CIT. Journal of Computing and Information Technology, Vol. 25, No. 1, March 2017, 31–45
doi: 10.20532/cit.2017.1003573

Unibot, a Universal Agent Architecture for Robots

Goran Zaharija[1], Saša Mladenović[1] and Lada Maleš[2]

[1]Faculty of Science, University of Split, Split, Croatia
[2] Faculty of Humanities and Social Sciences, University of Split, Split, Croatia

Today there are numerous robots in different applications domains despite the fact that they still have limitations in perception, actuation and decision process. Consequently, robots usually have limited autonomy, they are domain specific or have difficulty to adapt on new environments. Learning is the property that makes an agent intelligent and the crucial property that a robot should have to proliferate into the human society. Embedding the learning ability into the robot may simplify the development of the robot control mechanism. The motivation for this research is to develop the agent architecture of the universal robot – Unibot. In our approach the agent is the robot i.e. Unibot that acts in the physical world and is capable of learning. The Unibot conducts several simultaneous simulations of a problem of interest like path-finding. The novelty in our approach is the Multi-Agent Decision Support System which is developed and integrated into the Unibot agent architecture in order to execute simultaneous simulations. Furthermore, the Unibot calculates and evaluates between multiple solutions, decides which action should be performed and performs the action. The prototype of the Unibot agent architecture is described and evaluated in the experiment supported by the Lego Mindstorms robot and the NetLogo.

ACM CCS (2012) Classification: Computing methodologies → Artificial intelligence → Knowledge representation and reasoning → Cognitive robotics

Computing methodologies → Artificial intelligence → Distributed artificial intelligence → Intelligent agents

Computing methodologies → Artificial intelligence → Distributed artificial intelligence → Multi-agent systems

Computing methodologies → Machine learning → Machine learning approaches → Neural networks

Keywords: agent-based systems; learning and adaptive systems; AI reasoning methods

1. Introduction

Intelligence can be regarded as a concept humans employ to describe actions of a certain quality [1]. However, it is obvious that, after the decades of study, we still do not know very much about it [2]. In between plethora of intelligence definitions [3], experts tend to agree that intelligence is presented by the capacity to learn from experience and the capacity to adapt to an environment. Traditional artificial intelligence (AI) proceeds to developmental, human-like, functional computer models. In general, AI can be understood regarding computer programs where provided input is processed to generate desired output at the end. Analogously, the natural intelligence is perceived as a result of complex computer operations [4]. When applying these ideas to build robots in the real world, it is difficult to achieve good results relying on mentioned intelligence perception. In this situation, robot's intelligence is limited by the programmer's expressiveness in a specific formal programming language. As social robots are entering into a human physical environment, efficient concept and task learning, and collaboration between both robots and humans are to be expected [5]. If the robot is unable to perform a given task, a teacher must be able to teach it without the necessity to be an expert programmer. To be able to learn, the robot first has to share the mental representation of the concepts in the world used by its teacher. It must be able to quickly learn some new concept from the human or robot teacher after just a few trials. It is unacceptable to require several hundred or thousands of trials as in classical machine learning approaches. Knowledge transfer as a new learning framework, if done successfully, would greatly improve the performance of machine learning [6]. Deep-learning (DL) is currently one of the most commonly used techniques in machine learning [7]. DL methods can be divided in two categories: recognition (perception) and control (decision making). Deep-learning has many applications in artificial intelligence [8]. Although DL provides a great advantage in high-dimensional data representation and processing by using multiple levels of abstraction [8], it requires complex computations and a big training set. Also, those levels of abstraction are not easily recognizable by humans. Given an Artificial Neural Network (ANN) trained by one of common DL techniques, it is not clearly obvious what each level of ANN represents. Our aim is to be able to teach a robot complex tasks that can later be easily executed, explained, and formalized to be used by another robot or a person. That means that every layer of trained ANN must represent a series of concepts that are easily recognizable by humans. Consequently, learning techniques described in this paper do not use standard DL methods. Moreover, once a new skill is learned, the robot is competent, therefore able to provide assistance to a robot or human partner by sharing acquired knowledge. Corresponding experimental study belongs to the category of applied researches that consider both theoretical and practical research goal [9]. The theoretical research goal is achieved by formulating and evaluating the formal model of the proposed agent architecture for selecting appropriate search algorithm based on problem classification. The practical goal is carried out by implementing and evaluating knowledge, methods, and tools based on the observed system for solving path-finding problems in a maze, which is one of the most commonly used problem representations. Main goals of this experimental research are designing new intelligent agent architecture and developing a prototype based on described approach. The long term goal of this research is to contribute to the theory of intelligent agent architectures, multi-agent systems, and problem-solving systems, with the emphasis on the field of robotics. While considering what characteristics the Unibot must have to effectively be part of the human world, we examine both the problem of learning and that of collaborating with other robots and humans. By observing how people and animals learn, other forms of individual and social learning, and training techniques have been investigated, like learning from demonstration [10], clicker training [11], learning by observation [12], and tutelage [13]. Those learning techniques are based on learning perspectives: behavioral, cognitive and social cognitive, used in synergy in education [14]. Learning perspectives for the creation of the Unibot are interesting as a logical alternative to programming robots. In the following section the robot learning based on human learning perspective is presented, creating both the theoretical and the practical frameworks for the Unibot realization. In the second section two learning perspectives implemented in the proposed architecture are explained. The third section describes the problem solving process that is comprised of the agent environment definition, the problem representation, the explanation of used distance metrics and search algorithms. In the fourth section, the agent learning process, the overview of the Unibot agent architecture and the Multi-Agent Decision Support System is presented. The experimental evaluation supported by the Lego Mindstorms robot and the NetLogo, a programmable modeling environment, is given in the fifth section.

2. Learning

Learning can be defined as the change in behavior that must be brought about by the interaction of a person with its environment. Thus, learning can be defined as any relatively permanent change in behavior, knowledge, and cognitive skills, which comes about through experiences [15]. As previously mentioned, human learning can be viewed from three major perspectives. In the following sections two of those perspectives will be presented, taking into account that the learner in this study is the robot that can be used in robotic engineering and artificial intelligence course [16].

2.1 Behavioral Learning Perspective

From the behavioral perspective, learning is defined regarding observable events, called stimuli and responses. Stimulus is an observable environmental event that has a potential to exert control over a behavioral response. The response is an obvious behavior by a learner.

Behavioral learning perspective includes the concept of classical conditioning. Classical conditioning is a type of learning based on the association of a stimulus that does not ordinarily elicit a particular response with another stimulus that does elicit the response [16]. A brief presentation of robot learning principles based on the classical conditioning through reinforced response and operant conditioning, and cognitive learning is described. Classical conditioning defines two types of stimulus and two types of response. They are unconditioned stimulus, conditioned stimulus, unconditioned response, and conditioned response, as explained in Figure 1.

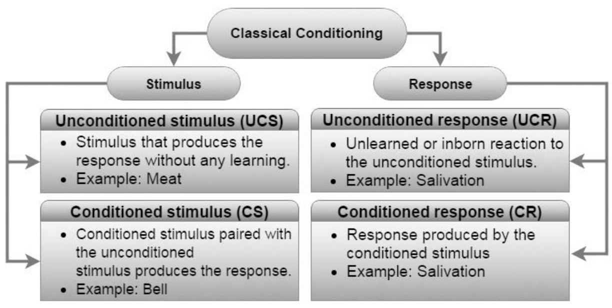
Classical Conditioning
Stimulus
Response
Unconditioned stimulus (UCS)
• Stimulus that produces the response without any learning.
• Example: Meat
Unconditioned response (UCR)
• Unlearned or inborn reaction to the unconditioned stimulus.
• Example: Salivation
Conditioned stimulus (CS)
• Conditioned stimulus paired with the unconditioned stimulus produces the response.
• Example: Bell
Conditioned response (CR)
• Response produced by the conditioned stimulus
• Example: Salivation

Figure 1. Types of stimuli and responses in classical conditioning.

2.2. Cognitive Learning Perspective

From the cognitive learning perspective, learning involves the transformation of information from the environment into knowledge stored in mind. Learning occurs when new knowledge is acquired, or existing knowledge is modified by experience. Cognitive learning theories are used to explain simple tasks such as remembering the name of a new friend as well as the complex ones such as interpreting an abstract drawing. This learning approach focuses on how children process information through attention, memory, thinking, and other cognitive processes. Social cognitive learning perspective examines the process involved as people learn from observing others and gradually acquire control over their behavior. In other words, social cognitivist believes that people learn a new behavior simply by watching what other people do.

Human-style tutelage is a social and a collaborative process [12] and usually takes the form of dialogue, a fundamentally cooperative activity [13]. To be a good instructor, one must maintain an accurate mental model of the learner's state (e.g., what is understood so far, what remains confusing or unknown) to appropriately structure the learning task with timely feedback and guidance. The learner (robot or otherwise) helps the instructor by expressing their internal state via communicative acts (e.g. expressions, gestures, or vocalizations that reveal understanding, confusion, attention, etc.). Through reciprocal and tightly coupled interaction, the learner and instructor cooperate to help both, the instructor to maintain a good mental model of the learner, and the learner to leverage from the instruction to build the appropriate models, representations, and associations.

Regardless of the learning perspective used, after forming a proper mental model the learner will be able to apply it to a problem-solving.

3. Problem Solving

Problem-solving in Artificial Intelligence (AI) can be presented as a systematical search of possible outcomes with a task of finding some predefined goal or solution. One of the most frequently used approaches to problem-solving in AI is applying search algorithms [17]. In computer science, a search algorithm is considered an algorithm that is used for evaluating a set of all possible states and selecting the appropriate solution. To successfully implement a particular problem-solving system one or more search algorithms capable of performing the appropriate search are required. A set of all states of a particular problem is called state-space and can be denoted as:

$$S = \left\{ S_k : k \in \left\{ 1, \ldots, n \right\} \right\}, \quad (1)$$

The main goal of every algorithm is to make a transition from an initial state to a goal state which represents a solution of a given problem. These states can be marked as S_i and S_g (initial and goal states, consecutively), where:

$$S_i \in S \setminus \left\{ S_g \right\},\ i, g \in \left\{ 1, \ldots, n \right\},\ i \neq g. \quad (2)$$

The problem with designing such systems is the absence of formal procedures that can be used, given an initial and goal states, to determine a finite set of intermediate states. For this paper, let us define that every system possesses a finite set of actions, defined as:

$$A = \left\{ A_k : k \in \{1,\dots,m\} \right\}. \tag{3}$$

Transition from one state to another is achieved by performing a single action A_k from the set A:

$$A = \left\{ A_k : S \to S, k \in \{1,\dots,m\} \right\}, \tag{4}$$

$$A_k(S_l) = S_j,\ \ S_l, S_j \in S,\ \ l, j \in \{1,\dots,n\}. \tag{5}$$

This means that in every state S_l we can perform any action from the set A, but there could be a subset ($B \subset A$) of actions that do not cause state change:

$$(\forall A_k \in B) A_k(S_l) = S_l, \quad S_l \in S. \tag{6}$$

These actions are called illegal actions, and are not taken into consideration during state-space search. Legal actions are all actions that cause a transition from one state to another. It is possible to have less legal actions than remaining states; therefore, for every state S_k we define a set $S' \subseteq S$ which contains the states that can be achieved by performing legal actions:

$$\begin{gathered}(\forall S_k \in S)(\exists S' \subseteq S)(\exists A' \subseteq A) \\ (\forall A_k \in A') A_k(S_k) \in S'.\end{gathered} \tag{7}$$

State-space search can be complete or partial, depending on the selected search algorithm. Method of searching itself is simple and requires linear exploration of all possible states whilst testing the current state against the goal state. However, it can be resource demanding as in some cases it would be necessary to explore all possible system states. Solution to a given problem is represented as a sequence of actions that lead from the initial to the goal state. We can denote this sequence as a vector $\vec{s}$. Set of states between initial and goal states is not always unambiguous, because it is possible to have multiple paths between the two above mentioned states, and consequently multiple sequences of actions (solution vectors) leading to the goal state. Those sequences (vectors) are often different, based on their complexity and number of necessary actions (states between initial and goal state). In cases with multiple solutions, a single solution is chosen using arbitrary evaluation function *eval*(G), where G is a set of all possible solution vectors:

$$G = \left\{ \vec{s_j};\ j \in \{1,\dots,k\} \right\}, \tag{8}$$

where k is the number of possible solutions, and $k \geq 2$. Using different search algorithms can produce different solutions. Efficiency of a given solution is determined using different characteristics like termination, completeness, admissibility, time complexity and space complexity [18]. Every problem possesses a related problem domain, i.e. the area of expertise or application that needs to be examined to solve a problem. In the context of a robot and intelligent agent, problem domain can be represented as agent's environment.

3.1. Environment

Every agent (simulated or physical) is situated inside some particular environment and interacts with that environment. Type and intensity of an interaction depends on an agent's architecture and capabilities. The agent cannot be observed isolated from its environment. Therefore, properties of the agent environment should be defined. Russell and Norvig [19] identified different kinds of the environment: fully observable vs. partially observable, deterministic vs. stochastic, episodic vs. sequential, static vs. dynamic, discrete vs. continuous, and single agent vs. multi-agent. Identifying the environment in which the agent (robot) operates is an important step in valid problem representation. The proposed agent architecture is suitable for partially observable, stochastic, dynamic, sequential, discrete multi-agent environments.

3.2. Problem Representation

There are many different problems applicable to AI and search algorithms, but most of them can be placed in one of three main categories [18]: path-finding problems, two-player games and constraint-satisfaction problems. In real world situations, especially those involving robots, path-finding problems are the ones most frequently occurring [20], [21]. Furthermore, path-finding problems are closest to state-based model of the world, with each position representing a single state. One of the most common problems used for testing robots and their AI are maze traversal problems [22]. A maze is, in most cases, a two-dimensional lattice-like structure, consisting of finite number of identical sized cells [23]. The robot is placed in one empty cell (initial state) called starting cell, with the task of finding the way to the target cell in the maze (goal state). Not all cells are adjacent, so the robot must traverse through multiple cells to reach its goal. Some cells can contain obstacles, thus preventing a robot from passing through that cell. There can be multiple passageways to the target cell. This is equivalent to the above described state-space search. That leads to a conclusion that almost every problem can be represented as a maze, with each cell representing a single state of a problem space, and a solution to the problem is a sequence of actions. Each maze can be described and classified according to some of its characteristics like size, density, some obstacles, etc. [24]. Another advantage of using mazes for problem representation is that every maze can be easily converted to a graph and thus formally represented using graph theory [25].

For the purpose of this paper, we represent a maze as a directed weighted graph consisting of nodes and links. A starting cell represents root node. Other (non-root) graph nodes represent maze cells in which the agent (robot) must make a decision, e.g. a cell where a robot can turn left or continue forward. Additionally, dead end cells (agent has nowhere to go but back) are represented as leaf nodes. A target cell that marks the goal state of the robot is also considered a leaf node but is marked differently than dead end nodes. Figure 2 shows an example how a single maze is transformed in a directed weighted graph (tree).

Every link has its weight that represents an arbitrary cost of performing an action that induces state change from parent to a child node. This approach enables search algorithms to perform evaluation function on a given solution. In the context of mazes and path-finding approach, a cost of action is associated with distance, so it

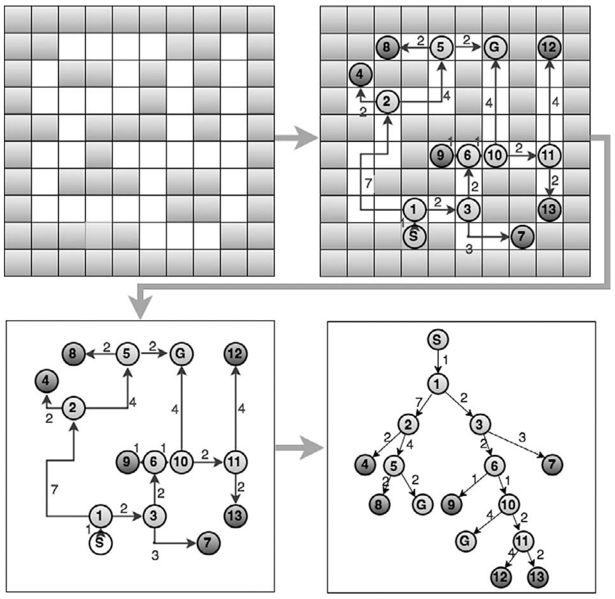

Figure 2. Converting maze to a directed weighted graph.

is important to mention most common ways of calculating distance. The Euclidean distance measurement is based on a real world measurement. It represents a shortest two-dimensional distance between two objects and can be calculated using a formula based on the Pythagorean Theorem. When using Euclidean distance in the path-finding problems, it is considered that the agent is located in the continuous environment. The Euclidean distance is used primarily in real-world applications where precise measurement is required. The Manhattan distance represents grid distance and is also referred to as taxicab distance, as it represents a taxi driving on a city (originally Manhattan) grid [26]. The Manhattan distance between two points is calculated by adding the absolute distance of each of the dimensions [27]. Agents using Manhattan distance are considered to be in a discrete environment using von Neumann type of neighborhood [28] i.e. an agent can move only in four directions (north, east, south, and west). The Manhattan distance is simple to understand, and it is used for agents or robots that have limited set of actions e.g. can only make 90 degrees turns. This also makes it appropriate for educational purposes [29]. The third distance metric is the Chebyshev distance, also known as the chessboard metric. The Chebyshev distance is calculated by taking the maximum of the dimension differences [27]. Similar to the Manhattan distance, agents that use this metric are in a discrete environment, but instead of the von Neumann neighborhood, the Moore neighborhood [30] is applied. Figure 3 shows described types of distance metrics. The described metrics are used for determining the distance between two points in the environment, and usually are the important part of the decision-making process, but to find a solution to a given problem, a search algorithm must be applied.

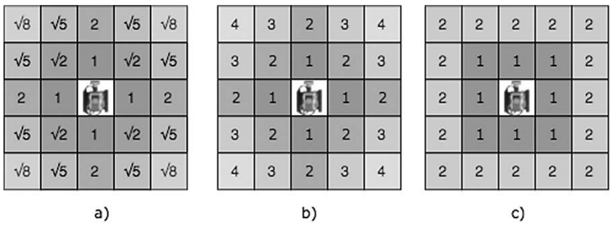

Figure 3. Different types of distance metrics: a) Euclidian; b) Manhattan; c) Chebyshevh.

3.3. Search Algorithms

Search algorithms can be classified regarding different characteristics. Most common classifications are: uninformed vs. informed, local vs. global, and systematic vs. stochastic. In the rest of the paper, search strategies (algorithms) will be divided according to the first mentioned classification - informed and uninformed (sometimes referred to as blind) [31]. Blind search strategy algorithms may only use information available in the problem definition: initial state, allowed operations on states and terminal state (they do not take into account location of the goal). Algorithms generate successors and differentiate goal (terminal) state from non-goal states. Strategies are distinguished by the order in which nodes are expanded. Directed search strategies use the additional (heuristic) information to determine which node will be expanded (whether one node is more promising than the other). Heuristic information is problem dependent (domain specific information) and must be known in advance. The heuristic function $h(n)$ takes a node n and returns the non -negative real number.That is an estimate of the path cost from node n to a goal node. It provides a way to inform the search about the direction to a goal. Using heuristics reduces problem dimension from exponential to polynomial. There is no guarantee that the goal will be found, or that solution would be optimal [32], [33]. These algorithms can be used by agents located in a fully or partially observable environments. The only condition, in the latter case, is that the part of the visible environment includes the goal node. Classical implementations of state search algorithms are well known and have a variety of applications in AI [34]. Taking into consideration that there are many different search algorithms, including their variations, it is necessary to design an intelligent agent architecture that will be able to support multiple search algorithms and apply them to problem-solving. A detailed description will be given in the following sections concerning agent architecture and experimental study.

4. Unibot Agent Architecture

An intelligent agent (robot) is composed of sensors, actuators, the mental model of its world and decision-making mechanism (system). An interaction with the environment is managed through sensors and actuators. Sensors enable the agent to perceive its surrounding (input data) while actuators enable it to affect the surroundings. The agent creates its mental model by processing raw perceptual data obtained through sensors. This mental model represents agent's reality and can be different from the physical ("real") world or even some other agent's mental model. This paper does not specifically address the problems of creating such model as it is too specific and greatly depends on the individual agent. A mental model is considered, from the agent's perspective, as a real environment.

The mental model is represented as a set of concepts and relations between those concepts. Concepts can be divided into concepts describing the perception of the world and concepts describing physical actions that the agent can perform.

The agent model for environment information representation can be described as ordered triple (objects, attributes, relations). Set O is a countable finite set of objects uniquely defined by the symbolic notation:

$$O = \{O_i : \mathrm{k} \in \{1, \ldots, n\}\}. \qquad (9)$$

For every *i*-th element of the set *O*, there exists a countable finite set of attributes At_i that uniquely describes that particular object:

$$At_i = \left\{ At_{ij} : \mathrm{j} \in \left\{1,\ldots,m\right\}\right\}, \tag{10}$$

and is implemented as artificial neural network (multi-layer perceptron). The function of this neural network is to establish a relation between object *Oi* and related set of attributes At_i. Similarly, the agent can be defined as ordered triple (*Sn*, *At*, *M*), where *Sn* denotes a countable finite set of sensors:

$$Sn = \left\{ Sn_n : n \in \left\{1,\ldots,i\right\}\right\}. \tag{11}$$

Element *Ac* denotes a countable finite set of actuators:

$$Ac = \left\{ Ac_n : n \in \left\{1,\ldots,j\right\}\right\}, \tag{12}$$

and *M* denotes a countable finite set of mental models:

$$M = \left\{ M_n : n \in \left\{1,\ldots,k\right\}\right\}, \tag{13}$$

where *i*, *j*, and *k* are the numbers of sensors, actuators and mental models, respectively. For example, in our experiment, physical representation of an agent was implemented using Lego Mindstorms robot that was equipped with two sensors (colour and ultrasound), two actuators (left and right motors) and possessed one mental model (maze):

$$\begin{aligned} Sn_{lego} &= \left\{ Sn_{1color}, Sn_{2ultrasound} \right\}, \\ Ac_{lego} &= \left\{ Ac_{1left\_} \right\} \\ M_{lego} &= \left\{ m_{1maze} \right\}. \end{aligned} \tag{14}$$

In the proposed agent architecture there are two distinct operating modes: a) problem exploration and b) problem-solving. Each of these modes is described in the following sections.

4.1. Problem Exploration

When presented with a problem, it cannot be presumed that the agent already possesses complete knowledge about that particular problem and related concepts, as well as its full state-space. By doing so, we would limit the agent usability to only a predefined set of problems. To conform to previously unknown problems and environments, the agent must possess a mechanism to learn and adapt to its current environment and given problem. The agent can acquire new knowledge using one of the principles described in previous sections. Unconditioned learning corresponds to the reactive part of the architecture, which produces a simple set of rules for the agent's primitive actions. Cognitive and social cognitive approaches are associated with proactive agent's behavior and knowledge acquisition. Origin (and type) of knowledge is not important, as it can be self-taught or transferred from another agent. To be able to learn from each other, agents must have at least one common characteristic that will be used as a foundation for knowledge exchange

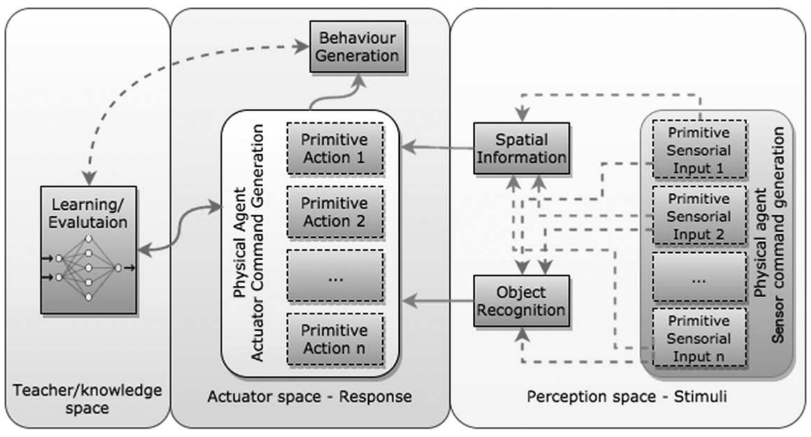
Behaviour Generation
Learning/ Evalutaion
Physical Agent Actuator Command Generation
Primitive Action 1
Primitive Action 2
...
Primitive Action n
Spatial Information
Object Recognition
Primitive Sensorial Input 1
Primitive Sensorial Input 2
...
Primitive Sensorial Input n
Physical agent Sensor command generation
Teacher/knowledge space
Actuator space - Response
Perception space - Stimuli

Figure 4. Learning process for the agent space navigation.

to satisfy the main requirement for successful knowledge transfer. The agent that is trying to learn from another agent needs the ability to receive the same type of sensorial input as the teacher agent possesses. The advantage of this approach is the possibility of the agent to learn from many different agents with which it shares some common sensorial input. Underlying mechanisms for learning and knowledge transfer are described in previous research [35], [36]. An overview of the cognitive learning process for the agent space navigation is presented in Figure 4.

Once the agent is familiar with a state-space of the current problem, informed search algorithms can be applied to find the desired solution. Depending on solution requirements, it is not always necessary for the agent to familiarize itself with the full state-space of a given problem, but only with a portion of it. The only requirement is that the portion contains at least one valid path from the initial to the goal state. Once the agent has acquired adequate knowledge, it can progress to the second internal state and find the desired solution within given constraints.

4.2. Problem Solving

In the second operating mode, the agent performs search within explored state-space and tries to find the best available solution for the given problem. The final solution may not always be the optimal one, depending on the algorithm used for the problem exploration, as well on the algorithm used for problem-solving. To guarantee an optimal solution, state-space exploration must be fully discovered (exhaustive search), and analyzed by one of optimal search algorithms (e.g. A*). However, searching for the optimal solution is not always the most advisable strategy. In some cases, a sub-optimal solution can be calculated faster and therefore is more desirable. Because of that, an agent must possess the ability to evaluate different solutions based on given heuristic. Agents that perform a single task, or any predetermined number of tasks, usually possess innate knowledge and always utilize the same search algorithm for the same particular problem. This type of behavior influences the agent's adaptability, which is one of the basic characteristics of an intelligent agent [37]. The agent's ability to operate in a dynamic environment is reduced, as well as its ability for solving previously unknown problems. The core idea of the proposed agent architecture is to define an agent that is capable of learning new tasks and actions to solve newly emerged problems. Simultaneously, the agent would, through practice, determine the most appropriate (not necessarily optimal) solution for the particular problem. Furthermore, by analyzing encountered problems, and classifying them into some groups, the agent could be trained to solve a particular set of problems, thus avoiding the need for learning how to solve every single distinct problem. Classification of problems is a separate topic and is too complex to be included in this paper. However, this is an important part of the proposed architecture and is the main focus of future research.

To meet those requirements, without affecting the agent's performance, it is necessary to design a system that allows simultaneous execution of several different problem-solving approaches (i.e. different algorithms). Such system can be implemented using the proposed multi-agent approach developed for decision making. Algorithms that manage to reach a satisfying solution under given constraints calculate the actions that an agent must perform to carry out that solution, along with the cost of performing that action. Due to an agent operating in a dynamic environment, even when the desirable solution is found, the agent must examine its environment after every action performed to detect any possible changes that would interfere with its planned actions. When the agent is presented with more than one solution, the proposed Multi-Agent Decision Support System (MADSS), depicted in Figure 5, uses the operating function to select the most appropriate solution, based on a given heuristic. Simultaneously, the system updates its mental model and knowledge by associating a problem set with a particular algorithm. A use case of the agent (Unibot) path finding problem is solved by using the Multi-Agent Decision Support System (MADSS) as presented in Figure 5.

The MADSS is a part of the proposed agent architecture enabling the agent to evaluate, choose and execute an action based on the operating function result. The complete proposed agent architecture is composed of two main parts: physical and mental. The physical part includes sensors and actuators while the mental

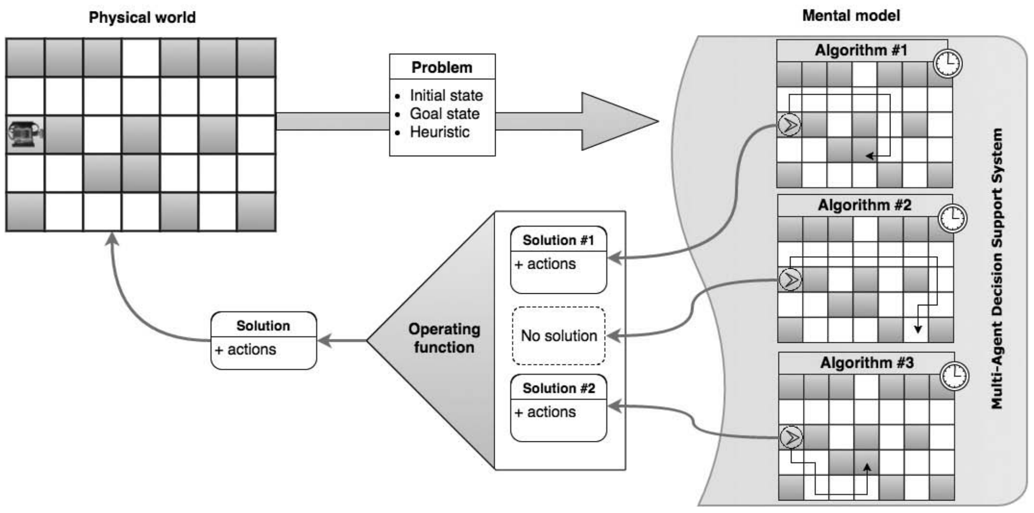

Figure 5. Overview of the Multi-agent Decision Support System used for calculate and evaluate multiple solutions for a given problem.

part consists of: the mental model, the learning module, the MADSS, the decision module and the available actions set.

Figure 6 gives a complete overview of the proposed agent architecture.

The research methodology and the experimental evaluation of the proposed agent architecture is presented in the next section.

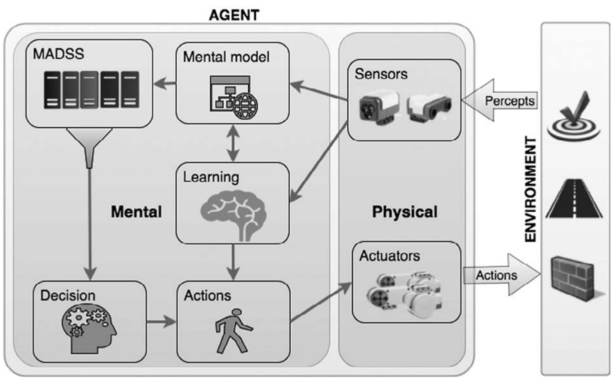

Figure 6. The proposed agent architecture.

5. Experimental Evaluation

The research methodology used in this research matches the pluralistic approach which combines an understanding of observed phenomena with the goal of understanding their cause, and empirical/analytical tradition [38]. Experiments based on understanding particular phenomena are described with a set of question/answer structures. Consequently, the goal of this experimental study is to provide answers to a set of research questions regarding the proposed intelligent agent architecture. The purpose of the research is better understanding of selecting appropriate problem-solving techniques in intelligent multi-agent systems.

Research questions are related to three distinctive phases of agent behavior: learning, exploration and solving.

- Learning phase – **Research Question I.**: How can an agent learn new concepts related to the environment, allowing him to move through the environment?
- Exploration phase – **Research Question II.**: How can an agent successfully navigate a maze, simultaneously searching for the goal and making the mental model of the environment?
- Solving phase – **Research Question III.**: How can an agent use the MADSS for selecting the most appropriate path from one cell in a maze to another?

5.1. Experimental setup and framework

For the purpose of the experiment, LEGO Mindstorms robot was used as a physical embodiment of Unibot intelligent agent, because the platform is widely used and modular [39]. The agent mental model (mapping and learning) and Multi-Agent Decision Support System were implemented using NetLogo – a multi-agent programmable modeling environment [40]. The simulated approach was chosen as it provided more flexibility, easier data collection, and better visualization of the mental model. Two-way real-time communication between simulations and the physical robot was implemented using the framework described in [41]. The proposed scenario was placing the Unibot in a maze with the goal of finding the shortest path from starting cell to the goal (green cell). The Unibot was constructed having two sensors (ultrasound and color) and two actuators controlling the wheels connected with tracks on each side of the robot. The environment of the robot was defined with three different concepts (wall, space, goal) and represented as a maze. The maze was designed with black and white tiles representing space, a green tile representing goal and boxes placed on the tiles were obstacles (walls). The complete software framework is available for download (please refer to http://mapmf.pmfst.unist.hr/heritage/Unibot_software.zip).

5.2. Research Question I

Initially, a robot did not possess the knowledge about any of those concepts regarding its environment, only its primitive set of actions, and was operating in the problem exploration mode described in 4.1. An agent activity during this phase includes training an artificial neural network (ANN) for each concept. Concepts are learned by obtaining perceptual stimuli from the robot's sensors in the physical world, which are forwarded to the simulation used to generate the data-set for the ANN associated with that particular concept. Each ANN consists of two input nodes, one hidden layer with two nodes and one output node. Recognized concepts, in the form of trained ANNs, represent a layer of the agent's knowledge related to its object recognition capability. The second layer of knowledge is learning allowed actions i.e. when the agent can perform some action. This knowledge is linked to object recognition, as the agent must be aware of its environment (current state) to determine all possible legal actions. Allowed actions are trained in the same way as concepts, the only difference being the source and number of input nodes. Source of these input nodes are the output nodes of ANNs from object recognition layer, and, accordingly, the number of input nodes depends on the number of concepts the agent has learned to recognize. Finally, individual ANNs from both layers were connected in one multi-layered ANN representing the agent's acquired knowledge. Figure 7 depicts different parts of the experiments conducted in the first phase. To test the social cognitive approach described in previous sections, the role of the teacher in this experimen-

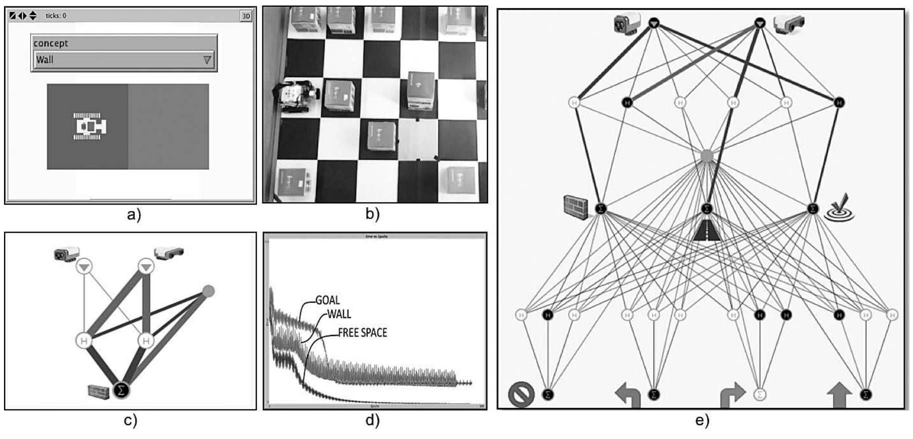

Figure 7. Learning phase: a) Simulation used for learning different concepts;
b) Physical embodiment of agent used for gathering sensors data;
c) Trained ANN for simple concept (wall);
d) Final results for learned concepts (number of errors over time);
e) Multilayered AN representing agent's knowledge.

tal phase was taken by both human and by the robot that was previously trained for navigating inside the same maze. This covered the first experimental question and demonstrated the agent learning ability.

5.3. Research question II

Having successfully concluded the first part of the experiments regarding learning, the second part was testing robot's application of gained knowledge through navigation across the maze while simultaneously building the mental model of its environment. This was carried out in both simulated and physical environments. Robot's sensory input was interpreted using previously described multi-layered ANN and recognized concepts were depicted in the simulation that represented robot's mental model of the world. In this case, the agent had perfect memory and was operating in a static environment. However, the agent cannot be self-aware and possess meta-knowledge about its environment, so it is necessary to, at least periodically, update the mental model by processing sensory input through object recognition module (trained ANN) and refresh its mental model accordingly. Aside from allowing the agent to operate in a dynamic environment, this can also be useful for correcting any prior potential errors in the agent's mental model. Figure 8 shows the robot in the real world and its mental representation of fully explored maze. Experiments were then conducted in additional three different maze configurations with the same type of elements (concepts). In all cases, the robot successfully managed to explore its environment and create an appropriate mental model that was later used in the third experimental phase. It should be emphasized that there is a difference between real-world and agent's mental models. This is best demonstrated by the different color of boxes. In the physical world, there are two types of boxes, with different height and color on top. However, the robot's sensors are not capable of distinguishing these differences causing the both types of boxes to be identified as the same type of object. This is not regarded as an error in the proposed architecture, but as a limitation of this particular robot. On the contrary, the proposed agent architecture is designed to overcome limitations of a single agent. By sharing knowledge between different agents and cooperating, it is possible to associate additional properties to any learned concept by using the same principles of the classical conditioning described in Section 2 and depicted in Figure 1.

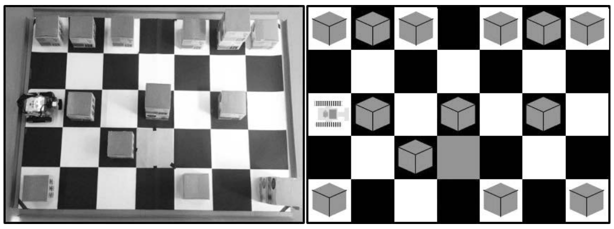
Figure 8. Physical and simulated representations of the agent in the maze.

5.4. Research question III

The last phase of the experimental study had a purpose to test the agent's Multi-Agent Decision Support System. This was done with a robot that successfully finished two previous phases and had explored the entire maze (exhaustive search). Fully explored mazes were better suitable for this part of the experimental study because they provided the robot with multiple solutions (paths to the goal) to choose amongst. For testing purposes, the robot was placed in a starting cell and presented with a problem of finding the best path to the goal. As already mentioned, search algorithms present a basis for path-finding problems and, conforming to the proposed architecture, they were required to run simultaneously with the task of choosing the desired solution assigned to the MADSS. Multi-agent implementations of classical search algorithms were used, both blind (depth-first, breadth-first, depth-limited, iterative deepening) and informed (greedy best-first, A*, hill-climbing, tabu). The Manhattan distance metric was used. Four physical maze configurations, used in the previous phase, where selected to test the performance of the robot's decision-making system. Each maze configuration possessed same unique trait compared to other three mazes and was classified based on the attributes defined by Bagnall and Zatuchna [24]. The attributes used in the experiment are:

$$avg.dist = \phi_m = \frac{\sum_{i=1}^{n} dist(c_i)}{n};$$

(n is a num. of cells, and c_i is an i-th cell)

$$size = s_m$$

$$density = \delta_m = \frac{o_m}{s_m} \tag{15}$$

$$obstacles = o_m = w_i + p + 0.5 \cdot w_s;$$

(w_i is an internal wall cell, p is num of partitions, and w_s is a surrounding wall cell)

Table 1 shows calculated maze characteristics.

Obtained solutions were evaluated using four different operating functions: path length, number of robots turns (rotations), total number of agents in simulation and number of simulation steps. Path length determines the number of cells that a robot must traverse through to reach the final cell and the solution with the minimal path is considered the best one. Second operating function emphasizes the importance of keeping the same heading while moving, so the solution with least number of turns is most desirable. A number of agents in a simulation matches the application's memory requirement; therefore, the third operating function evaluates a solution based on that factor. The fourth operating function is linked with the time required for calculating the solution and represented through the number of executed simulation steps. Table 2 shows detailed results for all four operating functions applied on the solutions provided by eight different algorithms in four different mazes. Analyzing obtained results, it can be noted that some algorithms produced

Table 1. Classification of the mazes used in the experimental phase.

Maze	Size	Density	Obstacles	Avg. dist. to goal	Num. of paths
#1	46	0.34	35	6.80	3
#2	77	0.44	64	5.76	14
#3	74	0.46	64	7.76	30
#4	92	0.46	84	9.67	25

Table 2. Detailed experimental results of the agent's MADSS system, used in four different maze configurations, with eight different search algorithms.

Maze	Operating function	Breadth-first	Depth-first	Limited	Iterative	Greedy	Dijsktra	A*	Tabu
#1	Path length	9	9	9	9	9	9	9	9
	Turns	2	2	2	2	2	2	2	2
	Agents	16	16	16	21	11	15	12	16
	Steps (simulation)	9	16	17	20	9	13	10	16
#2	Path length	10	22	22	10	10	10	10	n/a
	Turns	4	6	6	4	4	4	4	n/a
	Agents	33	32	32	106	16	28	24	n/a
	Steps (simulation)	10	27	27	88	12	22	18	n/a
#3	Path length	12	16	16	16	12	12	12	16
	Turns	4	6	6	6	4	4	4	6
	Agents	90	24	24	119	18	80	39	24
	Steps (simulation)	12	20	20	91	12	65	28	20
#4	Path length	14	18	18	18	18	14	14	18
	Turns	6	6	6	6	8	6	6	6
	Agents	76	27	27	179	32	68	42	27
	Steps (simulation)	14	21	21	148	25	61	35	23

identical results according to one operating function (heuristic), but opposite results when evaluated by some other operating function. This leads to the conclusion that there is no universally "best" algorithm. Instead, the algorithm selection depends on the chosen problem. This confirms the importance of problem classification and necessity of training agents to identify a type of problem and apply adequate solution technique.

6. Conclusion and Future Work

The above described experiments were conducted with the goal of responding to the given research questions. To conduct experiments, five different simulations were developed, each constituting a part of the proposed Unibot architecture. Additionally, a psychical robot was constructed, along with the associated maze, to demonstrate the physical embodiment of the agent and to test the relation between physical and simulated parts of the architecture. The first experimental phase demonstrated that the agent can use a multi-layered artificial neural network to learn new concepts related to its environment. The second experimental phase confirmed the validity of the first phase by successfully using learned concepts for maze navigation and demonstrated the principles of creating and refreshing the agent's mental model. The third phase proved the agent's ability to use Multi -Agent Decision Support System (MADSS) to select the most appropriate solution for different problem types. Altogether, this proved the validity of the proposed agent architecture. As previously stated, future work will be focused on the topic of problem classification, to fully benefit from the proposed agent architecture, primarily the MADSS. One drawback of the conducted experiments was small number of tested maze configurations. This was caused by the necessity of physically constructing every maze, which was the main limiting factor for additional maze configuration. However, conducted experiments proved a correlation between the physical and the simulated environments, which enables further experiments to be conducted solely in the simulated environment, thus removing the aforementioned limitation. Experimental studies also drew attention to the differences between simulated and physical implementations of the agent. Although these differences are well known in the fields of robotics and artificial intelligence [42], future work is planned on implementing and testing a localization subsystem inside the proposed architecture, with the main goal of further extending the adaptability and universality of the architecture.

References

[1] W. Jordaan, "*Man in Context*", Lexicon, Johannesburg, 2007.

[2] P. Wang, "On the Working Definition of Intelligence", Technical report, Citeseer, 1995.

[3] M. Z. Altan, "Intelligence Reframed: Multiple Intelligences for the 21st Century.: Howard Gardner", TESOL Quarterly, vol. 35, no. 1, pp. 204–205, 2001.

[4] R. Pfeifer and Christian Scheier, "Understanding Intelligence", MIT press, 2001.

[5] T. Fong et al., "A Survey of Socially Interactive robots", Robotics and Autonomous Systems, vol. 42, no. 3–4, pp. 143–166, 2003. http://dx.doi.org/10.1016/S0921-8890(02)00372-X

[6] S. Pan and Q. Yang, "A Survey on Transfer Learning", IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, 2010. http://dx.doi.org/ 10.1109/TKDE.2009.191

[7] L. Tai and M. Liu, "Deep-Learning in Mobile Robotics – from Perception to Control Systems: A Survey on Why and Why Not", arXiv preprint arXiv:1612.07139, 2016.

[8] Y. Bengio, "Learning Deep Architectures for AI", Foundations and trends® in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009. http://doi.org/10.1561/2200000006

[9] N. Roll-Hansen, "Why the Distinction Between Basic (Theoretical) and Applied (Practical) Research is Important in the Politics of Science", London School of Economics and Political Science, Contingency and Dissent in Science Project, 2009.

[10] B. Argall *et al.*, "A Survey of Robot Learning from Demonstration", *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009. http://dx.doi.org/ 10.1016/j.robot.2008.10.024

[11] B. Blumberg *et al.*, "Integrated Learning for Interactive Synthetic Characters", *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 417–426, 2002. http://dx.doi.org/ 10.1145/566654.566597

[12] M. M. Veloso *et al.*, "Focus: a Generalized Method for Object Discovery for Robots that Observe and Interact with Humans", in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, ACM, pp. 102–109, 2006.

[13] C. Breazeal *et al.*, "Tutelage and Colaboration for Humanoid Robots", *International Journal of Humanoid Robotics*, vol. 1, no. 2, pp. 315–348, 2004.
http://dx.doi.org/ 10.1142/S0219843604000150

[14] J. Bransford *et al.*, "Learning Theories and Education: Toward a Decade of Synergy", Handbook of educational psychology, 2nd Ed., pp. 209–244, 2006.
http://dx.doi.org/ 10.4324/9780203874790.ch10

[15] B. Lahey, "Psychology: an Introduction", McGraw-Hill Higher Education, New York, 2012.

[16] A. C. Murillo *et al.*, "A Practical Mobile Robotics Engineering Course Using LEGO Mindstorms", in International Conference on Research and Education in Robotics, Springer Berlin Heidelberg, pp. 221–235, 2011.
http://dx.doi.org/10.1007/978-3-642-21975-7_20

[17] R. H. Landau *et al.*, "Computational Physics: Problem Solving with Python", John Wiley & Sons, 2015.

[18] R. E. Korf, "Artificial Intelligence Search Algorithms", in Algorithms and Theory of Computation Handbook, chapter Artificial Intelligence Search Algorithms, (M. J. Atallah and M. Blanton, Eds.), Chapman & Hall/CRC, pp. 22–22, 2010.
http://dx.doi.org/10.1201/9781420049503-c37

[19] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach", Prentice Hall, 2003.

[20] Z. Ni and H. He, "Heuristic Dynamic Programming with Internal Goal Representation", *Soft Computing*, vol. 17, no. 11, pp. 2101–2108, 2013.
http://dx.doi.org/10.1007/s00500-013-1112-9

[21] X. Pang and P. Werbos, "Neural Network Design for *J* Function Approximation in Dynamic Programming", arXiv preprint adap-org/9806001, 1998.

[22] S. Yadav *et al.*, "The Maze Problem Solved by Micro Mouse", *International Journal of Engineering and Advanced Technology (IJEAT)*, pp. 2249–8958, 2012.

[23] M. Foltin, "Automated Maze Generation and Human Interaction", Brno: Masaryk University Faculty Of Informatics, 2011.

[24] A. Bagnall and Z. Zatuchna, "Foundations of Learning Classifier Systems, chapter On the Classification of Maze Problems", Springer, Berlin, Heidelberg, pp. 305–316, 2005.

[25] A. Levitin and S. Mukherjee, "Introduction to the Design & Analysis of Algorithms", Addison-Wesley Reading, MA, 2003.

[26] E. F. Krause, "Taxicab geometry: An Adventure in Non-Euclidean Geometry", Courier Corporation, 2012.

[27] M. M. Deza and E. Deza, "Encyclopedia of distances", Springer, 2009.

[28] C. M. Macal and M. J. North, "Agent-Based Modeling and Simulation", in *Winter simulation conference*, Winter Simulation Conference, pp. 86–98, 2009.

[29] G. Zaharija *et al.*, "Use of Robots and Tangible Programming for Informal Computer Science Introduction", *Procedia – Social and Behavioral Sciences*, vol. 174, pp. 3878-3884, 2015.
http://dx.doi.org/10.1016/j.sbspro.2015.01.1128

[30] A. Laurie and N. Jaggi, "Role of 'Vision' in Neighbourhood Racial Segregation: A Variant of the Schelling Segregation Model", *Urban Studies*, vol. 40, no. 13, pp. 2687–2704, 2003.
http://dx.doi.org/10.1080/0042098032000146849

[31] D. McShan *et al.*, "PathMiner: Predicting Metabolic Pathways by Heuristic Search", *Bioinformatics*, vol. 19, no. 13, pp. 1692–1698, 2003.
http://dx.doi.org/ 10.1093/bioinformatics/btg217

[32] R. K. Ahuja *et al.*, *Network flowsprentice-hall*, Englewood Cliffs, NJ, 1993.

[33] N. I. N. G. Ai-Bing *et al.*, "Solving Degree-Constrained Minimum Spanning Tree with a New Algorithm and Engineering" in *IEEE International Conference on Managament Science and Engineering ICMSE 2007*, pp. 381–386, 2007.
http://dx.doi.org/ 10.1109/ICMSE.2007.4421877

[34] W. Zhang, "State-Space Search: Algorithms, Complexity, Extensions, and Applications", Springer Science & Business Media, 1999.

[35] S. Mladenovic *et al.*, "An Approach to Universal Interaction on the Case of Knowledge Transfer", in *Universal Access in Human-Computer Interaction User and Context Diversity*, Springer, pp. 604–613, 2013.
http://dx.doi.org/10.1007/978-3-642-39191-0_65

[36] G. Zaharija *et al.*, "Learning from Each Other: An Agent Based Approach", in *Universal Access in Human-Computer Interaction Universal Access to Information and Knowledge*, Springer, pp. 475–486. 2014.
http://dx.doi.org/10.1007/978-3-319-07440-5_44

[37] D. Tapia *et al.*, "Agents and Ambient Intelligence: Case Studies", *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, no. 2, pp. 85–93, 2009.
http://dx.doi.org/ 10.1007/s12652-009-0006-2

[38] J. Mingers, "Combining IS Research Methods: Towards a Pluralist Methodology", *Information*

Systems Research, vol. 12, no. 3, pp. 240–259, 2001.
http://dx.doi.org/ 10.1287/isre.12.3.240.9709

[39] E. Danahy *et al.*, "Lego-Based Robotics in Higher Education: 15 Years of Student Creativity", *International Journal of Advanced Robotic Systems*, vol. 11, no. 2, 2014.
http://doi.org/10.5772/58249

[40] U. Wilensky, "Netlogo, Center for Connected Learning and Computer-Based Modeling", Northwestern University, Evanston, IL, 1999. Available http://ccl.northwestern.edu/netlogo/

[41] G. Zaharija *et al.*, "Learn–Lego Robot and Netlogo", in *International Scientific and Professional Conference (CIET2014)*, pp. 209–218, 2014.

[42] O. Miglino *et al.*, "Evolving Mobile Robots in Simulated and Real Environments", *Artificial Life*, vol. 2, no. 4, pp. 417–434, 1995.
http://dx.doi.org/ 10.1162/artl.1995.2.417

Received: March 2017
Revised: March 2017
Accepted: March 2017

Contact addresses:

Goran Zaharija
Faculty of Science
University of Split
Split, Croatia
e-mail: goran.zaharija@pmfst.hr

Saša Mladenović
Faculty of Science
University of Split
Split, Croatia
e-mail: sasa.mladenovic@pmfst.hr

Lada Maleš
Faculty of Humanities and Social Sciences
University of Split
Split, Croatia
e-mail: lada.males@ffst.hr

GORAN ZAHARIJA received his BSc and MSc degrees from the University of Split, Faculty of Electrical Engineering, Mechanical Engineering, and Naval Architecture, in 2008 and 2010, respectively. He is a research assistant at the Computer Science Department of the Faculty of Science, University of Split. He teaches several undergraduate courses in programming and computer architecture. He is currently working on his PhD research in the field of artificial intelligence. His other scientific interests include machine learning, artificial neural networks and multi-agent systems.

SAŠA MLADENOVIĆ is an assistant professor of computer science at the Computer Science Department of the Faculty of Science, University of Split. He received his PhD degree in 2011. from the Faculty of Electrical Engineering, Mechanical Engineering, and Naval Architecture, University of Split. From 1999 to 2006 he was acting as Technical Manager of the Toll collection system department of Ecsat, Croatia, member of CS group designer, integrator and operator of mission critical systems, France. His research interests include interoperability, intelligent technologies like ontology and multi-agent systems, computer science education and especially engineering applications of intelligent technologies. He has been IEEE member since 1996.

LADA MALEŠ received her BSc degree in 1996 from the University of Split, Faculty of Electrical Engineering, Mechanical Engineering, and Naval Architecture, the MSc degree in 2002 and the PhD degree in 2017 all from the University of Zagreb, Faculty of Electrical Engineering and Computing. From 1997 to 2004 she worked at the University of Split, Faculty of Science as a research assistant. Since 2004 she has been working at the University of Split, Faculty of Humanities and Social Sciences as senior lecturer. Her research interests are in the areas of spatial and temporal knowledge representation and reasoning, and multi-agent systems. She is a member of the IEEE.