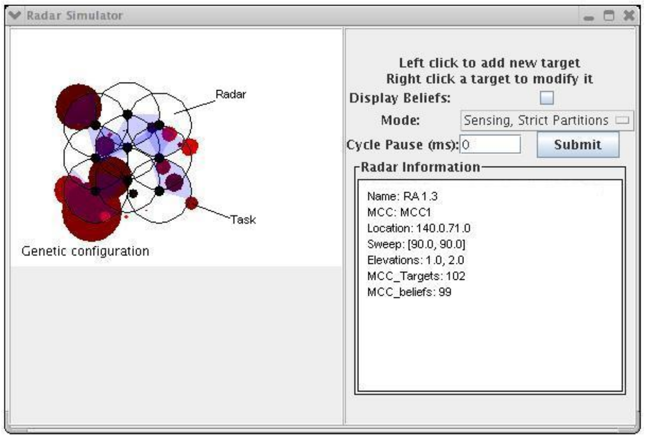
4. Empirical Evaluation

We use the simulator of the NetRads radar system [7] to evaluate our algorithm. In this simulator, radars are clustered based on location, and each cluster of radars has a single MCC. Each MCC has a feature repository where it stores information regarding tasks in its spatial region, where each task represents a weather event. The simulator additionally contains a function that abstractly simulates the mapping from physical events and scans of the radars to what the MCC eventually sees as the result of those scans. MCCs discover and track the movement of the weather events through this process.

Tasks are created at a MCC based on radar moment data that has been just received. Tasks may be either *pinpointing* or *non-pinpointing*.

4.1. Experiment Setup

For the experiments reported here, we use the simulation setup where there are 3, 12 and 30 MCCs (agents). This is the setup used by Krainin et. al [7]. Fig 4 is the snapshot of the radar simulator for a particular real-time scenario. In Fig 4, each hollow circle represents a radar and each filled circle represents a task (we are only concerned about *rotation* and *storm* tasks in the evaluation.). The Radar Information Panel (Fig 4) provides information about a particular radar including its name, its MCC supervisor, its physical location in the plane coordinate system, the angle range it sweeps, the target task it scans and the belief value of the negotiation algorithm in Phase 4: *Neg*. We test the results for three different types of *weather scenarios*. They are defined as: *High Rotation Low Storm* (HRLS), *Low Rotation High Storm* (LRHS), and *Medium Rotation Medium Storm* (MRMS). HRLS denotes the scenario in which the number of rotations overwhelms the number of storms in a series of heartbeats (e.g. Lots of rotation phenomena move in followed by a few storm phenomena, and then followed by lots of rotation phenomena.). LRHS stands for the scenario in which the number of storms overwhelms the number of rotations in a series of heartbeats. MRMS denotes the scenario in which the number of storms approximately equals that of rotations. Suppose there are 80 total tasks, HRLS contained 60 rotation tasks, 20 storm tasks as well as each of the other two types; LRHS contained 60 storm tasks, 20 rotation tasks as well as each of the other two types; MRMS contained 40 storm tasks, 40 rotation tasks as well as each of the other two types.


Radar Simulator
Radar
Task
Genetic configuration
Left click to add new target
Right click a target to modify it
Submit
Display Beliefs:
Mode: Sensing, Strict Partitions
Cycle Pause (ms): 0

Radar Information

Name: RA 1.3
MCC: MCC1
Location: 140.0.71.0
Sweep: [90.0, 90.0]
Elevations: 1.0, 2.0
MCC_Targets: 102
MCC_beliefs: 99

Fig. 4. Snapshot of Radar Simulator.

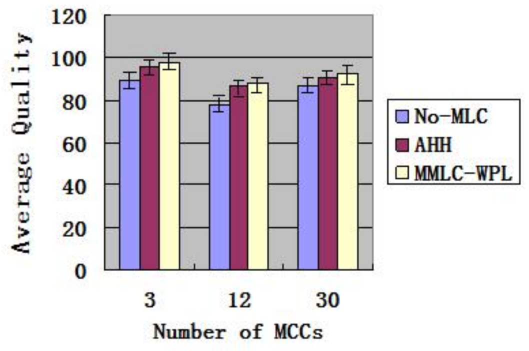
We generate the training/test cases by varying such parameters as number of MCCs, number and types of tasks, initial heartbeat for each MCC, *percentPinpointing* and etc. *percentPinpointing* is defined as the percentage of pinpointing tasks to all tasks in a specific training/test case. We vary *percentPinpointing* to evaluate the performance on different numbers of pinpointing tasks. We also scale up the number of tasks in training/test cases. *Average Quality* (defined in *Data Processing* phase of a heartbeat) and *Negotiation Time* are the parameters to compare the scanning performance. *Negotiation Time* denotes the total time (seconds) MCCs spend in *Neg* (Phase 4).

We compare the results of three methods: *No-MLC*, *Adaptive Heuristic Heartbeat (AHH)* and *MMLC-WPL*. *No-MLC* is the method that without meta-level control module (It has all the phases except *MMLC Module* in a heartbeat). *AHH* is the method where we incorporate simple heuristics in meta-level control to adaptively change the heartbeat of each MCC. The heuristics are simple: For each MCC_i, at the end of *Data Processing* (Phase 1), if there are more rotation phenomena in the region of MCC_i, MCC_i sets the longer heartbeat for its next heartbeat period, otherwise, MCC_i sets the shorter heartbeat for its next heartbeat period (longer heartbeat is better for rotations due to the need for more scanned elevations, and shorter heartbeat is better for storms). The heuristics also help to address the radar handoff issues. Assigning the same heartbeat to the neighboring $MCCs$ with overlapping region results in better communication/negotiation in the *Neg* phase so as to help reducing the amount of data correlation in the next heartbeat period which has some of the same effect as handing off radars. *MMLC-WPL* augments MCCs with meta-level control based on offline RL (WPL) to adjust the system heartbeat and re-organize the sub-nets of radars so as to adapt to changing weather conditions. For the *MMLC Module* phase, we used 50 training cases and each has a long sequence of training data (500 heartbeat periods) to learn the policies for all the abstract scenarios offline. The learning rate (η in Algorithm 1) is set to 0.01. Using each method mentioned above, we ran 30 test cases for each of three weather scenarios.

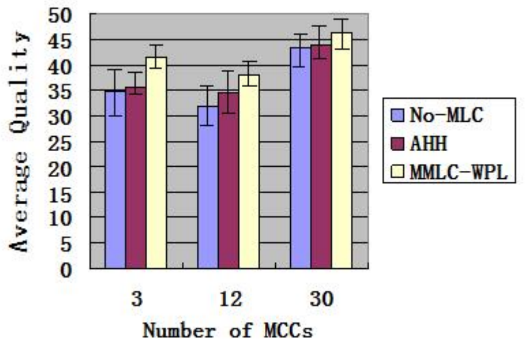
4.2. Discussion

We ran test cases for each *weather scenario* with the number of MCCs to be 3, 12 and 30 (*percentPinpointing* is set to 60%, the number of tasks is 80.). Fig 5 shows the performance of *No-MLC*, *AHH* and *MMLC-WPL* on *Average Quality* for a variety of scenarios. In HRLS scenarios, all the MCCs have to handle HRLS scenarios simultaneously. *AHH* performs significantly ($p < 0.05$) better than *No-MLC* on *Average Quality* in all comparisons (Fig 5(a)). This shows the effectiveness of adding meta-level control in the system in HRLS scenarios. According to the simple rules in *AHH*, the three MCCs would all set their heartbeat to 60 seconds for HRLS. The three MCCs would then have more time to spend on *LO* and *Neg* phases so that the final configurations of scanning tasks for the next heartbeat period would be more optimized. This results in larger *Average Quality*. In HRLS scenarios, *MMLC-WPL* performs significantly (p values from t-tests are 0.0032, 0.024 and 0.0076 respectively) better than *No-MLC* and a little better than *AHH*. The minor discrepancy of performance between *MMLC-WPL* and *AHH* on HRLS scenarios leads to the speculation that the 60 seconds heartbeat is critical for rotations due to the need for more scanned elevations. Rotations need more time for scanning as they must be scanned at the lowest six elevations. Storms, on the other hand, must be scanned at the lowest four elevations to obtain useful information.

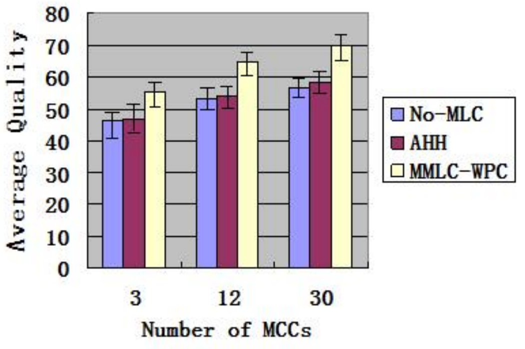
In both LRHS and MRMS scenarios (Fig 5(b) and Fig 5(c)), *AHH* performs a little better than *No-MLC*. *MMLC-WPL* performs significantly better than *No-MLC* (In LRHS scenarios, p values are 0.0085, 0.0048 and 0.015 respectively; In MRMS scenarios, p values are 0.027, 0.0094 and 0.032 respectively) and *AHH* (In LRHS scenarios, p values are 0.014, 0.0051 and 0.00074 respectively; In MRMS scenarios, p values are 0.0062, 0.039 and 0.04 respectively). We can see that the 30 seconds heartbeat is not a profound factor in LRHS



(a) **HRLS Scenarios**



(b) **LRHS Scenarios**



(c) **MRMS Scenarios**

Fig. 5. *Average Quality* of *No-MLC*, *AHH* and *MMLC-WPL* in different weather scenarios for number of MCCs to be 3, 12 and 30.

scenarios (*AHH* increases small amount of *Average Quality*.). In *MMLC-WPL*, each MCC adopts the policy appropriate to its *neighborhood scenario*. Allocating radars with large data correlation to the same MCC reduces the time for negotiation between MCCs which would increase the time for *LO*. In certain situations (e.g., there are many internal tasks compared to boundary tasks) it is better to do a good job in local optimization and allocate fewer cycles to negotiation while in other situations more cycles for negotiation would be better (e.g., many pinpointing tasks exist in boundary regions between MCCs). *MMLC-WPL* performs significantly better on learning policies so as to control when and which radars should be moved.

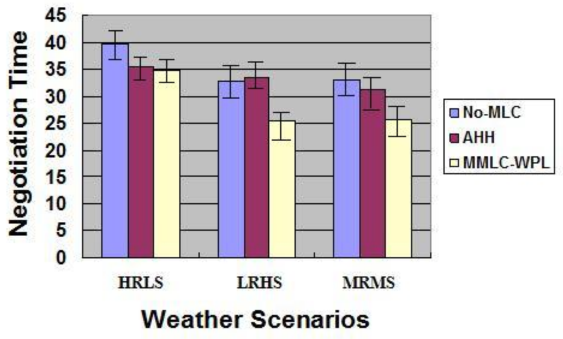


Fig. 6. *Negotiation Time* of *No-MLC*, *AHH* and *MMLC-WPL* with 3 MCCs in different weather scenarios.

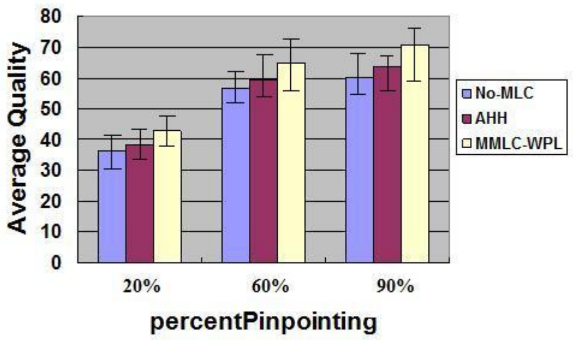


Fig. 7. *Average Quality* of *No-MLC*, *AHH* and *MMLC-WPL* with 3 MCCs, for *percentPinpointing* to be 20%, 60% and 90%.

In Fig 6, *MMLC-WPL* performs significantly better than *No-MLC* on *Negotiation Time* (p values are 0.041, 0.029 and 0.0071 respectively) for each *weather scenario*. *MMLC-WPL* uses least time on *Neg* phase and achieves highest *Average Quality* in each *weather scenario*. This shows that adaptive meta-level control allows for effective use of the heartbeat i.e. by ensuring that meta-level control parameters are coordinated so that negotiations converge quickly, more time can be spent on data processing. *AHH* does not perform better than *No-MLC* on all *weather scenarios* (It spends more *Negotiation Time* than *No-MLC* in LRHS scenarios) since *AHH* is not as adaptive as *MMLC-WPL* in dynamic conditions.

We varied *percentPinpointing* (setting it to 20%, 60% and 90%) and ran test cases with 3 MCCs on all the three *weather scenarios*. In Fig 7, we see that *Average Quality* increases with the increase of the percentage of pinpointing tasks to all tasks for *No-MLC*, *AHH* and *MMLC-WPL*. More pinpointing

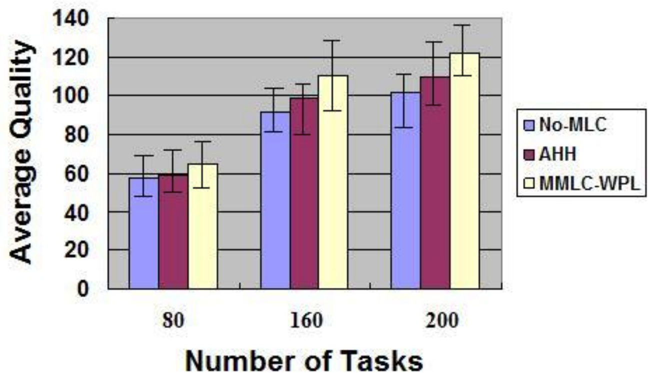


Fig. 8. *Average Quality* of *No-MLC*, *AHH* and *MMLC-WPL* with 3 MCCs, for number of tasks to be 80, 160 and 200.

tasks occurring in the boundary regions between MCCs would increase the utilities for scanning pinpointing tasks so as to increase *Average Quality* of all the scanning tasks. In all *percentPinpointing* settings (20%, 60% and 90%), *AHH* performs better than *No-MLC* and *MMLC-WPL* achieves the best performance.

In Fig 8, we scaled up the number of total tasks to 160 and 200 in the 3-MCC setup and compared the performance with that of 80 tasks (*percentPinpointing* is fixed at 60%). *Average Quality* increases substantially with the increase of number of tasks for all three methods. *MMLC-WPL* performs significantly better than *No-MLC* (p values are 0.038, 0.014 and 0.00043 respectively) and *AHH* (p values are 0.029, 0.0033 and 0.005 respectively) on *Average Quality*.

5. Conclusion and Future Work

In this paper, we describe a Multiagent meta-level control (MMLC) module that coordinates DEC-MDPs at the meta-level and implements a RL-based algorithm to learn the policies of the individual MDPs. Previous work in the NetRads domain [7] showed that a decentralized technique at the deliberation-level with a low number of required optimizations improved tasked allocation in the time-constrained domain. Our hypothesis in this paper is that MMLC that reasons about the deliberative-level approach and coordinates the deliberation across agents leads to improvement in performance.

MMLC equips each agent to carefully choreograph the progression of what deliberations agents should do and when. It also makes agents account for what could happen as deliberation plays out. In our approach, policies for abstract meta-level scenarios are learned offline and each agent adopts the policy appropriate to its scenario at runtime. Empirical evaluation shows that multiagent meta-level control is an efficient way as the problem scales (up to 30 agents) to allocate resources and reorganize the network with the goal of improving performance in the context of a multiagent tornado tracking application. Our model can be applied to other domains such as meeting scheduling and sensor networks where two agents with different views of policies for negotiation need to be reconciled.

Conflicting agent meta-actions can arise based on our use of our learned DEC-MDP policies. We plan on extending our current approach to introduce a dynamic coordination of agent meta-actions to reduce the occurrence of conflicting meta-action choices.

6. Acknowledgements

This work is supported partially by the Engineering Research Centers Program of the National Science Foundation under NSF Co-operative Agreement No.EEC-0313747. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

References

[1] S. Abdallah and V. Lesser. Learning the Task Allocation Game. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 850–857, Hakodate, Japan, 2006. ACM Press.

[2] S. Abdallah and V. Lesser. Multiagent Reinforcement Learning and Self-Organization in a Network of Agents. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 172–179, Honolulu, May 2007. IFAAMAS.

[3] D. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence(UAI)*, pages 32–37, 2000.

[4] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[5] M. Bowling and M. Veloso. Scalable Learning in Stochastic Games. In *Proceedings of AAAI 2002 Workshop on Game Theoretic and Decision Theoretic Agents*, July 2002.

[6] M. Cox and A. Raja. Metareasoning: A Manifesto. In *Proceedings of AAAI 2008 Workshop on Metareasoning: Thinking about Thinking*, pages 1–4, Chicago,IL, July 2008.

[7] M. Krainin, B. An, and V. Lesser. An Application of Automated Negotiation to Distributed Task Allocation. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, pages 138–145, Fremont, California, November 2007. IEEE Computer Society Press.

[8] A. Raja and V. Lesser. A Framework for Meta-level Control in Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems*, 15(2):147–196, October 2007.

[9] S. J. Russell and P. Norvig. *Artificial Intelligence A Modern Approach*. Pearson Education, 2006.

[10] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, 1998.