IMPLEMENTATION OF THE Q-LEARNING ALGORITHM FOR OPTIMISING JUDICIAL ADVISORY EXPERT SYSTEM (JAES)

Mr. Vishal Barot
Research Associate
Wolfson School of Mechanical and Manufacturing Engineering
Loughborough University, Loughborough, United Kingdom
V.Barot@lboro.ac.uk

Dr. Jenny Carter
Principal Lecturer
Centre for Computational Intelligence, School of Computing
De Montfort University, Leicester, United Kingdom
jennyc@dmu.ac.uk

Abstract
The work presented in this paper is an implementation of the Q-learning algorithm to optimise the rule execution process of Judicial Advisory Expert System (JAES). JAES covers the Sale of Goods Act (SGA) sections 16 to 20, UK. Addition of further sections to JAES led to its performance degradation; hence adopting an algorithm which improves the rule execution process was necessary. The existing approach of depth-first search was computationally expensive; therefore we had to adopt the Q-learning algorithm to learn the paths of the execution process to optimise its performance. Prior to implementing the Q-learning algorithm, we evaluated JAES with real world legal case hearings, to determine its effectiveness to reach legal conclusions. For a successful implementation of the Q-learning algorithm, we had to modify the existing modular decomposition structure to accommodate the changes in the design. Furthermore, after the implementation, we evaluated the effectiveness and the performance of JAES and compared the results using scenario-based legal cases. The results were impressive, as we were able to optimise the JAES rule execution process by 33%.

1 Introduction
Artificial Intelligence (AI) aims at understanding and representing domain intelligence by constructing computer systems that can exhibit intelligent behaviour. One of its branches concerned with the reasoning process and the representation of knowledge for inferencing by a computer system is an expert system [1]. A Judicial Advisory Expert System (JAES) has been successfully designed and implemented to resolve ownership conflicts covering the Sale of Goods Act (SGA) sections 16 to 20. This has led to the addition of further sections such as sections 12 to 15 of the SGA to JAES, consequently adding more rules to the knowledge base. One potential technical problem identified from such a module addition is the size of the knowledge base. If the system consists of several hundreds of rules, it takes a substantial amount of time to produce a desired conclusion.

Furthermore, as the system contains large quantity of information in its working memory [2], it slows down the performance of JAES unless a very good searching and indexing technique is implemented. Fig1 clearly shows a need to optimise the performance of JAES in terms of its rule execution process. This is because as the number of rules increases, the performance degrades. It is realised that the optimisation needed in this context is at the design level of JAES, to make the best use of available resources. The current approach using a depth-first search (DFS) technique was computationally expensive [3] and needed to be replaced.

The aim of this work is to implement an effective learning / searching algorithm to optimise the JAES rule execution process, by learning the best available path for the rule execution. We have adopted and integrated the Q-learning algorithm [4] for the optimisation process in JAES. Q-learning is a simplification of the reinforcement learning technique which has been extensively applied in the field of robotics, multi-agent systems, games, just to mention a few [5, 6].
The paper is organised as follows. The next section describes the legal foundation, design and development technique implemented in JAES. Section three covers the preliminary evaluation (pre-implementation of Q-learning) stage where real world legal cases are resolved using JAES. Section four describes the approach to the implementation of the Q-learning algorithm followed by an evaluation section, which details the post-implementation Q-learning process and the results obtained from this. Conclusions are drawn based on the evaluation results in section six of this paper.

2 Foundation

JAES has been designed to resolve disputes between buyers and sellers when the goods being delivered or sold are either damaged or lost. The main dispute which arises is the ownership of the goods and therefore the ownership of the risk. The domain under consideration is SGA sections 16 to 20 which covers the transfer of property and the risk between parties [7]. Property can be defined as the “general property in goods” and it is usually used by the lawyers to signify the title or the ownership of goods. In everyday life, the same is applied to the sale of goods [8], yet the SGA deals with the transfer of property as between the buyer and the seller and contrasts this with the transfer of risk.

According to English law, the buyer can become the owner of the goods before they are actually delivered to him/her and, conversely, the seller can retain the property even if the goods are delivered to the buyer [9]. Due to the legal and commercial importance of property and ownership, SGA contains detailed rules governing various aspects of its transfer including the time when the property is transferred from the seller to the buyer. It is this aspect which is covered within sections 16 to 20 of the SGA and the transfer of property and risk from the seller to the buyer apply whether the goods are specific or unascertained [10]. JAES takes into account all these rules of SGA (sections 16 – 20) and represents them using rule-based forward chaining technique [11]. It has been designed using Unified Modelling Language (UML) [12] and developed to execute on a Microsoft Windows.NET platform.

In order to manage the complexity and provide an ability to adapt the system to any changes rapidly, a modular decomposition structure is employed as shown in the fig 2. It facilitates the system development using previously developed system modules. This increases the flexibility, functionality, efficiency and the usability of JAES. It decomposes the design architecture into three levels i.e. System_Package, Sub_Package and Modules. Every Sub_Package has its corresponding module implementation. The knowledge base is a problem-specific module which contains all the information to control the inference engine [13]. The construction of JAES is done using standard .NET classes to execute knowledge base rules. The representation of the rules is kept as simple but as explanatory as possible in order to provide the correct guidance to the user.

3 Preliminary evaluation

JAES was initially evaluated to resolve scenario-based legal cases dispute resolutions. This was done to ensure that the system was effective enough to be further evaluated and implemented on the real-world legal ownership dispute case environment, within the SGA sections of 16 to 20. From the successful evaluation results obtained, it was further evaluated on real-world legal cases. The set up was carried out on a single CPU based laptop installed with MS Windows XP operating system and JAES. The approach adopted for testing JAES was to present several different real world legal cases with known solutions to ensure that the actual conclusion matched the expected conclusion, in terms of the SGA sections as well as the risk ownership. These cases were real court hearings of the year 2005 / 6, where the conflict resolution emerged between parties under sections 16 to 20 of the SGA. A total of 10 such real court hearings were evaluated using JAES and it proved that the effectiveness of JAES was unquestionable. Table 1 clearly shows that for each real world legal case under evaluation, JAES provided the...
corresponding expected outcome. One such real world legal case is shown in the fig 3. For anonymity reasons, original names of people and places are replaced and other details are obfuscated. Since the goods were specific (S17(3)) and there was no expressed intention as to when title was passed (S17(5)), plus the risk prima facie passed with the title (S20(1)), the risk was going to be borne by Bill as shown.

Fig 3: Real world case hearing : Test Case 4

Real-world legal cases hearings : Test Cases 1 – 10, Year: 2005 / 06

X – Sections Applied, B – Buyer, S - Seller

<table>
<thead>
<tr>
<th>Ref:</th>
<th>Sections:</th>
<th>Risk Owner:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X B B</td>
</tr>
<tr>
<td>2</td>
<td>X X</td>
<td>X B B</td>
</tr>
<tr>
<td>3</td>
<td>X X X</td>
<td>X S S</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X S S</td>
</tr>
<tr>
<td>5</td>
<td>X X X</td>
<td>B B</td>
</tr>
<tr>
<td>6</td>
<td>X X</td>
<td>X X S S</td>
</tr>
<tr>
<td>7</td>
<td>X X X</td>
<td>X S S</td>
</tr>
<tr>
<td>8</td>
<td>X X X</td>
<td>X B B</td>
</tr>
<tr>
<td>9</td>
<td>X X</td>
<td>S S</td>
</tr>
<tr>
<td>10</td>
<td>X X X</td>
<td>X X S S</td>
</tr>
</tbody>
</table>

Table 1: Real-world test cases hearing summary

Both the evaluations i.e. scenario-based and real world legal cases proved the effectiveness of JAES to reach a legal conclusion. Addition of further rules to the existing JAES design questioned the performance metric, with which JAES responded to resolve a conflict and conclude the results, as shown in the fig 1. There was a need to employ an algorithm in the design process which could aid the optimisation of the rule execution process. Ultimate aim of our work was to implement a simple algorithm that can learn the best sequence of actions to take while searching through the knowledge base, in order to reach the decision as quickly and as accurately as possible. This led to the adoption of the Q-learning algorithm described in the next section of this paper.

4 Q-learning algorithm implementation

Prior to the implementation of the Q-learning algorithm, it becomes necessary to understand the existing searching scheme integrated into the JAES design. The inference engine searched through the knowledge base using a depth-first search approach [14] as shown in fig 4, where the actions were executed whenever the corresponding conditions of a rule were true. The system assigned values to the attributes, evaluated the conditions and checked to see if all of the conditions in a rule were satisfied. Such a method is computationally expensive in applications like JAES which have a deep hierarchy. This is because the number of paths in JAES exponentially grew with the number of nodes, when the method traversed all the paths from the current node to the leaf node [3].

Fig 4: Depth – First Search in JAES

The Q-learning algorithm is a simplification of the reinforcement technique [4] which defines a Q-function for each state-action pair \((s,a)\) [15]. For every state there are a number of possible actions that could be taken by the agent. Each action within each state has a value according to how much or little rewards the agent gets for completing that action. These values (i.e. Q-values) are stored in a Q-table. The approach to the implementation of Q-learning algorithm is done via creating Software agents in JAES, which learns by interacting with its environment. The desired sequence of actions are optimally chosen to maximise the rewards accumulated over time by the agent [3]. The best possible actions are chosen based on a number of episodes (i.e. exploration from one state to the other to reach the goal) that the agent has to undergo in order to learn the environment for executing the best possible path. In other words, the agent undergoes learning sessions by exploring its environment and getting its corresponding reward until it reaches its goal state (i.e. desired conclusion).
The immediate reward the agent gets to execute an action $a$ from state $s$, plus the value of following an optimal policy thereafter is the Q-value. The higher the Q-value, the greater the chance of that action being chosen. The simple algorithm employed in JAES is as follows [16]:

1. Initialise all the $Q(s,a)$ values to zero.
2. Observe the current state $s$.
3. Do While not reach the goal state:
   - Select one action $a$ from all the possible actions and execute it.
   - Receive the immediate reward $r$.
   - Observe the new state $s'$.
   - Get maximum Q-value of the state $s'$ based on all the possible actions.
   - Update the $Q(s,a)$ as follows:
     $$Q(s,a) = r(s,a) + \gamma \cdot \text{MAX}[Q(s',a')]$$
     (1)
4. Set the next state $s'$ as the current state $s$.

Based on the above algorithm, once the Q function is converged (equation 1), the optimal JAES policy is to select the actions in each state with the highest reward. This provides a mechanism of searching through the best possible execution path to reach a legal decision. We attempt to show this by the following example: Assuming that the system has to make a legal decision which deals with only 4 rules in JAES. The ultimate aim is to identify the best optimal path the system has to take to make the decision. Initially, the Q-table will have all the Q-values as zero (as the agent has no experience of its environment) as shown in the table 2. After the convergence, the Q-values will be populated as shown in table 3. The corresponding state flow diagram is shown in fig 5. The process by which the JAES Software agent reaches its goal (i.e. decision) in an optimum way is by tracing the sequence of states. By doing this it can easily compute by finding the action that result in the maximum Q for a state.

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R1</td>
</tr>
<tr>
<td>R1</td>
<td>0</td>
</tr>
<tr>
<td>R2</td>
<td>0</td>
</tr>
<tr>
<td>R3</td>
<td>0</td>
</tr>
<tr>
<td>R4</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Initial Q-table values

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R1</td>
</tr>
<tr>
<td>R1</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>52</td>
</tr>
<tr>
<td>R3</td>
<td>80</td>
</tr>
<tr>
<td>R4</td>
<td>62</td>
</tr>
</tbody>
</table>

Table 3: Final Q-table values after convergence

**Fig 5: State flow diagram**

Suppose the initial state $s$ is R2 (Rule 2) and the goal for the legal decision is R4 (Rule 4). The agent can use the Q-table shown in the table 3 as follows:

- From R2, the maximum Q-value produces an action to go to R3.
- From R3, the maximum Q-value produces an action to go to R1.
- From R1, the maximum Q-value produces an action to go to R4 (goal state).

Thus the sequence of the rule execution is:

**R2 – R3 – R1 – R4**

This is just an example which deals with 4 rules but the design and execution becomes very complex when dealing with hundreds of rules. JAES sections 16 to 20 were represented using approximately 800 rules. Since the actions were needed in every state, from our research, we identified that the agent occupied more memory space for searching through JAES. In addition, incorporation of this Q-learning algorithm to the existing design (i.e. the modular decomposition structure) seemed to degrade the performance. The approach adopted was to re-design the modular decomposition structure and create a “Master Search Module” as shown in Fig 6. In this design, every Software agent is created for its corresponding module implementation. Each agent carries out the Q-learning approach in its
environment (i.e. within its search module) and arbitrates the final results to the master search module [17]. The master search module makes a final decision by selecting the most suitable action based on the Q-values received from each agent search module. The value of one agent search module is independent of the value of the other agent search module. The highest Q-value is selected as the execution path for making a legal decision. This approach clearly optimised the performance metric with which JAES resolved conflicts as discussed in the next section.

5 Evaluation

Implementation of the Q-learning algorithm was initially evaluated using five scenario-based legal cases, to determine the number of episodes required to converge the Q-function when making a decision. The experimental setup was similar to the one used in the evaluation of real-world legal cases. Results of this evaluation are shown in the table 4. The maximum limit for convergence was assumed to be 100 episodes after which, we deduced the algorithm to have converged, based on the difference between the learned value and the optimal value being less than 0.01. As it shows from the table 4, the results were astonishing since the numbers of episodes required were fewer than assumed. Furthermore, the results produced an interesting outcome which enabled us to identify the number of rules executed by the algorithm to converge; we call this relation as the ER (Episode-Rule) relation. Throughout the evaluation process, the ER value was approximately 8. This indicated that, on average, the algorithm converged for every 8 rules. To further explain this ER value, suppose a legal case only required 8 rules to reach a conclusion, then only 1 episode is enough to resolve the conflict. This result is very significant in the sense that we are now more or less able to forecast the number of episodes required to converge a rule set.

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>Episodes (E)</th>
<th>Executed Rules (R)</th>
<th>ER = R/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>279</td>
<td>7.97</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>258</td>
<td>8.06</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>194</td>
<td>8.08</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>225</td>
<td>8.04</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>239</td>
<td>7.97</td>
</tr>
</tbody>
</table>

Table 4: Q-learning implementation evaluation results

These results encouraged our research work, as we further evaluated the performance of the implemented algorithm to determine the time taken to resolve a legal dispute. The approach adopted was to compare the performance results from the earlier approach (using DFS technique) to the newer approach (using post-convergence Q-learning algorithm). The main aim of this comparison was to observe the optimisation effects of implementing Q-learning algorithm to JAES. JAES optimisation is noticeable from fig 7, where the time taken to execute rules by the new approach for reaching a conclusion is less compared to the earlier approach.

<table>
<thead>
<tr>
<th>JAES Optimisation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-learning</td>
<td>DFS</td>
</tr>
</tbody>
</table>

Fig7: JAES Performance Optimisation – Average results
From the fig 7, the Q-learning implementation took 4 minutes to process 400 rules in a test case whereas the DFS approach took approximately 6 minutes. This clearly indicates the performance optimisation of 33% (on average) in resolving a legal dispute. The rationale of this outcome is clear: Q-learning seeks optimal values as it converges from any initial condition. In addition, one of the benefits presented by such an implementation is that the learned states can be reused to solve a new problem efficiently and reliably.

6 Conclusion

The work highlighted in this paper is the optimisation of JAES by implementation of the Q-learning algorithm. In this process, we replaced the existing approach of the DFS and noticed the performance improvements of 33%. During the implementation stage, we realised that the existing modular decomposition structure was degrading the performance of JAES. This was due to the agent occupying more memory space whilst searching through the entire JAES module set. The approach adopted was to redesign the structure, thus optimising the rule execution process of JAES.

While evaluating the effectiveness and determining the number of episodes required for converging a rule set, we identified that on average, the ER (Episode-Rule) relation yielded a value of 8. This is very significant as we are now able to roughly forecast the number of episodes required to converge a rule set. Our ultimate aim of this work is to cover all the sections of the SGA making it a useful Software toolset, in an academic and professional environment to solve legal disputes rapidly and accurately with minimum resources. The future direction of work is further evaluation of JAES using different version of the Q-learning approach in complicated scenarios with deep hierarchies to demonstrate the robustness of our legal expert system, and improve the performance by implementing other design and implementation techniques and comparing the results obtained.

Acknowledgements

The support presented by the department of law at De Montfort University is appreciated. We would also like to thank Mr. Charles Stevens and Dr. Brian Bramer for their participation in JAES design and evaluation process.

References