# MAGE: ADDITIONS TO THE AGE ALGORITHM FOR LEARNING IN MULTI-AGENT SYSTEMS

Michael L. Dowell and Dr. Larry M. Stephens

Department of Electrical & Computer Engineering
University of South Carolina
Columbia, South Carolina 29208, USA
e-mail: Dowell@ece.scarolina.edu
Stephens@ece.scarolina.edu

## ABSTRACT

This paper introduces the Modeling Action Group Estimation algorithm (MAGE) which improves the Action Group Estimation (AGE) algorithms developed by Weiss[4]. These algorithms allow groups of agents to learn to coordinate their actions by adjusting estimates for individual actions and groups of actions.

## 1 INTRODUCTION

The performance of Multi-Agent systems is affected by how agents coordinate their actions to achieve their goals. The **Ac**tion **E**stimation (ACE) and **A**ction **G**roup **E**stimation (AGE) algorithms developed by Weiss [4] address this concern by providing a method for agents to learn to coordinate their actions. These algorithms use reinforcement learning methods for multi-agent systems based on an action-oriented version [2,3] of Holland's bucket brigade learning model for classifier systems [1].

## 2 FORMAL SPECIFICATION

The set of all agents is denoted by $AG = \{ag_1, \ldots, ag_n\}$ where $n \in \mathbf{N}$. The environment is represented by environmental states *S, T, U, ...* where $S = \{s_1, \ldots, s_m\}$. An agent can sense all or part of an environmental state with $S_i$ representing the part of the environmental state *S* that agent $ag_i$ senses where $ag_i \in AG$ and $S_i \subseteq S$. An agent may not be able to distinguish between two different environmental states. For example, given

two different environmental states $S$ and $T$, $S_i = T_i$ is possible if agent $ag_i$ cannot sense the difference between these environmental states. In addition, the combination of all agents sensing information may not be complete since $\bigcup_{i=1}^{n} S_i = S$ is not required.

The allowed actions agent $ag_i$ can carry out is the *action potential* of $ag_i$ and is denoted as $AP_i$ where $AP_i = \{a_i^1, ..., a_i^{p_i}\}$ with $p_i$ equaling the number of allowed actions for agent $ag_i$. An agent may not be able to carry out all of its actions within its *action potential* for a particular environmental state. The set of actions an agent $ag_i$ can carry out in environmental state S is represented as its set of *possible actions*, $A_i(S)$, where $A_i(S) \subseteq AP_i$ with the number of actions being less than or equal to the number of allowed actions.

### 2.1 The ACE algorithm

The **Ac**tion **E**stimation algorithm (ACE) allows each agent to learn to estimate the goal relevance of its actions. ACE consists of a cycle of the following five steps:

1. Each agent $ag_i$ senses $S_i$ from the current environmental state $S$ where $S_i \subseteq S$.

2. Each agent $ag_i$ determines its set of *possible actions*, $A_i(S)$.

3. Each agent $ag_i$ computes a bid $B_i^j(S)$ for each of its actions $a_i^j$ in $A_i(S)$ using the following formula:

$$B_i^j(S) = \begin{cases} (\alpha + \beta) \bullet E_i^j(S) & : \quad E_i^j(S) > \theta, \\ 0 & : \quad otherwise \end{cases} \qquad (1)$$

   An agent's estimate of the goal relevance for action $a_i^j$ is represented as $E_i^j(S)$ where $a_i^j \in A_i(S)$. The agent is willing to risk a fraction of $E_i^j(S)$ for performing the action. A small constant, $\alpha$, the *risk factor*, represents this fraction. The *noise term*, $\beta$, is a small random number used to introduce noise to avoid getting stuck in a local learning minima while cycling through the algorithm. From Weiss' paper, its appears the *risk factor* combined with the *noise term* is in the range [0,1]. To prevent useless actions from executing, an the *estimate minimum*, $\theta$, is introduced.

4. Once the agents have completed their bids, each agent announces its bid for each action $a_i^j$ in its set of *possible actions*, $A_i(S)$.

5. Once the agents have announced their bids, the actions to be executed are selected and placed in the *activity context* represented by the set $C(S)$. To accomplish this task, all actions of every agent are collected into a *context potential* denoted by $CP(S)$ where $CP(S) = \bigcup_{i=1}^{n} A_i(S)$ and the *activity context* is initially empty, $C(S) = \varnothing$. Until $CP(S)$ is empty, the action with the highest bid is removed from $CP(S)$ and added to $C(S)$. After an action is selected, any actions incompatible with it left in the $CP(S)$ are removed so that these actions will not be selected later. This means $C(S) \subseteq CP(S)$. This procedure is formally defined by the following steps:

   i) $CP(S) = \bigcup_{i=1}^{n} A_i(S)$ and $C(S) = \varnothing$

   ii) until $CP(S) = \varnothing$ do
   - select $a_l^j \in CP(S)$ with $B_l^j(S) \geq B_k^l(S)$ for all $a_k^l \in CP(S)$
   - $C(S) = C(S) \cup \{a_l^j\}$
   - $CP(S) = CP(S) - (\{a_l^j\} \cup \{a_k^l \in CP(S) : a_k^l$ and $a_l^j$ are incompatible$\})$

6. The final step allows each agent to learn by adjusting its estimates $E_l^j(S)$ for each selected action $a_l^j$ in $C(S)$. It is important to realize only the estimates for selected actions are adjusted. These estimates are increased by two different components:
   - any external reward, $R$, is evenly distributed,
   - the sum of the bids for all actions in the activity context is evenly distributed.

   Each selected action receives an equal increase based upon this method. In addition, each estimate is decreased by the amount of its bid. Formally, this is defined in the following formula where $E_l^j(S)$ is $a_l^j$'s estimate and $B_l^j(S)$ is $a_l^j$'s bid. The other two terms equal the same amount for each $a_l^j$ within the *activity context*.

$$E_i^j(S) = E_i^j(S) - B_i^j(S) + \frac{R}{|C(S)|} + \frac{\sum_{a_i^j \in C} B_i^j(S)}{|C(S)|} \qquad (2)$$

   The external reward, $R$, and the sum of the individual bids is divided by the number of actions in the *activity context*, $|C(S)|$. This procedure does not require each agent to perform an action.

## 2.2 The AGE algorithm

The **A**ction **G**roup **E**stimation (AGE) algorithm has the same basic cycle as the ACE algorithm with one important difference. An agent's action becomes part of an *activity context* and is not considered independently. Therefore, the bids are not for individual actions but are for groups of actions. While an agent's estimate of goal relevance for an action is still based upon knowledge of the environmental state $S_i$, it now depends upon the *activity context C*. Therefore, the estimate of the goal relevance of an action $a_i^j$ includes the additional *activity context* term and is represented as $E_i^j(C)$. With this addition, bids for each action now depend upon actions of the other agents; and, thus, bids are for an *activity context* and not for a single action.

To determine the *activity context* with the highest bid, all *activity contexts* must be generated. The set of *activity contexts* for environmental state *S* is represented by the *possible activity context*, *PAC*(*S*), where *PAC*(*S*) = $\{C_1(S), C_2(S), ..., C_r(S)\}$ with *r* equaling the number of *activity contexts*. Generating a *PAC*(*S*) requires each agent to announce its every action $a_i^j$ where $a_i^j \in A_i(S)$. The *PAC*(*S*) is the cross product of each agent's set of *possible actions*. Any *activity contexts* with incompatible pairs of actions are not added to *PAC*(*S*). Formally, this set is determined by the following:

$$PAC(S) = \{A_1 \times A_2 \ldots \times A_n : (\forall a_k^l, a_p^q \in C(S): a_k^l \text{ and } a_p^q \text{ are compatible})\} \qquad (3)$$

This modification changes the third step of the ACE algorithm. Each agent $ag_i$ computes a bid $B_i^j(C_k(S))$ for each of its actions $a_i^j$ in every *activity context* $C_k(S)$ in its *PAC*(*S*). Each bid is calculated as follows:

$$B_i^j(C_k(S)) = \begin{cases} (\alpha + \beta) \bullet E_i^j(C_k(S)) & : \quad E_i^j(C_k(S)) > \theta, \\ 0 & : \quad \text{otherwise} \end{cases} \qquad (4)$$

where $E_i^j(C_k(S))$ is $ag_i$'s estimate of the goal relevance of action $a_i^j$ based upon *S* and $C_k(S)$. The composite bid $B_k(C_k(S))$ for an *activity context*, $C_k(S)$, is the sum of bids for each action in $C_k(S)$ and is defined as follows:

$$B_k(C_k(S)) = \sum_{a_i^j \in C(S)} B_i^j(C_k(S)) \qquad (5)$$

The fifth step is simplified to select the *activity context* in the *PAC*(*S*) with the highest bid and all the actions in the *activity context* are executed.

The final step changes such that each bid and estimate includes the additional *activity context* term and are represented as $B_i^j(C(S))$and $E_i^j(C(S))$. The estimates are adjusted according to the same method:

$$E_i^j(C(\mathrm{S})) = E_i^j(C(\mathrm{S})) - B_i^j(C(\mathrm{S})) + \frac{R}{|C(\mathrm{S})|} + \frac{B_C(S)}{|C(\mathrm{S})|} \qquad (6)$$

## 3 THE MAGE ALGORITHM

A major drawback of the AGE algorithm is the overhead for generating large numbers of *activity contexts*. Part of this overhead is in communication costs. Initially, each agent $ag_i$ announces its every action $a_i^j$ in $A_i(\mathrm{S})$. After the bids are computed, each agent announces its bids. In the ACE algorithm, there is one bid for each action, but in the AGE algorithm, each agent has one bid for each *activity context* $C(S)$ in its $PAC(S)$.

The number of *activity contexts* $C(S)$ increases rapidly. This number, $|PAC(S)|$, is the product of sums of each agent's actions and is bounded by:

$$|PAC(S)| \leq \prod_{i=1}^{n} |A_i(S)| \qquad (7)$$

The large number of *activity contexts* leads not only to a large number of bid announcements from each agent but to considerable computing to determine the bids since there is overhead for each *activity context* in this cycle. Each agent generates every *activity context* and tests each for validity. Each valid *activity context* has a bid $B_i^j(C(S))$ generated by each agent. These bids, when combined, produce a composite bid $B_k(C_k(S))$. Finally, the *activity context* with the highest composite bid is selected from $PAC(S)$.

For example, three agents, each with three actions, will have twenty-seven *activity contexts* to be considered. Not all of these will be valid, but each is generated to determine if it is valid. If an environment, such as the blocks world, allows only a few valid *activity contexts,* the overhead will be less since only valid *activity contexts* are used in the rest of the cycle. Each agent announces each of its three actions for a total of nine action announcements. After these announcements, each agent generates and tests all twenty-seven *activity contexts*. Only one of these twenty-seven is selected. The overhead for the twenty-six *activity contexts* not selected is wasted. In addition, only three of the nine action announcements are required since an agent executes only one action at a time.

To improve the AGE algorithm, the number of *activity contexts* considered must be kept to a minimum. This is accomplished by allowing each agent to model the other agents. Providing the ability to predict the probable actions of other agents enables an

agent to consider only the best *activity contexts* and thus compute bids for fewer *activity contexts*. This method avoids the overhead for *activity contexts* not selected.

The modeling information is represented by a *model structure*. Each agent has a *model structure* that contains estimates of the other agent's likely actions for each environmental state. The *model structure* is defined as a set of tuples given by:

$$M_i = \begin{Bmatrix} \langle S_i, ag_j, a_j^k, ME[S_i, a_j^k] \rangle, \ldots, \langle S_i, ag_m, a_m^n, ME[S_i, a_m^n] \rangle \\ \ldots, \\ \langle V_i, ag_v, a_v^w, ME[V_i, a_v^w] \rangle, \ldots, \langle V_i, ag_x, a_x^y, ME[V_i, a_x^y] \rangle \end{Bmatrix} \quad (8)$$

where $M_i$ is the *model structure* for agent $ag_i$, and $S_i$ to $V_i$ are the environmental states sensed by $ag_i$ during problem-solving. Each tuple contains an agent $ag_j$, its action $a_j^k$, and its *modeling estimate*, $ME(S_i,a_j^k)$ where $a_j^k$ is the *k*th action of agent $ag_j$. The *modeling estimate* is an estimate of another agent's likelihood of taking a specific action based upon the environmental state $S_i$. Initially, each agent's *model structure* is empty but as new environmental states are encountered, these are added to an agent's *model structure* by observing the actions of the other agents. The addition of the modeling aspect to the AGE algorithm is termed the **M**odeling **A**ction **G**roup **E**stimation (MAGE) algorithm. The MAGE algorithm consists of a cycle of the following steps:

1. Each agent $ag_i$ senses $S_i$ from the current environmental state *S* where $S_i \subseteq S$.

2. Each agent $ag_i$ determines its set of *possible actions*, $A_i(S)$.

3. Using the information in $M_i$, each agent $ag_i$ estimates the actions other agents select for the current environmental state *S*. Each agent collects these actions in a set of *modeled actions*, $MA_i(S)$, where $MA_i(S) = \{a_j^1, \ldots, a_j^{uj}\}$ with $u_j \leq q_j$ with $q_j$ equaling the number of actions for agent $ag_j$ in *S*.

4. Each agent's set of *modeled actions* are used to generate a *modeled context potential*, denoted by *MCP*(*S*) where $MCP(S) = \bigcup_{i=1}^{n} MA_i(S)$. The number of actions within *MCP*(*S*) is less than or equal to the number of actions within the *context potential*, *CP*(*S*). This is important because this number determines how many *activity contexts* are generated for the bidding process.

5. The actions in the *MCP*(*S*) are used to generate the *modeled possible activity context*, *MPAC*(*S*), where $MPAC(S) = \{C_1(S), C_2(S), \ldots, C_t(S)\}$. Comparing *t*, the

number of activity contexts within *MPAC*(*S*), with *r*, the number of *activity contexts* in *PAC*(*S*), shows that $t \leq r$. Formally, the *modeled possible activity context* is given by:

$$MPAC(S) = \{MPAC(S) \subseteq MCP(S) : (\forall a_k^l, a_p^q \in C(S) : a_k^l \text{ and } a_p^q \text{ are compatible})\} \quad (9)$$

There can be one or more *activity contexts* in an agent's *MPAC*(*S*) since the agent may not be able to distinguish between different environmental states such as *T* and *V*. In this event, an agent $ag_i$ would only sense a single environmental state $T_i$ from *T* and *V*. Another agent $ag_j$ may take different actions for these different environmental states and thus the modeling estimates of $ag_i$ for $ag_j$ would predict two actions for environmental state $T_i$.

6. The final step allows agents to learn by adjusting *modeling estimates*, $ME(S_i,a_j^k)$, for each of the other agent's actions If the current environmental state has not been encountered, a new set of tuples is added to an agent's *modeling structure*. The set has the following form:

$$\left\{\left\langle S_i, ag_j, a_j^k, ME\left[S_i, a_j^k\right]\right\rangle, \ldots, \left\langle S_i, ag_m, a_m^n, ME\left[S_i, a_m^n\right]\right\rangle\right\} \quad (10)$$

A tuple for each of the other agents is added to represent its selected action. The tuple has the following form:

$$\left\langle S_i, ag_j, a_j^k, ME\left[S_i, a_j^k\right]\right\rangle \quad (11)$$

where $a_j^k$ is the action taken by agent $ag_j$, and $ME(S_j,a_j^k)$ is the *modeling estimate* for this action. If $S_j$ has been encountered, existing *modeling estimates* are adjusted.

## 4 MAGE EXAMPLES

The task domain used for the examples is the blocks world. This is the same task domain chosen by Weiss [28] and will allow for a direct comparison of the AGE and MAGE algorithms. The problem is to transform an initial configuration of blocks into a goal configuration. The goal configuration is shown in Figure 1.

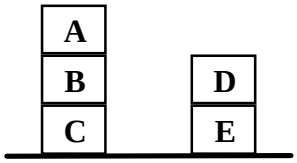


Figure 1: The Goal Configuration

Each agent has limited motor capabilities and is responsible for moving one block. For example, agent $ag_1$ is responsible for moving block **A**, agent $ag_2$ is responsible for moving block **B**, etc. In addition, agent $ag_1$ has only two allowed actions: moving block **A** on top of block **B**, represented by *put*(**A**,**B**), or moving block **A** on the table, *put*(**A**,⊥). A precondition for executing a *put* action is the block to be moved must be clear and the block, if any, used as a support is also clear. A block is clear when there are no other blocks on top of it. Each agent's responsibilities and allowed actions are shown in Table 1.

| **Agent** | **Block Assignment** | **Allowed Actions** |
|---|---|---|
| $ag_1$ | **A** | *put*(**A**,**B**), *put*(**A**,⊥) |
| $ag_2$ | **B** | *put*(**B**,**C**), *put*(**B**,**D**) |
| $ag_3$ | **C** | *put*(**C**,**D**), *put*(**C**,⊥) |
| $ag_4$ | **D** | *put*(**D**,**C**), *put*(**D**,**E**) |
| $ag_5$ | **E** | *put*(**E**,**B**), *put*(**E**,**C**), *put*(**E**,⊥) |

Table 1: Assignments and Possible Actions

Each agent also has limited sensing capabilities. An agent senses only information relevant to its actions and, thus, has only local information about the environment. Each agent senses information about blocks that are part of its actions. For example, agent $ag_1$ senses information about blocks **A** and **B** since these are the only blocks that are part of its actions. The information gathered about an assigned block is whether the block's top is clear and the block's support, i.e., what the block is positioned above. More formally, each agent, for each of its actions *put*(*x*,*y*), senses information about *z* where $z \in \{x,y\}$. The information sensed is whether *z* is clear and the block on which *z* is positioned. Each agent's sensing responsibilities are shown in Table 2.

| Agent | Blocks Sensed |
|---|---|
| $ag_1$ | **A, B** |
| $ag_2$ | **B, C, D** |
| $ag_3$ | **C, D** |
| $ag_4$ | **C, D, E** |
| $ag_5$ | **B, C, E** |

Table 2: Agent Sensing Capabilities

Initially, the *model structure*, $M_i$, of each agent $ag_i$ is empty. As problems are solved, each new environmental state is added to each agent's *model structure*. The first example starts with the blocks in the configuration shown in Figure 2.

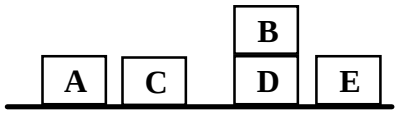


Figure 2: Configuration for Example 1

Each agent's actions and its environmental state sensed are shown in Table 3. An agent's possible actions are the actions an agent can execute in a given environmental state. The **On** column represents the support for each agent's blocks while the **Clear** column represents whether the top of each agent's block is clear. In the **Possible Actions** column, a Null action is also represented since an agent can chose to do nothing for a given environmental state. The information shown in Table 3 is the result of executing steps 1 and 2 of the MAGE algorithm.

| Agent | Possible Actions | Env. State | On | Clear |
|---|---|---|---|---|
| $ag_1$ | *put*(**A**,**B**), Null | $S_1$ | A⊥, BD | A, B |
| $ag_2$ | *put*(**B**,**C**), Null | $S_2$ | BD, C⊥, D⊥ | B, C |
| $ag_3$ | Null | $S_3$ | C⊥, D⊥ | C |
| $ag_4$ | Null | $S_4$ | C⊥, D⊥, E⊥ | C, E |
| $ag_5$ | *put*(**E**,**B**), *put*(**E**,**C**), Null | $S_5$ | BD, C⊥, E⊥ | B, C, E |

Table 3: Environment States and Possible Actions for Example 1

Since no modeling information is initially in the *model structures*, all compatible *activity contexts* are generated and added to the set of *possible activity contexts*, *PAC*(*S*).

Each agent $ag_i$ computes a bid $B_i^j(C_k(S))$ for each *activity context*, $C_k(S)$, in its $PAC(S)$. A $PAC(S)$, without Null actions, is shown below.

$$PAC(S) = \{ \{put(\mathbf{A},\mathbf{B})\}, \{put(\mathbf{B},\mathbf{C})\}, \{put(\mathbf{E},\mathbf{B})\}, \{put(\mathbf{E},\mathbf{C})\}, \{put(\mathbf{A},\mathbf{B}), put(\mathbf{E},\mathbf{C})\}\}$$

Each agent computes five bids, one for each $C_k(S)$ in the $PAC(S)$. The *activity context* selected for execution will be $C_2(S)$, $\{put(\mathbf{B},\mathbf{C})\}$, since this action will transform the configuration into a configuration that is closer to the goal configuration.

Since this is the first problem to be solved, each agent's *model structure* is empty. Step 6 of the algorithm will add the current environmental state, $S_i$, to each agent's *model structure*. Each agent's *model structure*, after sensing the environment and selecting the action $put(\mathbf{B},\mathbf{C})$, will be:

$M_1$={⟨$S_1$, $ag_2$, $put(\mathbf{B},\mathbf{C})$,1⟩,⟨$S_1$, $ag_3$, Null, 1⟩, ⟨$S_1$, $ag_4$, Null, 1⟩, ⟨$S_1$, $ag_5$, Null, 1⟩}
$M_2$={⟨$S_2$, $ag_1$, Null, 1⟩,⟨$S_2$, $ag_3$, Null, 1⟩, ⟨$S_2$, $ag_4$, Null, 1⟩, ⟨$S_2$, $ag_5$, Null, 1⟩}
$M_3$={⟨$S_3$, $ag_1$, Null, 1⟩,⟨$S_3$, $ag_2$, $put(\mathbf{B},\mathbf{C})$,1⟩, ⟨$S_3$, $ag_4$, Null, 1⟩, ⟨$S_3$, $ag_5$, Null, 1⟩}
$M_4$={⟨$S_4$, $ag_1$, Null, 1⟩,⟨$S_4$, $ag_2$, $put(\mathbf{B},\mathbf{C})$,1⟩, ⟨$S_4$, $ag_3$, Null, 1⟩, ⟨$S_4$, $ag_5$, Null, 1⟩}
$M_5$={⟨$S_5$, $ag_1$, Null, 1⟩,⟨$S_5$, $ag_2$, $put(\mathbf{B},\mathbf{C})$,1⟩, ⟨$S_5$, $ag_3$, Null, 1⟩, ⟨$S_5$, $ag_4$, Null, 1⟩}

where $S_i$ is the environmental state sensed by agent $ag_i$. After agent $ag_2$ executes the $put(\mathbf{B},\mathbf{C})$ action, the block's configuration is as shown in Figure 3.

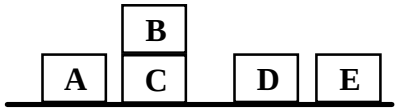


Figure 3: Second Configuration for Example 1

Given the second configuration, each agent's actions and environmental state sensed are shown in Table 4.

| Agent | Possible Actions | Env. State | On | Clear |
|---|---|---|---|---|
| $ag_1$ | $put(\mathbf{A},\mathbf{B})$, Null | $T_1$ | A⊥, BC | A, B |
| $ag_2$ | $put(\mathbf{B},\mathbf{D})$, Null | $T_2$ | BC, C⊥, D⊥ | B, D |
| $ag_3$ | Null | $T_3$ | C⊥, D⊥ | D |
| $ag_4$ | $put(\mathbf{D},\mathbf{E})$, Null | $T_4$ | C⊥, D⊥, E⊥ | D, E |
| $ag_5$ | $put(\mathbf{E},\mathbf{B})$, Null | $T_5$ | BC, C⊥, E⊥ | B, E |

Table 4: Environment States and Actions for Second Configuration of Example 1

A search of each agent's *modeling structure* results in no matches, therefore, all compatible *activity contexts* are generated and added to a reinitialized *possible activity context*, *PAC*(*S*). Each agent then computes a bid as shown before. The *PAC*(*S*), without Null actions, for the second configuration is shown below.

$PAC(S)$ = { {*put*(**A**,**B**)}, {*put*(**B**,**D**)} {*put*(**D**,**E**)}, {*put*(**E**,**B**)}, {*put*(**A**,**B**), *put*(**D**,**E**)} }

The number of *activity contexts* in a *PAC*(*S*) is again five, and each agent computes and broadcasts five bids. The *activity context* selected will be $C_5(S)$, {*put*(**A**,**B**), *put*(**D**,**E**)}, since this action transforms the configuration into the goal configuration.

Step 6 adds $T_i$ to each agent's *modeling structure* since it does not match $S_i$. The *modeling structure* of each agent, after selecting $C_5(S)$, is shown below:

$M_1$={⟨$S_1$,...},⟨$T_1$,$ag_2$,Null,1⟩, ⟨$T_1$, $ag_3$, Null, 1⟩, ⟨$T_1$,$ag_4$,*put*(**D**,**E**),1⟩, ⟨$T_1$,$ag_5$, Null,1⟩}
$M_2$={⟨$S_2$,...},⟨$T_2$,$ag_1$,*put*(**A**,**B**),1⟩,⟨$T_2$,$ag_3$,Null,1⟩,⟨$T_2$,$ag_4$,*put*(**D**,**E**),1⟩, ⟨$T_2$,$ag_5$, Null,1⟩}
$M_3$={⟨$S_3$,...},⟨$T_3$,$ag_1$,*put*(**A**,**B**),1⟩,⟨$T_3$,$ag_2$,Null,1⟩,⟨$T_3$,$ag_4$,*put*(**D**,**E**),1⟩, ⟨$T_3$,$ag_5$, Null,1⟩}
$M_4$={⟨$S_4$,...},⟨$T_4$,$ag_1$,*put*(**A**,**B**),1⟩,⟨$T_4$,$ag_2$,Null,1⟩,⟨$T_4$, $ag_3$, Null, 1⟩, ⟨$T_4$,$ag_5$, Null,1⟩}
$M_5$={⟨$S_5$,...},⟨$T_5$,$ag_1$,*put*(**A**,**B**),1⟩,⟨$T_5$,$ag_2$,Null,1⟩,⟨$T_5$, $ag_3$, Null, 1⟩, ⟨$T_5$,$ag_4$, *put*(**D**,**E**),1⟩}

The second example starts with the blocks as shown in Figure 4. The *activity context* selected is {*put*(**C**,⊥)} and the resulting configuration is shown in Figure 5. The *activity context* selected for Figure 5 is {*put*(**B**,**C**)}.

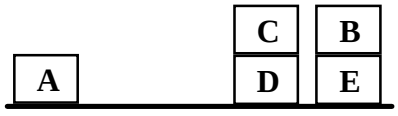


Figure 4: Initial Configuration for Example 2

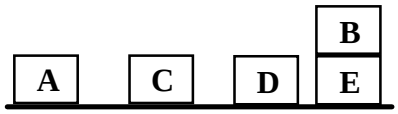


Figure 5: Second Configuration for Example 2

After both actions are executed, the configuration is the same as shown in Figure 3. The AGE algorithm will repeat the same series of steps including generating bids for all five *activity contexts*. However, an agent utilizing the MAGE algorithm matches this environmental state with the environmental state of the first problem. Therefore, each agent has a model of the other agents and can predict the actions of the other agents. An

agent generates only the most likely *activity contexts* and is not forced to generate all *activity contexts* for this configuration. For example, agent $ag_2$ matches the current environmental state with $T_2$, stored in its *modeling structure* as shown below.

$M_2$={⟨$S_2$,...},⟨$T_2$,$ag_1$,*put*(**A**,**B**),1⟩,⟨$T_2$,$ag_3$,Null,1⟩,⟨$T_2$,$ag_4$,*put*(**D**,**E**),1⟩, ⟨$T_2$,$ag_5$, Null,1⟩}

Agent $ag_2$ predicts the other agents will select these same actions in the given environmental state. Given these constraints, agent $ag_2$ generates only *activity contexts* that include the predicted actions. In this case, there is only one action, Null, for agent $ag_2$ to select and only one activity context is generated. Therefore, using the MAGE algorithm, agent $ag_2$ generates, computes, and announces a single bid instead of the five bids needed by the AGE algorithm. Each of the other agents will follow the same steps.

The type of learning demonstrated is rote learning, which requires the learning system to do the least amount of effort. In this context, the MAGE algorithm is memorizing the previous solution and recalling this information. The benefit of this learning is the time gained by each agent in avoiding the computation and communication for the extra four bids. The first time the problem is encountered, each agent announces five bids for a total of twenty-five messages. On subsequent encounters, each agent will announce only one bid for a total of five messages.

The MAGE algorithm also has more complicated learning strategies. With the previous examples, the MAGE algorithm matched the entire environmental state with previously stored environmental states. But, in this example, each agent senses a slightly different environmental state from the previous examples and, therefore, matches only part of the environmental state with part of the previously stored environmental states. To clarify the partial matching process, only agent $ag_4$ is modeled for this example. The partial environmental state used for matching in this example is from Table 4 and is shown in Table 5. The information shown in Table 5 is each agent's model for agent $ag_4$. The information for agent $ag_4$ is omitted since an agent does not model itself; an agent only models other agents. Since blocks **D** and **E** are the only blocks involved in the *put*(**D**,**E**) action, only information sensed for these blocks will be shown in the table. In addition, the information for agent $ag_1$ is also omitted since it cannot sense any information about the blocks, **D** and **E**, involved in the action and, thus, will never have enough information to accurately model the actions agent $ag_4$ takes for any environmental state.

| **Agent** | **Env. State** | **On** | **Clear** |
|---|---|---|---|
| $ag_2$ | $T_2$ | D⊥ | D |
| $ag_3$ | $T_3$ | D⊥ | D |
| $ag_5$ | $T_5$ | E⊥ | E |

Table 5: Environmental States from Table 4

Given the configuration shown in Figure 6, each agent's actions and environmental state sensed are shown in Table 6.

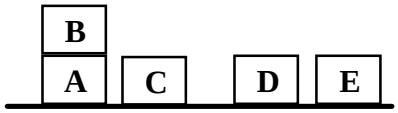


Figure 6: Configuration for Example 3

| **Agent** | **Possible Actions** | **Env. State** | **On** | **Clear** |
|---|---|---|---|---|
| $ag_1$ | Null | $U_1$ | A⊥, BA | B |
| $ag_2$ | *put*(**B**,**C**), *put*(**B**,**D**), Null | $U_2$ | BA, C⊥, D⊥ | B, C, D |
| $ag_3$ | *put*(**C**,**D**), Null | $U_3$ | C⊥, D⊥ | C, D |
| $ag_4$ | *put*(**D**,**C**), *put*(**D**,**E**), Null | $U_4$ | C⊥, D⊥, E⊥ | C, D, E |
| $ag_5$ | *put*(**E**,**B**), *put*(**E**,**C**), Null | $U_5$ | BA, C⊥, , E⊥ | B, C, E |

Table 6: Environmental States and Possible Actions for Figure 6

Using the AGE algorithm, each agent generates all compatible *activity contexts* and adds them to its *possible activity context*, *PAC*(*S*). The *PAC*(*S*), without Null actions, for this configuration is shown below. The number of possible *activity contexts*, *C*(*S*), within the *PAC*(*S*) is eleven; thus, each agent computes and broadcasts eleven bids. The activity context selected for execution will be $C_8(S)$, {*put*(**B**,**C**), *put*(**D**,**E**)}, since this selection transforms the configuration closer to the goal configuration than any of the other selections.

*PAC(S)* = { {*put*(**B**,**C**)}, {*put*(**B**,**D**)}, {*put*(**C**,**D**)}, {*put*(**D**,**C**)}, {*put*(**D**,**E**)},
{*put*(**E**,**B**)}, {*put*(**E**,**C**)}, {*put*(**B**,**C**), *put*(**D**,**E**)}, {*put*(**B**,**D**), *put*(**E**,**C**)},
{*put*(**C**,**D**), *put*(**E**,**B**)}, {*put*(**D**,**C**), *put*(**E**,**B**)}

Using the MAGE algorithm, each agent $ag_i$ first attempts to match the environmental state it senses, $U_i$, with a previous environmental state stored in its *modeling structure*, $M_i$. While there are no exact matches with the complete environmental state sensed by each agent, there are matches when considering only the information sensed for blocks **D** and **E**. To clarify this, only the information sensed for blocks **D** and **E** is shown in Table 7.

| Agent | Env. State | On | Clear |
|---|---|---|---|
| $ag_2$ | $U_2$ | D⊥ | D |
| $ag_3$ | $U_3$ | D⊥ | D |
| $ag_5$ | $U_5$ | E⊥ | E |

Table 7: Environmental States of Blocks D and E for Figure 6

Each agent, using this partial environmental state, can now match $U_i$ with $T_i$ from Table 5. This means each of the other agents selects the tuple, $\langle T_2, ag_4, put(\mathbf{D},\mathbf{E}), 1\rangle$, from its modeling structure $M_i$ and selects *put*(**D**,**E**) as the most likely action agent $ag_4$ will select. This information reduces the size of the *MPAC*(*S*), MAGE's version of the *PAC*(*S*), to two *activity contexts*, since only two contain the *put*(**D**,**E**) action.

*MPAC*(*S*) = { {*put*(**D**,**E**)}, {*put*(**B**,**C**), *put*(**D**,**E**)} }

In this example, the learning algorithm must search the *model structures* for a partial match and recall the relevant tuple from previous problem-solving experiences. The benefit of this is the time gained by avoiding the extra computation and communication required for the extra nine bids by each agent. The first time the problem is encountered, each agent announces eleven bids for a total of fifty-five messages being broadcast. On subsequent encounters, each agent broadcasts only two bids for a total of ten messages.

The final example illustrates learning that requires more inferencing than the previous learning strategies. Learning from examples requires the system to induce a general concept description from examples. This example learns a concept description from two of the previous examples and matches this description with an environmental state. It involves modeling agent $ag_2$'s actions as the other agents learn under what conditions agent $ag_2$ selects the *put*(**B**,**C**) action. In the first example, agent $ag_2$ selected *put*(**B**,**C**) when the other agents sensed the environmental states shown in Table 8. The concept description is learned by agents $ag_1$ and $ag_5$ so only their information is shown for Table 8. Since blocks **B** and **C** are the only blocks involved in the *put*(**B**,**C**) action, only information sensed for these blocks is shown in the table.

| Agent | Env. State | On | Clear |
|---|---|---|---|
| $ag_1$ | $S_1$ | BD | B |
| $ag_5$ | $S_5$ | BD, C⊥ | B, C |

Table 8: Environmental States of Blocks B and C for Figure 2

In the second example, agent $ag_2$ selected *put*(**B**,**C**) when the other agents sensed the environmental states shown in Table 9, from the configuration of blocks shown in Figure 5. Once again, the information for agents $ag_2$, $ag_3$, and $ag_4$ is omitted and only information sensed for blocks **B** and **C** are shown in the table.

| Agent | Env. State | On | Clear |
|---|---|---|---|
| $ag_1$ | $V_1$ | BE | B |
| $ag_5$ | $V_5$ | BE, C⊥ | B, C |

Table 9: Environmental States of Blocks B and C for Figure 5

Agents $ag_1$ and $ag_5$ sense differences between the two examples. In the first example, **B** is on top of **D** while in the second example **B** is on top of **E**. Therefore, a partial match does not work. Each agent's learning system must induce a concept description covering both environmental states. At this point, the learning system has a choice in determining how far to generalize the two examples. It can take a more conservative approach and describe the concept as **B** on top of only blocks **D** or **E**. In this case, a partial match would require **D** or **E** and would not allow any other blocks to match. A more liberal approach removes this restriction and describes the concept as **B** on top of any other block. This approach is taken in MAGE and the resulting concept description for each agent for this example is shown in Table 10.

| Agent | Env. State | On | Clear |
|---|---|---|---|
| $ag_1$ | $SV_1$ | B* | B |
| $ag_5$ | $SV_5$ | B*, C⊥ | B, C |

Table 10: Concept Description for Modeling $ag_2$ selecting *put*(**B**,**C**)

where * represents a variable or wild card that matches any of the other blocks. This new, partial environmental state, $SV_i$, is added to the modeling structures of agents $ag_1$ and $ag_5$.

This new concept is used for helping to solve the configuration of blocks for example 3, shown in Figure 6. Using the partial matching discussed earlier, there are still two *activity contexts* in the *MPAC*(*S*). Each agent's actions and environmental states sensed are shown in Table 6. The information for agent $ag_4$ is omitted since it is the agent being modeled. Only information for blocks B and C are shown for the other agents.

| Agent | Env. State | On | Clear |
|---|---|---|---|
| $ag_1$ | $W_1$ | BA | B |
| $ag_3$ | $W_4$ | C⊥ | C |
| $ag_4$ | $W_5$ | C⊥ | C |
| $ag_5$ | $W_3$ | BA, C⊥ | B, C |

Table 11: Environmental States of Block **B** and **C** for Figure 6

Agents $ag_3$ and $ag_4$ make partial matches and select *put*(**B**,**C**) as the most likely selection of agent $ag_2$. In addition, agents $ag_1$ and $ag_5$ will match BA from the environmental state with B* from their concept descriptions and, thus, also select *put*(**B**,**C**) as $ag_2$'s most likely selection. This means each agent will select $C_2(S)$, {*put*(**B**,**C**), *put*(**D**,**E**)}, from the *MPAC*(*S*) since it is the only *activity context* to satisfy the constraints from the partial and concept matches.

## 5 CONCLUSION

This paper introduced the Modeling Action Group Estimation algorithm (MAGE) which improves the Action Group Estimation (AGE) algorithms developed by Weiss[4]. These algorithms allow groups of agents to learn to coordinate their actions by adjusting estimates for individual actions and groups of actions. The MAGE algorithm introduces a *modeling structure* that reduces the number of activity contexts and bids needed and thus reduces the overhead required to select the activity context to be executed for each cycle in the problem solving process.

## REFERENCES


[1] Holland, J.H., "Properties of the bucket brigade", Proceedings of the First International Conference on Genetic Algorithms and Their Applications, J.J. Grefenstette (Ed.), 1985, pp. 1-7

[2] Weiss, G., "The action-oriented bucket brigade", Technical Report FKI-156-91, Institut für Informatik, Technische Universität München, 1991

[3] Weiss, G., "Learning the Goal Relevance of Actions in Classifier Systems", Proceedings of the 10th European Conference on Artificial Intelligence, B. Neuman (Ed.), Wiley, 1992, pp. 430-434

[4] Weiss, Gerhard, "Learning to Coordinate Actions in Multi-Agent Systems", IJCAI-93, Chambery, France, August 1993, pp. 311-316

[5] Singh, Munidar P., Michael N. Huhns and Larry M. Stephens, "Declarative Representations of Multiagent Systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 5, October 1993, pp. 721-739

[6] Brazdil, P. and M Gams, S. Sian, L. Torgo, W. van de Velde, "Learning in Distributed Systems and Multi-Agent Environments", Machine Learning - EWSL-91, Proceedings of the European Workshop Session on Machine Learning, Porto, Portugal, Springer-Verlag, March 6-8,1991, pp. 412-423

[7] Brazdil, P. and S. Muggleton, "Learning to Relate Terms in a Multiple Agent Environment", Machine Learning - EWSL-91, Proceedings of the European Workshop Session on Machine Learning, Porto, Portugal, Springer-Verlag, March 6-8,1991, pp.424-439

[8] Sian, Sati S., "Extending Learning to Multiple Agents: Issues and a Model for Multi-Agent Machine Learning", Machine Learning - EWSL-91, Proceedings of the European Workshop Session on Machine Learning, Porto, Portugal, Springer-Verlag, March 6-8,1991, pp.440-456

[9] Dowell, Michael L. and Ronald D. Bonnell, "Learning for Distributed Artificial Intelligence Systems", Proceedings of the Twenty-Third Southeastern Symposium on System Theory, Robert Werner (Ed.), 1991 pp. 218-221