# Improve Path Planning Performance by Monitoring Human Decisions

**CheeChian Cheng, Henry Hexmoor**
Computer Science Department, Southern Illinois University Carbondale, Carbondale IL 62901, USA

**Abstract -** *Behavioral based control is usually based on a set of previously codified rules to which the system must adhere. Adaptive behavioral modeling involves learning new domain knowledge and applying the newly learned knowledge to the situation while still complying with the rules in the process of performing the task. This paper proposes a solution to improve path planning by observing human decisions in path planning and chunking pieces that are superior to stored paths.*

**Keywords:** Artificial intelligence, machine learning, path planning, chunking, adaptive behavioral modeling

## 1 Introduction

Traditionally, machine learning was viewed as a branch of artificial intelligence, but starting in 2000, many researchers began to treat machine learning as a separate field by itself. According to Patrick Langley, this is a great loss to both fields [15]. Chunking is a psychological memory mechanism to perform memory tasks by reproducing smaller pieces of information [19]. Miller argued that most people can remember about seven (7) chunks in their short term memory (STM). By using chunking, an agent will be able to learn and apply his past experience (memory based learning) to solve a given problem.

In the gaming industry, most games, especially Role Playing Games (RPG), require a player to perform a series of steps to complete the level. The player (or agent) will need to explore the game and learn from exploration before exploiting the game. Hence, there are expert players who can play the game well based on his/her knowledge (or past experience) in the game while novice players spend most of their time exploring the game, learning and acquiring knowledge for most of the play time. One of the goals of modeling agent's decision is to create an intelligent agent (i.e., computer player) that plays like a human – expert player.

## 2 Related Work

### 2.1 Chunking

Chunking often refers to a process of forming a group of information that will act like a long-term memory to the agent [10]. The agent will perform memory tasks by recalling chunks of information and reconstructing them into vital information that helps them to perform various tasks. Chunking is often considered as a mature technology and has been widely used in various areas (especially in machine learning) including phrase (i.e. part of speech) recognition, general problem solving approaches as used in SOAR, chunking with support vector machine and other areas. Grover and Tobin have proposed a rule-based chunking approach to implement chunking and at the same time, provided the property of reusability [10]. They argued that by using XML as the basis for chunking rules, it would provide some degree of flexibility with respect to reusability that made it very appealing in machine learning.

### 2.2 Cognitive Maps

Cognitive Map is credited to Edward Tolman when he started his research on rats' behavior in exploring mazes [35]. Tolman argued that the rats must have known something about the maze's topography and the rats must have formed some kind of cognitive map during the training. Later, Tolman et al. argued that rats often take the shortest path to the goal box whenever possible and coined this behavior as a type of spatial insight [36]. The rats were able to construct and accumulate spatial knowledge about the maze, which allowed them to visualize the maze during the testing condition and figure out the right path to reach the goal box. By converting each junction of these paths into a state (of state diagram) and applying proper reinforcement (reward or punishment), a route can be derived by applying various algorithms, including reinforcement learning and Markov Decision Process, to simulate the rats' behavior.

### 2.3 Modeling of Ecology Movement

Researchers in ecology proposed a framework for fitting multiple random walks to animal movement paths by modeling the animals' movement [21]. By using the animal's step lengths and turning angles as features, the proposed framework will be able to classify the behavioral states of that animal by any machine learning approach, e.g. Support Machine Vector, K-Nearest Neighbor, Artificial Neural Network, etc.

### 2.4 Belief, Desire, and Intention (BDI) Model

Belief, Desire (Goal) and Intention model the behavior of a rational software agent by imbuing human practical

reasoning [8]. Belief (of an agent) refers to the state of the environment or the facts which may include forward inference rules that will lead to new beliefs (which can be implemented by any rule-based engine like Jess, CLIPS and Drools). Instead of using the term knowledge like most of the expert systems do, the agenda or output of the rule-based engine (the newly inferred belief) might not be true, in contrast to expert systems where the agenda of the rule-based engine is knowledge. Desire represents the goal which the agent wishes to accomplish. An agent can have a set of desires based on its current beliefs. Intention represents the action to engage the desire, in other words, what the agent has decided to do by choosing from the desire sets. In this framework, one of the limitations is the ability to learn from the agent's past decisions and adapt to a new environment.

## 2.5 Subsumption Architecture

Subsumption is an approach to prioritize behaviors into a hierarchy and output the behavioral action based on the agent's input sensors [1]. Each layer in the hierarchy describes a behavior or an action that the agent needs to take. These behaviors constantly compete among themselves based on its current inputs and its goals (intention), thus producing an output as the behavioral-based action. The main disadvantage of using this architecture is that the behavior-based action in each layer of the hierarchy tends to conflict with each other when there are too many layers. Another more democratic approach of behavioral competition is to use the voting system where each behavior is given a voting count based on its degree of relevancy towards its current inputs and goal [31].

## 2.6 Artificial Neural Network

Artificial Neural Network (ANN) consists of neurons, synaptic weight among neurons, and threshold/activation function; it is mimicking the neurophysiologic brain model with the ability to learn and work with inaccurate information as input and subsequently producing a set of outputs [27]. There are various types of ANNs; they are single perceptron, feed forward back propagation, Hopfield network, Bidirectional Associative Memory (BAM), Hebbian learning, Self-Organizing Map, hybrids of fuzzy and ANN, evolutionary ANN (Genetic Algorithm + ANN), and etc.

In single perceptron and feed forward back propagation ANN, ANN learns by repeatedly adjusting its synaptic weight between neurons until it reaches satisfactory results [38]. This process is called training in the field of ANN.

## 2.7 Genetic Algorithm

Genetic Algorithm (GA) is a stochastic search algorithm based on Neo-Darwinian natural evolution theory [23]. The key concepts in GA are reproduction, mutation, competition, and selection [20]. In Neo-Darwinian natural selection theory, the fitter organism will have a higher chance of being selected for reproduction. This, in turn, will produce a better next generation (fitter next generation) and if done repeatedly for a number of generations, it should reach maxima in term of fitness. However, this might be a local maxima, thus mutation is introduced so that the search can reach global maxima. Learning in GA can be perceived as producing offspring that has better fitness over generations.

## 2.8 Reinforcement Learning

Reinforcement Learning is an approach of machine learning that lies in between supervised and unsupervised learning. It was pioneered by Richard S. Sutton and Andrew G. Barto. This approach will observe two results from every lesson: win or loss; a "reward" will be awarded to the winning result and "reinforcement" to the losing result. This approach is sometimes named as "trial-and-error learning" besides being coined as "Reinforcement learning" [26]. In short, reinforcement learning is about how to traverse through various states and finally get rewarded, which is identical to, for example, a pet (animal) is given rewards (treat/food) when it had successfully performed a series of actions.

One of the learning approaches for reinforcement learning is Q-learning [39]. Q-learning learns the policy by assigning random numbers to every possible action in each state and readjusting these numbers based on its training or experience. This process continues until each action in the state points towards the goal state [26].

## 2.9 Problem Space

Problem space is the fundamental organizational unit of all human goal-oriented symbolic activity [24]. As defined by Newell, a problem space consists of symbolic structures and operators, where symbolic structures may represent the states of the space while operators take a state as input and produce a new state as output. A problem in problem spaces consists of a set of initial states, a set of goal states, and a set of path constraints. Problem space is mental constructs and may lead to external actions.

## 2.10 Production System

A production system refers to a system where it produces products based on a given set of input (facts) as shown in Figure 1. Facts are stored in working memory (WM) and serve as the input to the production system and the products of the production system are the output.

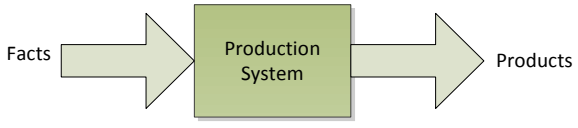


Figure 1. A Production System

A typical rule-based production system is made up of various rules, usually drawn out by domain expert, which will translate known facts to useful outputs. A rule is made up with antecedent (condition) and consequent (action or output) [23] and sometimes, the antecedent part of the rule is referred as Left-Hand-Side (LHS) while the consequent is referred as Right Hand Side (RHS) of the rules. These notations are interchangeable but LHS and RHS are commonly used throughout the A.I. community. One important difference between production system and computer program is production system matches the rules in parallel and continuous processing (analogy to a circuit board logic gates) while computer program does it sequentially.

### 2.11 RETE Algorithm

Charles Forgy proposed a fast algorithm for the many pattern/many object pattern match problem by building a network of nodes that processes the delta-change of working memory and produces conflict set [7]. Working memory is the content of data operated on by the productions, which is the content of variable for the conditional test or Left Hand Side (LHS). Conflict set is the output of the matching process of the productions. The conflict set will then be past to conflict resolution to select one production (rule) with a satisfied test condition. RETE algorithm will translate the production rules into a network (tree like) with nodes that hold memory [7].

### 2.12 Shortest Path Algorithm

In 1956, a Dutch computer scientist came up with an algorithm to find shortest path in a graph (Dijkstra's algorithm) and later in 1968, Hart, Nilsson, and Raphael came up with A-star (A*), a variant of Dijkstra's algorithm, which is more efficient in finding minimum-cost paths in a graph. Unlike Dijkstra's algorithm, A* uses heuristic and best-first search method to find the minimum-cost path by skipping most of the evaluation process in other nodes that will not produce a minimum-cost sub-graph [11]. A* is widely used in game industry due to its simplicity in implementation, low time and space complexity in finding shortest path from one point to another. However, A* will not work with negative cost edge and requires a heuristic function to work accurately; otherwise, it will not produce the shortest path (if heuristic function overestimated the distant to the destination).

## 3 Proposed Approach

The basic idea is to use a rule-based engine, preferably RETE based engine to produce output based on current states of the agent, current states of the environment and the chunking between two points. Each chunk is given a score based on a scoring function. A typical scoring function might be using the distance and elapse time between two points.

The system will learn from a human agent by comparing the chunk's score in its memory (rules). If human decision has a better score than its corresponding rules (based on current states of the agent and the environment), a new rule will be constructed and the old rule in the system will be replaced as shown in Figure 2.

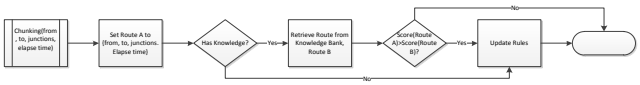


Figure 2. System Flow Chart

## 4 Implementation

A testbed was built for agent to run errand on every round of the game. An agent will be given k errands. The agent is free to choose any location to start with and the agent has to return to his/her base upon finishing the errand. Performance is measured by the distant travelled and time taken to complete the errand. This is identical to modeling human player or bot (agent) playing video or computer game and solving the riddle based on the human player's past experience. In each round of the game, the time and distance travelled will be recorded. Upon starting of the new round/game, the system will suggest to the player the route he/she should take based on his/her past experience. Of course, the player has the choice whether to follow the proposed route or exploring new route. After each exploration, a new chunk might be created if it has a better score (time and distance) and will be stored as a new knowledge. Average speed between two points is used as the scoring function in the implementation.

The testbed consists of Manhattan grid with several points of interest that the agent must visit. In each tour, the agent will travel through all the points of interest based on the agent's current state (urgency and service level) and the environment factors such as weather and time. During implementation, a RETE Rule-Based Engine was used for pattern matching. The agent's service level, urgency, weather and the time of the day were used as states and served as part of the chunking. However, in all machine learning approaches, they are constrained by boundaries of the modeled environment (in this case, they are service level, urgency, weather and the time of the day). Often, humans learn parameters of the environment that would not perceive pertinent prior to that point. No known machine learning technique is capable of ascertaining new environmental parameters.

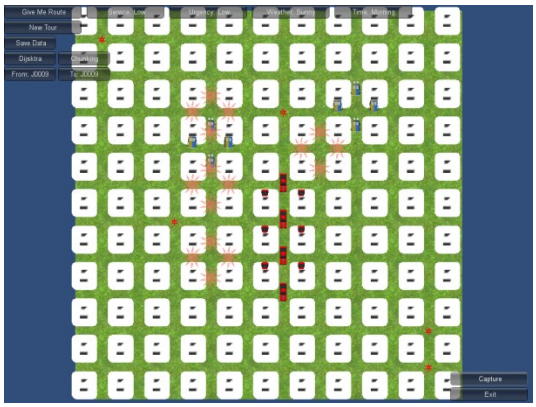

Figure 3. Implementation

At any given time, an agent will be given a series of errand, which forms a tour. In this case, an agent refers to the human. The agent always begins its tour at a point and ends the tour at the same point. Agent has the flexibility to choose which point to run his/her errand first. Upon choosing the destination point, the agent will recall from its long term memory for the route it has taken before under the same conditions (states), i.e., service, urgency, weather and time. If the agent has never run that route before, Dijkstra's algorithm will be used for finding the shortest path to the destination. The route will serve as a memory chunk and will be stored into the memory by adding a production rule into the agent's knowledge based system. A chunk can be defined as follows.

$$Chunk = \{ (from, to, service, urgency, weather, time), junction\ names, elapse\ time \}$$

In this case, the production rule will serve as a long term memory for the agent's knowledge. Rete engine is used for production by feeding in the Left Hand Side (LHS), i.e., from, to, service, urgency, weather and time states into the production system before producing the route as an output.

# 5 Conclusions

In conclusion, by improving agent's path planning with the approach mentioned in above, we have developed an approach to enhance agent route planning. Chunks of knowledge that represent the agent's past experience and knowledge are being reused for successive route planning. A suitable scoring function (distance between two points was used in the implementation) was used for determining whether a new chunk of information should supersede the previous chunk of knowledge. Our approach has considered nuances in routes that are meaningful for human route planners, and this has added a degree of cognitive reasoning and realism to route planning as well.

More effects of various categories of nuances on route planning should be considered, for example danger, outlook/exploration, pit and refuel stops, marker dropping and removal, and repair. The model will react and behave better with human-like cognitive reasoning when more categories of nuances are identified. However, the proposed approach does not analyze the cognitive reasoning of why the human agent makes that decision; it simply duplicates the decision made by human to mimic and imbue the cognitive reasoning. This is not suitable for systems such as diagnosis expert system in the medical field.

# 6 References


[1] Arkin, R. C. "Behavior-Based Robotics". The MIT Press, 1998.

[2] Bellman, R. E. "A Markov decision process"; Journal of Mathematical, 679-684, 1957.

[3] Braitenberg, V. "Vehicles: Experiments in Synthetic Psychology". The MIT Press, 1986.

[4] Derbinsky, N., Laird, J. E. "Extending Soar with Dissociated Symbolic Memories"; Proceedings of the Symposium on Human Memory for Artificial Agents 36th, AISB, 2010.

[5] Douglass, S. A., Ball, J., Rodgers, S. M. "Large declarative memories in ACT-R"; 9th International Conference of Cognitive Modeling, , Manchester, United Kingdom, 234, 2009.

[6] Finnsson, H., Bjornsson, Y. "Learning Simulation Control in General Game-Playing Agents"; National Conference on Artificial Intelligence, Atlanta, Georgia, 2010.

[7] Forgy, C. L. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem"; Artificial Intelligence, 19(1), 17-37, 1982.

[8] Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M. "The Belief-Desire-Intention Model of Agency". Springer, 1999.

[9] Gross, J. L., Yellen, J. "Graph Theory and Its Application". Chapman & Hall/CRC, 2006.

[10] Grover, C., Tobin, R. "Rule-Based chunking and reusability"; Proceedings of LREC 2006, 873-878, 2006.

[11] Hart, P. E., Nilsson, N. J., Raphael, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths"; Artificial Intelligence Group of the Applied Physics Laboratory, 4(2), 100-107, 1968.

[12] Hu, Q., Yue, W. "Markov Decision Processes With Their Applications". Springer, 2008.

[13] Kleinberg, J., Tardos, E. "Algorithm Design". Pearson Education Inc, 2005.

[14] Koning, K. D., Bredeweg, B., Breuker, J., Wielinga, B. “Model-based reasoning about learner behavior”; Artificial Intelligence, 117, 173–229, 2000.

[15] Langley, P. “The changing science of machine learning”; Machine Learning, 82(3), 275-279, 2011.

[16] Larid, J. E., Rosenbloom, P. S., Newell, A. “Chunking in Soar: The anatomy of a general learning mechanism”; Machine Learning, 1, 11-46, 1986.

[17] Loh, C. “Researching and Developing Serious Games as Interactive Learning Instructions”; International Journal of Gaming and Computer Mediated Simulations, 1(4), 1-19, 2009.

[18] McDermott, J., Newell, A., Moore, J. “The efficiency of certain production system implementations”; SIGART Bull, 63, 1977.

[19] Miller, G. A. “The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information”; The Psychological Review, 63, 81-97, 1956.

[20] Mitchell, M. “An Introduction to Genetic Algorithms”. The MIT Press, 1998.

[21] Morales, J. M., Haydon, D. T., Frai, J. “Extracting more out of relocation data: Building movement models as mixtures of random walks”; Ecology, 85(9), 2436-2445, 2004.

[22] Nayak, P., Gupta, A., Rosenbloom, P. “Comparison of the RETE and TREAT production matchers for Soar (a summary)”; The Soar papers (The MIT Press), 1993.

[23] Negnevitsky, M. “Artificial Intelligence: A Guide to Intelligent Systems”. Addison-Wesley, 2005.

[24] Newell, A. “Reasoning, problem solving and decision processes: The problem space as a fundamental category”; Attention and performance VIII, 1979.

[25] Nilsson, N. J. “Aritificial Intelligent: A New Synthesis”. Morgan Kaufmann Publishers Inc., 1998.

[26] Nilsson, N. J. “The Quest For Artificial Intelligence: A history of ideas and achievements”. Cambridge University Press, 2010.

[27] Rabuñal, J. R., Dorado, J. “Artificial Neural Networks In Real-Life Applications”. Idea Group Inc., 2006.

[28] Reid, A., Staddon, J. “A Dynamic Route Finder for the Cognitive Map”; Psychological Review, 105(3), 585-601, 1998.

[29] Resnick, M. “Turtles, Termites, and Traffic Jams: Explorations in massively parallel microworlds”. The MIT Press, 1997.

[30] Rosenblatt, J. K. “DAMN: A Distributed Architecture for”; Experimental & Theoretical Artificial Intelligence 9(2-3), 339-360, 1997.

[31] Rosenblatt, J. K., Payton, D. W. “A Fine-Grained Alternative to the Subsumption”; IEEE/INNS International Joint 2, 317-324, 1989.

[32] Roth, B. H. “An architecture for adaptive intelligent systems”; Artificial Intelligence, 72(1-2), 329-365, 1995.

[33] Scholkpof, B., Mallot, H. A. “View-Based Cognitive Mapping and Path Planning”; Adaptive Behavior, 3, 331-348, 1995.

[34] Sutton, R. S., Barto, G. A. “Reinforcement Learning: An Introduction”. The MIT Press, 1998.

[35] Tolman, E. C., Honzik, C. H. “"Insight" in rats”; University of California Publications in Psychology, 4, 215-232, 1930.

[36] Tolman, E. C., Ritchie, B. F., Kalish, D. “Studies in spatial learning: I. Orientation and the short-cut”; Journal of Experimental Psychology, 36, 13-24, 1946.

[37] Turing, A. M. “Computing Machinery and Intelligence”; Mind, 59(236), 433-460, 1950.

[38] Veelenturf, L. “Analysis and Application of Artificial Neural Networks”. Prentice Hall International (UK) Limited, 1995.

[39] Watkins, C. J. “Learning from Delayed Rewards, Ph.D. thesis”. Cambridge, England: Cambridge University, 1989