$ALP4AI$: Agent-based Learning Platform for Introductory Artificial Intelligence

Ramoni O. Lasisi and Robert Dupont
Department of Computer and Information Sciences
Virginia Military Institute
LasisiRO@vmi.edu and dupontrw20@mail.vmi.edu

Abstract

We develop *ALP4AI*, an *Agent-based Learning Platform for Introductory Artificial Intelligence*. *ALP4AI* is a graphical-based tool that is suitable for teaching introductory AI, places emphasis on hands-on learning, and provides for visualization of results. The tool we have developed is suitable for solving problems in both the *state space search* and *local search* problem domains. It provides for different environments modeling, including, environments that contain obstacles or are obstacle-free, single or multi-agent, uses micro or macro states, and contains single or multi goals. Students can also conduct and report results of experiments using *ALP4AI*.

1 Introduction

There has been much emphasis on hands-on and experiential learning in undergraduate computer science education (Rodger 2002), including computer programming education (von Hausswolff 2017) and artificial intelligence (Parsons and Skiar 2004; Bryce 2011; Perhinschi 2017). We set out to design, develop, and implement a new simulation and learning tool in artificial intelligence (AI) that we tagged, *ALP4AI: Agent-based Learning Platform for Introductory Artificial Intelligence.* The idea to develop this tool was conceived after teaching several sections of the Intoductory AI course to undergraduate students in a four-year college. Many of these students have completed two semesters of introductory programming in Java, and have basic introduction to a data structures course. Topics for the course were selected from the well-known and widely used AI textbook: *Artificial Intelligence - A Modern Approach, Third Edition* (Russell and Norvig 2010).

Many traditional methods of teaching computer science (CS) topics including AI algorithms (from instructors and textbooks) in most part provide description of the methods and use some data to illustrate the functionalities of these ideas. These descriptions may in some cases be followed with vizualization of the steps involved before students are asked to provide implementations of the ideas. Examples using this approach abound in many CS textbooks such as those used in introductory programming courses, data structures, design and analysis of algorithms, and several topics in AI. One concern about this approach is that some of these important topics in the field of AI are sophisticated and are difficult concepts for lower level undergraduate students to easily comprehend. Another issue is how much of the knowledge gained from using this approach is retained, and can be applied when students are faced with new problems from different domains than they have been exposed?

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

There have been several attempts in the literature to address this concern in AI using hands-on learning approaches to teaching AI algorithms. Parsons and Skiar (2004) use LEGO Mindstorms in teaching AI. The approach utilized by the authors is more towards students being able to program the robots and as well test out some of the functionalities while engaging in contests among the project groups. Thus, their approach is not about formulating problems or implementing AI solutions or algorithms. In a different hands-on method, Bryce (2011) use a project-based approach in the game of Wumpus World (WW) to teach introductory AI concepts. Description of the WW environment can be found in (Russell and Norvig 2010). Students are required to implement search, satisfiability, and declarative planning descriptions algorithms applied to the WW of different dimensions. As interesting as the WW project seems, it is bland and provides for a single agent in the environment. Having a multi-agent environment would be interesting to see and work with as well. It will open up new possibilities to seeing how different algorithms work in different environments.

Another work on introductory AI uses Java-based games (McGovern, Tidwell, and Rushing 2011). Although, the games here are graphical and also have elements of multi-agent environments, two of the three projects have students implement several variants of A^* algorithms. Finally, we looked at a related course that uses the Pac-Man game to illustrate the introduction of AI algorithms (DeNero and Klein 2010). In this course, students are required to implement various algorithms (single and multi-agents) to solve problems in the Pac-Man domain. In contrast to the works above, our emphasis in this project is to build a tool that provides not only hands-on and experiential learning to implement basic AI algorithms as many of the works cited did, but also to teach students the process of problem formulation and development of solutions that students can be able to apply to new problem domains in their future courses or careers.

We have developed *ALP4AI* to be a simple tool with little learning curve. Thus, students *do not* need several hours of

study or class periods to understand the details of and how to use the simulator. *ALP4AI* is a graphical-based AI learning tool written entirely in Java. It makes provision for several functionalities that students can use to model different AI problems and develop solutions for. These functionalities are based on the following themes:

- **Obstacle**: No obstacle or obstacles are present in an environment
- **Agent**: Single agent or multi-agent environment
- **State**: micro or macro state modeling of an environment
- **Goal**: Single goal or or multi-goal are present in an environment

An interesting part of the *ALP4AI* simulator is that themes can be combined in several ways to implement and test AI algorithms. For example, an environment may be defined to contain several obstacles, multi-agent, uses macro state, and be multi-goal. We initially set out to develop *ALP4AI* for four problem domains in AI, including, *state space search, local search, reasoning using propositional logic, and reasoning using first order logic*, however, we have been able to complete only two of the domains (state space search and local search) that we report in this paper.

2 The $ALP4AI$ Description

We provide a description of $ALP4AI$, our agent-based learning platform for introductory AI. $ALP4AI$ framework is situated in a two-dimensional grid that represents environments that agents need to explore. Agents are given the task of locating goals that are randomly placed in the environment. Goals represent desirable situations that agents need to achieve. Many interesting problems in introductory AI, including *state space search* and *local search* that we use to illustrate the functionalities of $ALP4AI$, can be modeled to use this environment.

$ALP4AI$ allows for parameterization of the environments. Different parameters that can be defined include, among others, provisions for *single* or *multi-agent* environments, presence of *single* or *multi-goal*, and *obstacle-free* or *obstacles-present* environments. Environments are further systematically organized into *states* so that agents can intelligently navigate them. We provide two implementations for agents' states: *micro* and *macro* states. The use of the idea of states to organize environments in $ALP4AI$ is fundamental to understanding basics of problems formulation in introductory AI. Furthermore, the problem domains that we consider in this work require a good understanding of the concept of states and *states' successors* to formulate AI problems.

Let s be a state that represents some abstractions of the world (i.e., environment) that an agent is currently in, the successor states are defined as the possible states that the agent can transits to from s. We present our framework of the concept of a state and its successors for both the micro and macro states framework implementations.

Micro State Framework

The micro state framework provides a straightforward description for states in an environment.

Definition 1. *A micro state s is determined by the indices of the cell an agent is currently in, as well as the direction the agent is facing. There are four directions, north, east, west, and south, that an agent may be facing.*

For example, an agent that is currently in a cell, say $grid[i][j]$, of the environment and facing north is in a different state than when it is in the same cell but facing south.

Definition 2. *The successors, $succ(s)$ of a micro state s consists of all micro states s' that share boundary with s.*

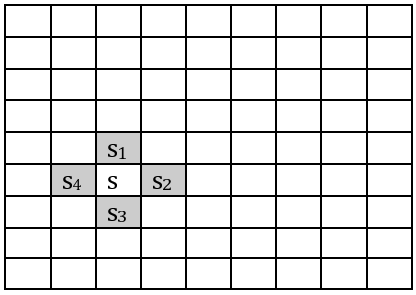

Figure 1: A micro state s (depending on where an agent is facing) and successors for s. $succ(s) = \{s_1, s_2, s_3, s_4\}$.

Note that the successors are potential states which are ultimately determined by the direction an agent will be facing after it transits to them. Figure 1 is an illustration of the concept of a micro state and its successor states. The micro state framework is a deliberate attempt to gently introduce and demonstrate to students several concepts of problem formulation in AI, including, percepts, actions, states, successors, transition model, goal, step costs, and path costs.

Macro State Framework

We provide a description of a more challenging framework that further probes students' understanding of the basic AI concepts learned and implemented using the micro state framework. Denote by c_{ij} a cell in the grid of an environment where $i, j \in \mathbb{N}$ are the Cartesian coordinates of c_{ij}.

Definition 3. *A macro state s with a reference cell, c_{ij}, consists of the reference cell c_{ij}, and all immediate cells $c'_{i'j'}$ from c_{ij} such that $|i - i'| = 1$ or $|j - j'| = 1$. Unlike in the micro state framework, the direction an agent is facing in the reference cell of a macro state is unimportant.*

An example of a macro state labeled s is shown in Figure 2 with a reference cell c_{ij}. The immediate cells from c_{ij} are shaded in gray. It is clear from Definition 3 that a macro state is composed of a 3×3 sub-matrix within an environment when the reference cell is not close to or on a boundary cell. We define n to be a multiple of 3 for any $n \times n$ two-dimensional grid of the environment. This constraint allows us to correctly map macro states to the $n \times n$ grids.

Definition 4. *The successors, $succ(s)$ of a macro state s consists of all macro states s' that share a boundary with s.*

Figure 2 shows an example of successors, $succ(s)$ for a macro state s. States s_1, s_2, s_3, and s_4 (shaded in

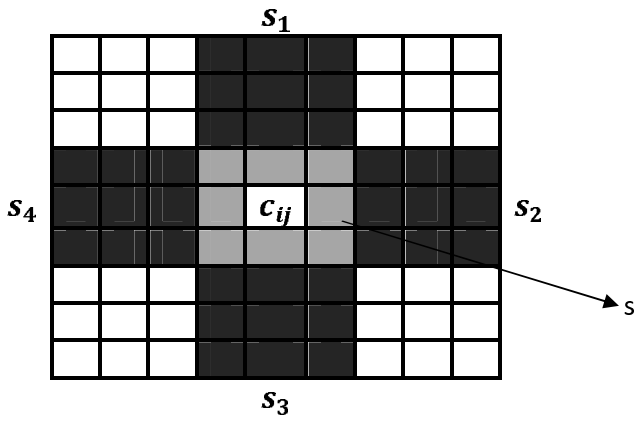

Figure 2: A macro state (with reference cell c_{ij}), denoted s, and successors, $succ(s)$ for s. $succ(s) = \{s_1, s_2, s_3, s_4\}$.

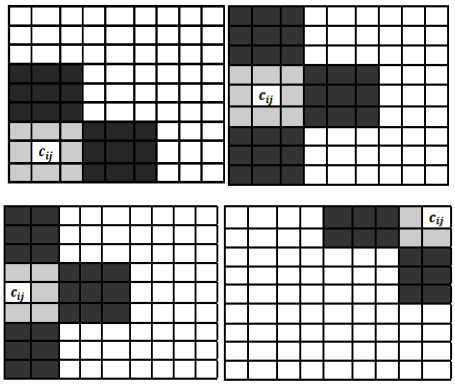

Figure 3: A view of some possible successors for different macro states with reference cells, c_{ij}

.

black) all share a boundary with s. Thus, $succ(s) = \{s_1, s_2, s_3, s_4\}$. The size, $|succ(s)|$, of the successors of a state s is $2 \leq |succ(s)| \leq 4$, depending on whether or not s is close to or on any of the boundaries. The successors of any macro state that is far away from the boundaries consist of only four successor macro states as shown above.

Figure 3 shows a view of some possible successors for different macro states with reference cells, c_{ij}. Notice how incomplete (both in the number of cells and successors) in how some of the successors were defined because of the positions of the reference cells. An agent will randomly be placed at a location. This location corresponds to a certain reference cell in the environment's grid. Since a macro state is made up of a 3×3 sub-matrix and the dimensions of a grid is a multiple of 3, we can locate the starting cell for a state.

Problem Formulation

The following is the formal problem formulation that we used to model domains that are considered in the $ALP4AI$ platform for both the micro and macro states frameworks:

- **States**: For the case of the micro state framework, a state consists of a cell in an environment including the direction an agent is facing. Whereas for the case of the macro state framework, we start by letting $i, j \in \{1, 2, 3\}$. A state is then identified as any $i \times j$ (excluding the 1×1) sub-matrix in the environment as determined by a reference cell.
- **Initial state**: Agents are randomly placed in a state in an environment when starting a new world. So any state in the environment may be designated as the initial state. However, in order to allow for repeatable experiments that compare performance of strategies in the different problem domains, we also make provisions for placing agents and goals in desired starting positions.
- **Actions**: The following four actions, *goForward, turnLeft, turnRight*, and *pickGoal*, are made available to an agent in the micro state framework. Thus, an agent can move from its current location in only one of four directions (i.e., north, east, west, and south) and is limited to moves of *length one* in a single time step. In the case of the macro state framework, however, an agent has five actions, *goLeft, goRight, goTop, goBottom*, and *pickGoal*. Unlike the actions in the micro state, each of these actions (except *pickGoal*), directly places an agent in the reference cell of the successor state that is selected by the transition model.
- **Transition model**: Appropriate transition models (also referred to as successor functions) that agents can use to select successor states are developed in the different problem domains. Examples of such functions are given in the next section.
- **Goal test**: For the case of the micro state framework, when an agent is initially placed in a state, it checks if the state is a goal state and reports accordingly. Whereas for the case of the macro state framework, when an agent is initially placed in a state, the agent first checks if the reference cell is a goal position and reports accordingly. If not, it performs a search of the immediate cells to the reference cell of the state and checks if the cell being searched contains the goal and reports accordingly.
- **Path cost**: Each step by an agent in either of the two frameworks costs 1. So, for a macro state with a 3×3 submatrix, the total time step is 10. This includes the cost of doing a goal test (in the 9 cells of the state) plus the cost of going directly to the reference cell of a successor state.

3 Problem Domains Used in $ALP4AI$

We provide a description of the two problem domains from introductory AI that are used in $ALP4AI$. The domains are the state space search and the local search.

State Space Search

The set of all possible states in a problem domain constitutes the search space that agents seek for feasible solutions. We model search space for the $ALP4AI$ platform that students are required to provide implementation of search algorithms (uninformed and informed) for. The search algorithms are expected to allow agents to reach goals that are randomly

placed in varying level of environments difficulty, including multi-agent and multi-goal environments. The strategies to be implemented include *Breadth First, Depth First, Iterative Deepening Depth First, Uniform Cost, Best First* and A^* search algorithms.

Students are first required to implement transition models for both the micro and macro states frameworks. Given a particular state of an environment and an agent's action, the successor function computes and returns the possible next states from the given state. The node[1] that corresponds to the given state is thus said to have been *expanded*. The successor nodes also correspond to different valid successor states that the agent can transit to. One essential skill that is expected to be gained here is the understanding of the *concept of a node in a tree and how it relates to the abstraction of a state in an environment*. All the search strategies employ the successor function to expand nodes. For example, suppose an agent is at location grid[0][0] in a micro state, facing west, and assuming that the goal is not at this location. Actions such as *goForward* and *pickGoal* are not applicable in this state. However, actions *turnRight* and *turnLeft* are applicable, thus their corresponding successor states (on application of the actions) will be returned by the successor function.

Algorithm 1 is the pseudocode for the transition model of our state space search domain. First, we implement a node data structure that is used to abstract information from the state it describes. Some data members in the data structure are: *parentNode, parentAction, pathCost*, *x,* and *y* indices that denote if a cell is in a micro or macro state. Actions $turnRight$ and $turnLeft$ are applicable in every micro state.

Local Search

Another domain considered in $ALP4AI$ is the local search problem domain. This domain allows students to appreciate the main differences and performance improvement from using the search algorithms in the state space search domain compared to local search algorithms. The search strategies to be implemented include several variants of the *hill climbing* algorithm. These variants are determined by the successor function used by the hill climbing algorithm. Again, students are required to implement successor functions that can also be used in multi-agent and multi-goal environments. The main take away we want students to experience using this problem domain is the understanding of the drastic reduction in the amount of nodes explored and the memory usage from state space search to local search algorithms. We implement a local search transition model called *stochastic local search*, with and without heuristics. Below provides the pseudocodes for the two models. Algorithms 2 and 3 are the pseudocodes for the transition model of our loca search domain.

Further, unlike in the state space search where the outcome of a search is a sequence of actions that leads from the initial state to the goal state, students are made aware that local search methods are less concerned with the sequence of these actions. We demonstrate implementation of two successor functions for the hill climbing algorithm, including randomly selecting successor nodes with and without heuristics in both the micro and macro state frameworks for the $ALP4AI$ platform. Since the action sequences are less important in local search, students are required to keep track of all states visited by agents in the process of finding a goal state. So, the outcome of a local search method is shown by coloring visited states in the environments green.

[1]Note the distinction between a state and its corresponding node.

Algorithm 1 : State space search transition model

Input: A node, $node$, that corresponds to a state
Output: A list of successor nodes to $node$

```
1: procedure expand(node)
2:     successors ← ∅
3:     if node.microState then
4:         return expandMicroState(node, successors)
5:     else
6:         return expandMacroState(node, successors)
7: end procedure
8:
9: procedure expandMicroState(node, successors)
10:    make new nodes, node1, node2, node3, from node
11:    node1.turnRight
12:    node2.turnLeft
13:    node1.pathCost ← node.pathCost + 1
14:    node2.pathCost ← node.pathCost + 1
15:    successors ← successors ∪ {node1, node2}
16:    if node.goForward is a valid action then
17:        node3.goForward
18:        node3.pathCost ← node.pathCost + 1
19:        successors ← successors ∪ {node3}
20:    return successors
21: end procedure
22:
23: procedure expandMacroState(node, successors)
24:    make node1, node2, node3, node4 from node
25:    if node.goLeft is a valid action then
26:        node1.goLeft
27:        node1.pathCost ← node.pathCost + 1
28:        successors ← successors ∪ {node1}
29:    if node.goRight is a valid action then
30:        node2.goRight
31:        node2.pathCost ← node.pathCost + 1
32:        successors ← successors ∪ {node2}
33:    if node.goTop is a valid action then
34:        node3.goTop
35:        node3.pathCost ← node.pathCost + 1
36:        successors ← successors ∪ {node3}
37:    if node.goBottom is a valid action then
38:        node4.goBottom
39:        node4.pathCost ← node.pathCost + 1
40:        successors ← successors ∪ {node4}
41:    return successors
42: end procedure
```

Algorithm 2 : Local search transition model without heuristic

Input: An agent a and current state s
Output: A successor state s'

procedure $successorState(a, s)$
  $successors \leftarrow \emptyset$
  foreach state $s' \in succ(s)$ **do**
    if s' has not been visited **then**
      $successors \leftarrow successors \cup \{s'\}$
  if $successors$ is empty **then**
    randomly select an s' from $succ(s)$
  else
    randomly select an s' from $successors$
  return s'
end procedure

Algorithm 3 : Local search transition model with heuristic

Input: An agent a and current state s
Output: A successor state s'

procedure $successorState(a, s)$
  $successors \leftarrow \emptyset$
  foreach state $s' \in succ(s)$ **do**
    if s' has not been visited **then**
      $successors \leftarrow successors \cup \{s'\}$
  if $SuccStates$ is empty **then**
    randomly select an s' from $succ(s)$
  else
    $s' \leftarrow \min_{s \in successors} distance_{est}(s, goal)$
  return s'
end procedure

4 $ALP4AI$ Using Micro State Framework

We provide snapshots to illustrate results of search algorithms using the micro state framework. Agents are shown in pictures with arrow heads while the goals are the diamond pictures shown in yellow. The black cells are obstacles that agents need to avoid. Agents, goals, and obstacles are all randomly distributed in the environments. Figure 4 is a 15×15 microstate environment showing the final search of a grid by an agent using the breadth first search to locate a goal in the state space search domain. A similar result is shown in Figure 5 but for three agents conducting search of an obstacles-present environment to locate three goals. Figure 6 is a 30×30 microstate environment showing the final search of a grid by two agents using the A^* search to locate three goals in the state space search domain. There are no obstacles in this environment.

5 $ALP4AI$ Using Macro State Framework

We also provide a snapshot to illustrate a result of search algorithms using the macro state framework. Agents are shown in pictures with arrow heads while the goals are the diamond pictures shown in yellow. Figure 7 is a 30×30 macrostate environment showing the search of a grid by two

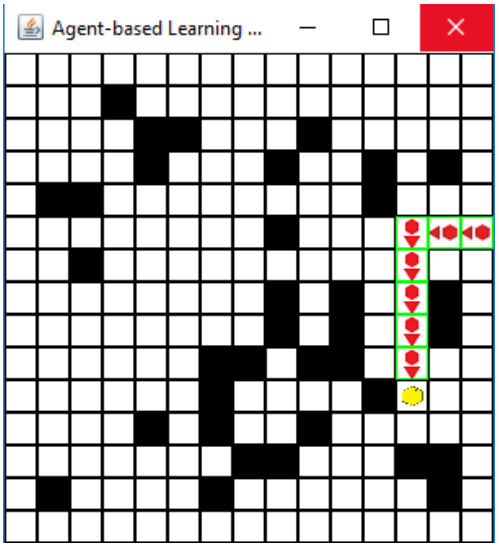

Figure 4: A 15×15 microstate environment showing the final search of a grid by an agent using the breadth first search to locate a goal in the state space search domain.

agents using the breadth first search to locate two goals in the state space search domain. There are no obstacles in this environment.

6 Evaluation

We measure the performance of the state space search algorithms as well as those of the local search algorithms using different grid sizes of the environments, determining the number of nodes expanded, average time to expand a node (in milliseconds), total CPU time to find goals (in milliseconds), and the number of goals found. Experiments are based on several combinations of the *ALP4AI* themes: obstacle, agent, state, and goal. *ALP4AI* makes provisions for conducting set of simulations and experiments to evaluate performance of suggested solutions by students. The setup of each experiment includes random placement of agent(s) and goal(s) in the environments. We allow agents to search for goals using the successor functions described for both the state space search and local search domains, and with or without heuristics. 50 trials of each experiments are expected to be completed and the results averaged.

7 Conclusions and Future Work

We develop *ALP4AI*, an *Agent-based Learning Platform for Introductory Artificial Intelligence*. The tool we have developed is suitable for solving problems in the state space search and local search problem domains in AI. $ALP4AI$ allows for parameterization of the environments. Different parameters that can be defined include provisions for single or multi-agent environments, presence of single or multi-goal, and obstacle-free or obstacles-present environments. Environments are further systematically organized into states so that agents can intelligently navigate them. Finally, we also provide two implementations for agents'

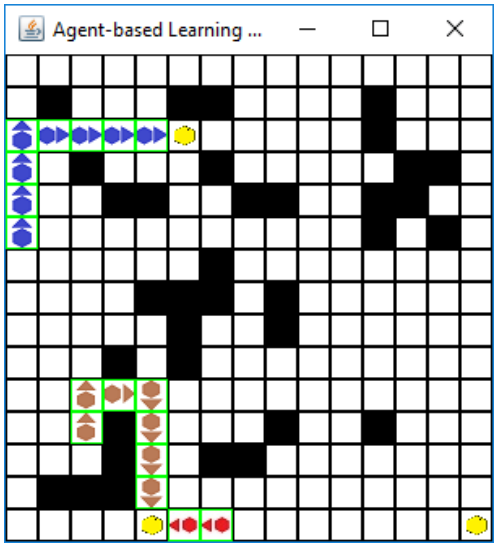

Figure 5: A 15×15 microstate environment showing the final search of a grid by three agents using the breadth first search to locate three goals in the state space search domain. Obstacles are present in this environment.

states: micro and macro states. Illustrations showing final results using the two problem domains are also discussed.

Future work will extend this platform to two more problem domains in AI: reasoning using propositional and first order logics.

References

Bryce, D. 2011. Wumpus world in introductory artificial intelligence. In *Consortium for Computing Sciences in Colleges*, 58–65.

DeNero, J., and Klein, D. 2010. Teaching introductory artificial intelligence with pac-man. In *Symposium on Educational Advances in Artificial Intelligence*.

McGovern, A.; Tidwell, Z.; and Rushing, D. 2011. Teaching introductory artificial intelligence through java-based games. In *Second Symposium on Educational Advances in Artificial Intelligence*, 1729–1736.

Parsons, S., and Skiar, E. 2004. Teaching ai using lego mindstorms. In *Greenwald, L., Dodds, Z., Howard, A., Tejada, S., Weinberg, J. (eds.) Accessible Hands-on AI and Robotics Education*, 8–13.

Perhinschi, M. G. 2017. An introductory course on computational artificial intelligence techniques for engineering students. *Computers in Education Journal* 8(3):1–9.

Rodger, S. H. 2002. Using hands-on visualization to teach computer science from beginning courses to advanced courses. In *Second Program Visualization Workshop*.

Russell, S., and Norvig, P. 2010. *Artificial Intelligence: A Mordern Approach Third Edition*. Prentice Hall.

von Hausswolff, K. 2017. Hands-on in computer programming education. In *2017 ACM Conference on International Computing Education Research,*.

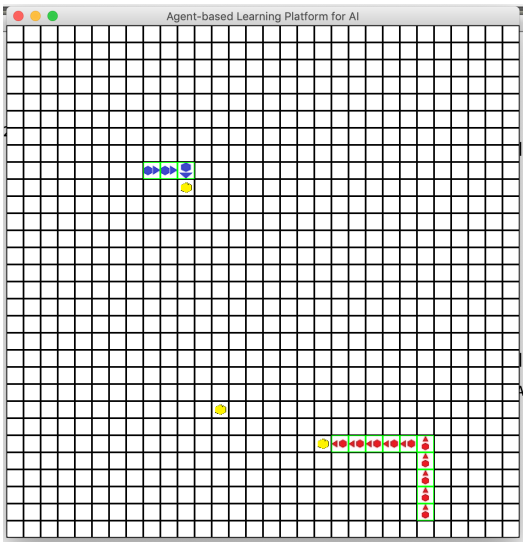

Figure 6: A 30×30 microstate environment showing the final search of a grid by two agents using the A^* first search to locate three goals in the state space search domain. There are no obstacles in this environment.

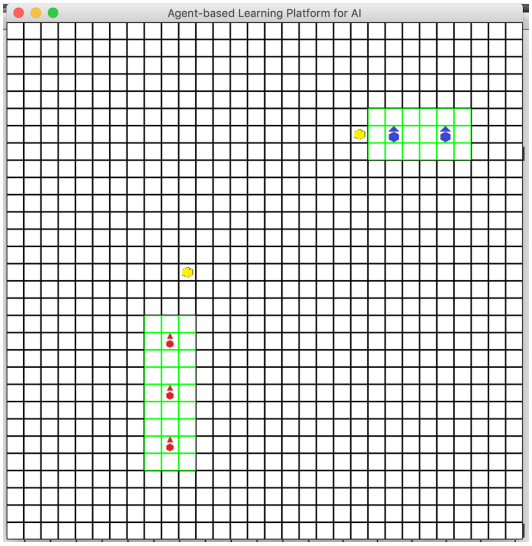

Figure 7: A 30×30 macrostate environment showing the final search of a grid by two agents using the breadth first search to locate two goals in the state space search domain. Notice the green color around the macro states visisted by each of the agents.