IADIS International Journal on Computer Science and Information Systems
Vol. 9, No. 2, pp. 53-66
ISSN: 1646-3692

EMOTIONAL DECISION MAKING RESPONSE OF NON-PLAYABLE CHARACTERS IN A ROLE-PLAYING GAME

Umair Azfar Khan. *Graduate School of Information Science and Electrical Engineering, Kyushu University Library, Kyushu University, Fukuoka, Japan*.

Yoshihiro Okada. *Graduate School of Information Science and Electrical Engineering, Kyushu University Library, Kyushu University, Fukuoka, Japan*.

ABSTRACT

Role-Playing Games (RPGs) provide the user with a story where the decisions taken by the user move the story forward. Apart from the user, there are many Non-Playable Characters (NPCs) in the games that provide coherence to the overall story. These characters however have limited choices and functionality and always behave the same way in a game. The actions performed by the NPCs are never unexpected and we do not receive an emotional response from the NPCs. The authors propose a system where every character has some specific objectives and to achieve those objectives the character formulates a plan and acts on it according to its defined ethical orientation. Once the plan has been created, there is a possibility that the character might behave emotionally to that plan based on its orientation. As a result, the user gets a dynamic system where every character has the possibility of influencing a change which is governed by the mental makeup of the characters.

KEYWORDS

Character creation, genetic algorithm, planning graph, decision making, agents.

1. INTRODUCTION

The area of Role-Playing Games (RPGs) has improved quite drastically over the past few years. This focus has been mainly towards graphical improvement and having scripted sequences to move the story forward. There have been many popular Role-Playing Games (RPGs) like Mass Effect series, The Witcher series, The Fallout series and the Elder Scrolls series which provide an illusion of a living world where the user chooses have an impact on the entire game world. These choices have consequences and these are sometimes reflected by

the behavior of the Non-Playable Characters (NPCs) during the course of the game. The NPCs however lack the capability of making these choices on their own and they do not exhibit an emotional response to a decision as well due to the lack of autonomous behavior of the NPCs. This trend is however slowly shifting towards creating characters that show cognitive behavior, that is, they have reaction while interacting with other characters, and show consistency over time towards their goals. We are seeing an increased use of techniques such as fuzzy logic and neural networks to enhance the decision functions of the characters (Ayesh et al. 2007). These features help in making realistic individual behaviors that help the NPCs blend into a crowd in a natural way.

Finite-State Machine (FSM) has been the most common methodology for programming Artificial Intelligence (AI) in games. This powerful organizational tool allows the implementation of intelligence systems with ease and flexibility by helping the programmer to break a problem into sub-problems (Schwab 2009). The FSM is composed of all the states that a character can be in and is initialized by declaring all the transitions for every state. If some transition has occurred during the time the game was run, then the correct state is acquired as the current state of the FSM. FSM adheres to the changes in its surroundings to initiate transition to the next state. It is a reactive mechanism and does not give the AI a suitable plan of action or initiative that governs the states that it must achieve in order to complete its goals. Orkin (2006) realized this limitation and introduced planning techniques based on STRIPS (Fikes and Nilsson 1971) and goal-oriented action planning (Orkin 2003) in the game, F.E.A.R., which lead to a more natural behavior. In real life whenever a plan is created, we always have an alternative plan in case the original one fails. Therefore failure does not result in the giving up of the plan but it means that a new approach should be taken to achieve the objectives. Using a planning graph allows the characters in the game with the same possibilities; therefore alternative approaches by the characters are used to arrive at the objectives.

Our research includes user interaction, evolutionary algorithms and autonomous agents for creating animated stories in a game world. The user makes choices while defining the characters and the objectives that they need to accomplish. The characters then communicate with the item they are supposed to interact with and are given a plan to follow. Pollack and Horty (1999) argue that in a dynamic and real-time planning the autonomous agents must be able to manage the plans that they generate. Due to the dynamic nature of the game, the plan is subject to change with the passage of time as the states of the items can change due to interaction with other characters.

Dignum et al., (2009) argues that coupling agents to games requires a design methodology where agents should be included from an early stage in the design process to profit from specific agent characteristics such as communication, cooperation, reasoning, proactive behavior, and adaptability. Our system however limits this interaction among the characters. The behavior here is better explained as competitive rather than cooperative as the characters might compete to perform actions on the same item. Since every character follows a planning graph, it has to follow a series of actions when done in a proper order lead it towards its goals. If at any point performing an action becomes impossible, a backup plan is followed to lead the character towards its goal. We also employ an orientation system which decides the alignment of the character towards good and bad. This alignment helps in deciding what choice a character will make when confronted with two decisions of opposite alignments that might lead it to its goal. This decision making process helps the AI to maintain its character throughout the story while the planning graph allows it different avenues by which a goal can

be achieved. It is quite possible that if the only possible action that can be done is of opposite orientation, the character might choose not to do the action rather than try to achieve its objectives. This decision is based on the mental resilience of the character as it is taken based on how strong the character feels about doing an action with an opposite alignment.

2. SYSTEM SETUP

The history of this project is set in the area of story generation. The departure was made from creation of a story narrative towards the creation of an animated story. This however did not change the important components of story that need to be put in place in order to make it meaningful. These components were the characters and the items around which the entire story is formulated due to the performance of actions. As a result, special emphasis was given towards the creation of both characters (which can be referred to as agents here) and items. As according to Dignum et al., (2009) the design methodology requires that the characters and the items be added to the game in an early stage; therefore an administrator decides how many characters need to be in the game and what items should be added to it. The state of the characters and the items holds a lot of importance as the characters take these into account while making a decision to perform an action.

The system first defines the Items that are going to be in the system, their initial states and the actions that can be performed on those items. Then the characters that are going to take part in the story are created, their physical and mental attributes are defined and they are given goal states with respect to every item in the system. As a result, not only the character that is controlled by the user has some objectives to accomplish, but the Non-Playable Characters (NPCs) also have several objectives to achieve. This is best explained in the following sections where we define the Item and Character generation.

2.1 Item Creation

Every item in an RPG is required for creating new quests and driving the story forward. In our system, every item is composed of a list of initial states and a list of actions that can be performed on it. Not every action can be done on an item in real world, therefore the item contains a list of all the doable actions. An item can be in one of many states which dictate what action can be done on the item at any given time. Based on the initial states of the item, actions can be performed on it, thus resulting in new states. If the user wants to create a complicated story, all that needs to be done is to add actions to the item thus increasing the functionality of that item.

2.1.1 Action

An action can only be performed if the item is in a certain state. A good example of this is that of an electrical switch which can be turned on if it is off, or it can be turned off it is on. At the same time, we can choose to do nothing at all and let the state of the switch persist. In order to do any action, there needs to be a list of states called *preconditions* that need to be met. Preconditions are states that an item must be in to allow an action to be performed. When any action is performed, it produces a list of *effects* that can change the state of the item. When

actions are performed in a series, then the effects of one action might become the preconditions for another, thus creating a plan that might result in reaching the goal states.

Any character that interacts with the item has some initial states with respect to that item. The initial states are compared with the preconditions of the actions that can be performed on the item. If the preconditions of the item match with the initial states of the character, then the character can perform that action on the item thus producing a list of effects. The production of effects means that the states of the item have changed and the new actions can only be performed if their preconditions match with the effects of the previous actions.

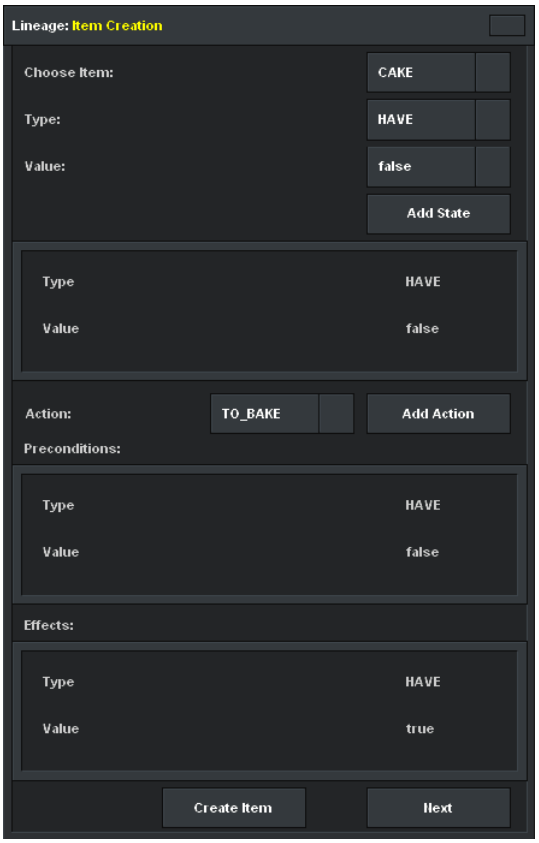

Figure 1. Item creation window

In the Item Creation Window as shown in Figure 1, the user first selects the item that needs to be added to the system. After that the user selects the type and value of the initial states that the item will be in at the start of the game. Afterwards the user adds the actions that can be performed on the item. As the action is selected, its preconditions and effects are displayed in the window so that the user knows what outcome can be expected by doing an action. Finally, when the user is satisfied, the item can be added to the system.

2.2 Character Creation

In our previous paper (Khan and Okada 2013) we have discussed our character creation process where we use Genetic Algorithms to create a character with both physical and mental characteristics. For the creation of mental attributes a Genetic Algorithm coupled with Knapsack problem is used to produce characters according to the desired orientation. This is achieved by giving different numerical values to actions that a character possesses and their combined sum beyond a chosen threshold defines the character's orientation towards good or evil. The numbers of actions that can be performed by a character are limited and the algorithm tries to fit the right actions to achieve the desired numerical value. Since some actions will be left out, the character will not be able to perform all the available actions. The threshold value can be changed depending on the flexibility the user wants to provide the algorithm. The aim of the algorithm is to arrive at a solution that matches the required sum as closely as possible. Since the requirement is to achieve a sum closer to and not exactly the desired value, it keeps the characters random.

As shown in Figure 2, in order to have an ideal good character, the sum should be equal to 45. But the probable solutions do not produce that exact value but are still acceptable. Similarly, in order to have an ideal bad character, the sum needs to be 145 but it is not achieved by any of the probable solutions. This is due to the fact that after many crossovers and mutations the good gene stayed within the chromosome and was not removed through subsequent generations. Therefore we can never be sure of what kind of character will be acquired at the end of subsequent iterations. In the search for a good character, we might get a selection where a few actions are bad, while initiating a search for a bad character might produce a selection where some actions are good. Currently, these actions have a certain numerical value but it does not reflect the cost for performing them. The cost for performing actions is dependent on the orientation of the character which will be explained in further detail when we explain the Decision Making process. For example, the action for fighting might be easy to do for a character with bad orientation, so it will have a lower cost for that character, but it will be very difficult to do for a good character and will have a higher cost for that character. This difference becomes very important if the same plan is applied to two characters with different orientations as their response will be very different.

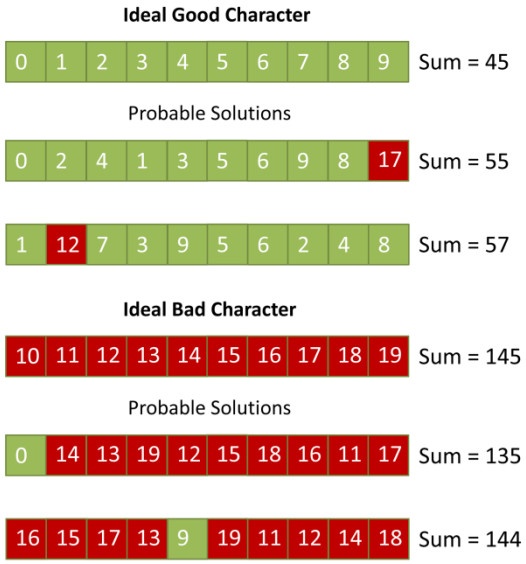

Figure 2. Probable solutions through genetic algorithm

In Figure 3 the administrator first selects the orientation of the character and then selects each item that was previously added to the system. The administrator then selects the goals that the character has for every item and finally adds the character into the system. If the Main Character checkbox has been selected, then the character will be controlled by the user and a plan will not be generated for the character to achieve its objectives. As all the required characters and their objectives get defined, the administrator moves forward to the next window to finalize the selections.

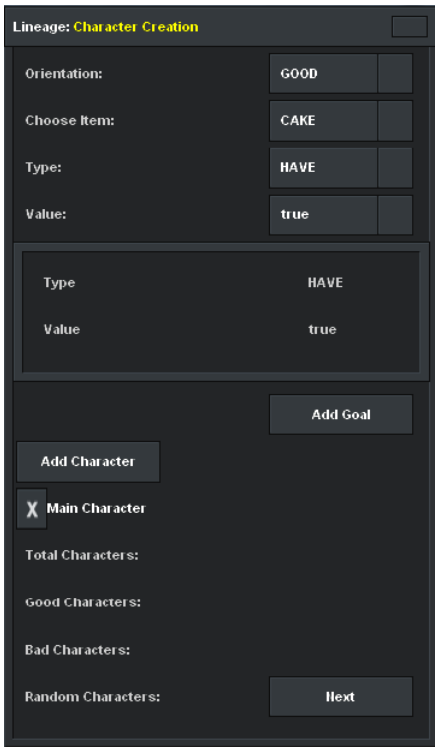

Figure 3. Character creation window

The final part for setting up the system is that for finalizing all the characters. We can see in Figure 4 that there are 4 separate windows that perform very distinct operations. The Abilities window produces abilities as according to the orientation of the character, selected by the administrator. The administrator can change the attributes by selecting "Good", "Bad" or "Random" to change the character's orientation if needed. The Physical Attributes window allows the user to change the physical appearance of the character. The character is produced by the system by taking into account all the past selections made by the administrator. Once the selection has been made, the administrator can add the character as a parent for the next series of creations or can even use one of the parents as the selection for the current character. If the administrator wants to use the previous selections to get a new looking character, then the Population window can be used to generate the next character. The administrator can select the number of generations that the genetic algorithm should use to create the next character. The GA uses sigma selection to arrive at the most favorable character according to the previous selections made by the administrator. Once the administrator is happy with the character, the Finalize Character button can be pressed to finalize the selection. When all the characters have been finalized, the "Simulate" button can be pressed to start the game. Now the characters will communicate with the items and get planning graphs in return so that they can select which actions they should do to achieve their objectives.

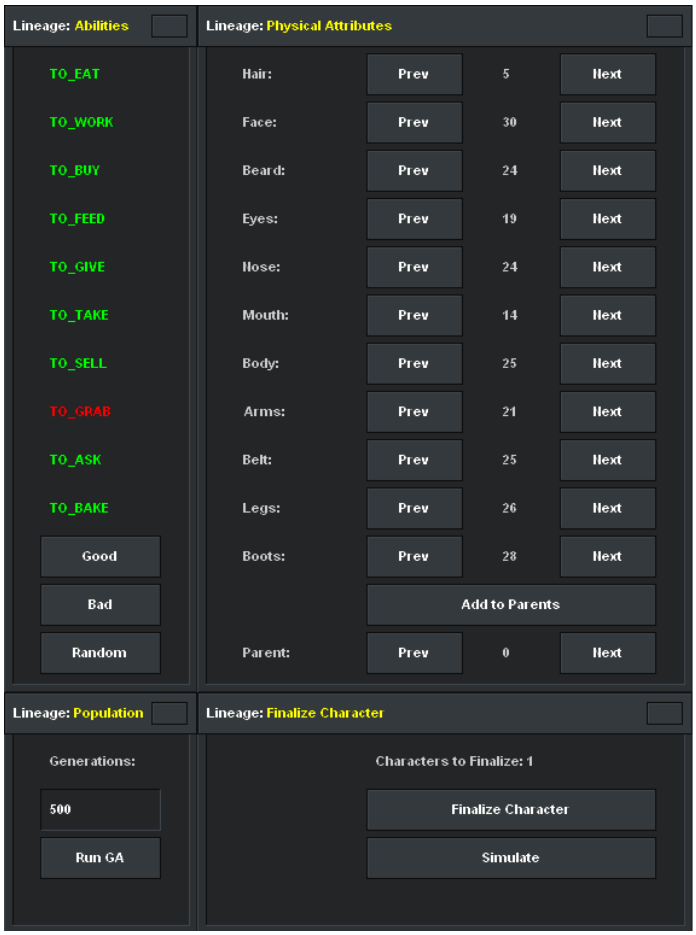

Figure 4. Character finalizing window

2.3 Planning Graph Creation

A Planning Graph encodes the planning problem in such a way that many useful constraints contained in the problem become available thus reducing the amount of search needed to find the correct series of actions. Furthermore, Planning Graphs can be constructed quickly: they have polynomial size and can be created in polynomial time (Blum and Furst 1997). For our system, we are using a technique called “Graphplan” and create a planning graph based on the defined rules for it. Since the first introduction of Graphplan with the STRIPS language, more expressive languages like (ADL) Action Description Language (Koehler et al. 1997) and (UCPOP) Universal Conditional Partial Order Planner (Gazen & Knoblock 1997) have been used for plan representation. These languages allow the use of disjunctive preconditions, conditional effects, and universally quantified preconditions and effects in action and goal representation.

We construct our planning graph by following the guidelines as given by Hong (2000) for the construction of a Goal Graph and the literature that we found with regards to the creation of a Graphplan. Due to the nature of our research, relying on a language like STRIPS or ADL was not an option. Our system is a dynamic system where the actions are done by both the user and the NPCs in real-time, thus changing the states of the item. As the states of an item are changed, the planning graphs made by other characters need to be updated to note this change. It should be kept in mind that for every item in the gaming world if a character has some goal associated with it, the character will get a separate planning graph for that item to achieve its goal. The concept of a planning graph is to map out all the possible outcomes that can take place within a certain time period. If the goal states are reached within that time period, we consider the planning graph to be a success. From the goal states we backtrack up to the initial state noting down all the actions and the sequence in which they need to be performed. At any level within the graph, there can be two or more actions that lead to the same state. Since all these actions have been stored, the character has a choice to pick any action that suits its needs.

We will now define how a graph can be constructed by changing initial states and preconditions. We start off by creating a graph for a character which is hungry. As shown in Figure 5 (a), the initial state of the character is Hungry and the state to achieve is Not Hungry. The possible actions that can be performed are Do Nothing and Eat. The character can perform the Eat action to reach the goal state. In Figure 5 (b) we change the preconditions for the Eat action by adding another state, Has Food. The graph changes a little but the outcome remains the same.

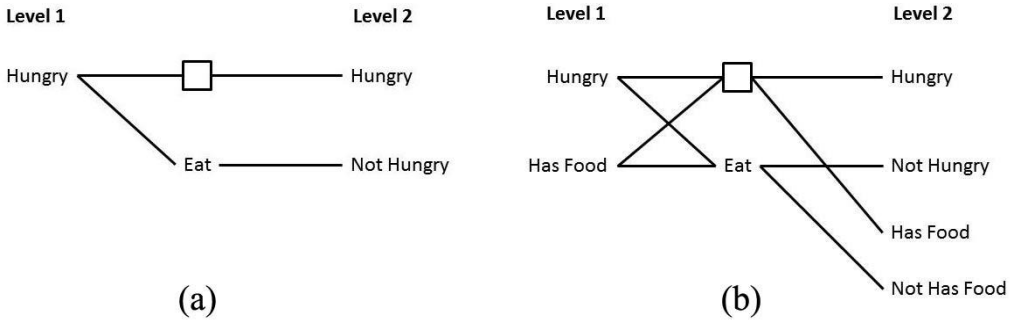

Figure 5. Change in a planning graph by increasing the preconditions

We change the initial condition from Has Food to Not Has Food. Since the character no longer has food, it cannot perform the Eat action. The character will now need to perform an action that is able to produce food and only then it can perform the Eat action. There can be many ways in which food can be acquired, but for this example we have chosen the BAKE action whose effect produces the state Has Food. The graph that is acquired is shown in Figure 6.

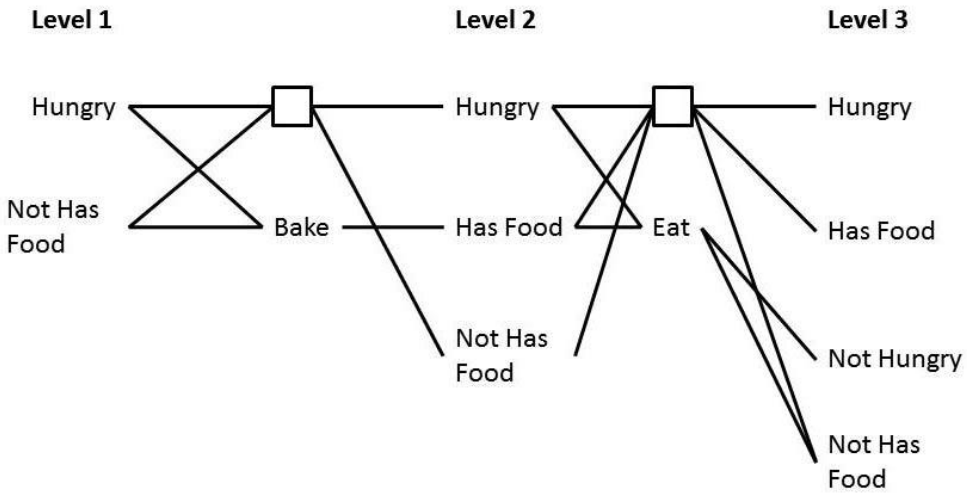

Figure 6. Planning graph that is acquired after changing initial conditions

The goal state is achieved in Level 3 and when we backtrack from that state to initial state, we find that the actions that the character will need to perform are Bake and Eat in order to reach the goal state. As a result, this is the order in which the actions will be performed by the character and it will disregard all the other actions that are present. In this example, there is only one path that leads to the goal state, but it is a possibility that there can be multiple routes that lead to the same goal.

We can apply this graph to one of the popular tales like the Little Red Riding Hood. When that story starts, the mother of Red Riding Hood initially does not have a cake that she wants to give away, so her initial states are ¬Have(Cake) and ¬Given(Cake). She wants to reach a state where she does not have a cake and she has already given a cake, so ¬Have(Cake) and Given(Cake) are the final states. Therefore, we see in Figure 7, that she first needs to bake the cake and once she has the possession of the cake, she can give it away. This is achieved in Level 3, however, we expand the graph one more time to see if the states are going to be repeated. As the states in Level 4 are the same as in Level 3, we arrive at the conclusion that the graph cannot be further traversed and back track from the final states in Level 3 to arrive at the beginning of the graph.

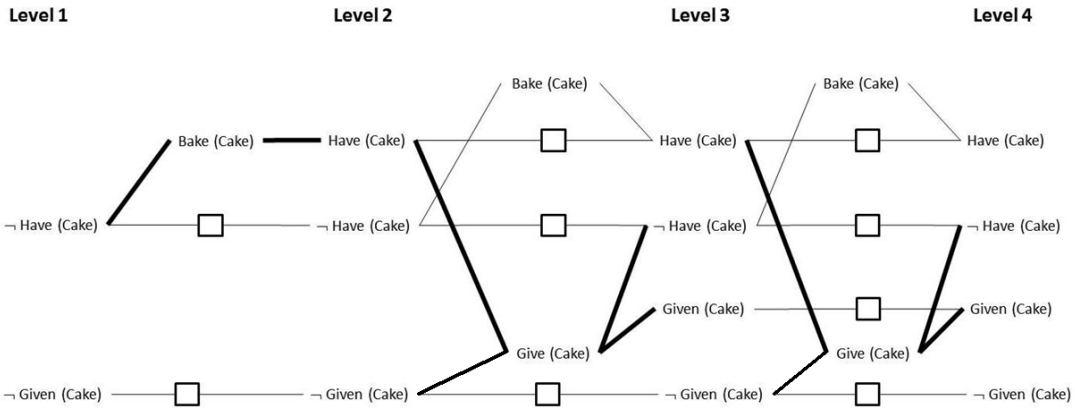

Figure 7. Planning graph stops generating levels if the states of two consecutive levels are the same

This path that is obtained is made up of actions from one level going to actions of the next level. The cost of actions provides the weights for this graph which will help in traversing it by using the A* algorithm, which is actually a path-finding algorithm used in games (Orkin 2006). A* is a directed algorithm and does not search for a path blindly (Matthews 2002) as it keeps checking the distance covered using a robust heuristic. So which route should be taken is dependent on the cost of the actions as dictated by A* but the decision is very much dependent on the orientation of the character. We therefore need a decision making process that can make a suitable choice for the character.

2.4 Determining Heuristic

Defining an admissible heuristic is complicated for this implementation of A* algorithm. One of the easiest ways by which it could be defined was to have the action with the lowest cost repeated over and over again. The problem with that approach is that actions depend on states and whenever an action is performed, it ends up changing the state of the item. As the effects of one action become the prerequisite for the other, it makes the application of the same action repeatedly impossible as shown in Figure 8.

Figure 8. Using same action repeatedly is an inadmissible Heuristic

The second technique that can be used is to have two or more actions be performed alternatively. The problem with this approach is that the two actions may not satisfy all the states that need to be achieved in order to arrive at the desired goal states. This can be solved by breaking the action selection into two parts; the first part being the selection of actions that satisfy all the initial states. An item can have many initial states and based on these initial

states, we can find the actions that can be performed and then select the least costing actions from these doable actions that satisfy all the initial states.

This only completes the first part as all the initial states get satisfied. The second part is about satisfying all the goal states. As with the initial states, an item can have many objective states. We select all the actions that can reach one of those states and then select the least costing actions that can satisfy those states. As a result, we get a list of actions that must be used to get a viable heuristic. All the states get covered and the problem that a single action cannot be consecutively repeated, gets solved. A small example is shown in Figure 9 where the two actions completely define the heuristic for the A* Algorithm to start from the initial states and end up at the final states.

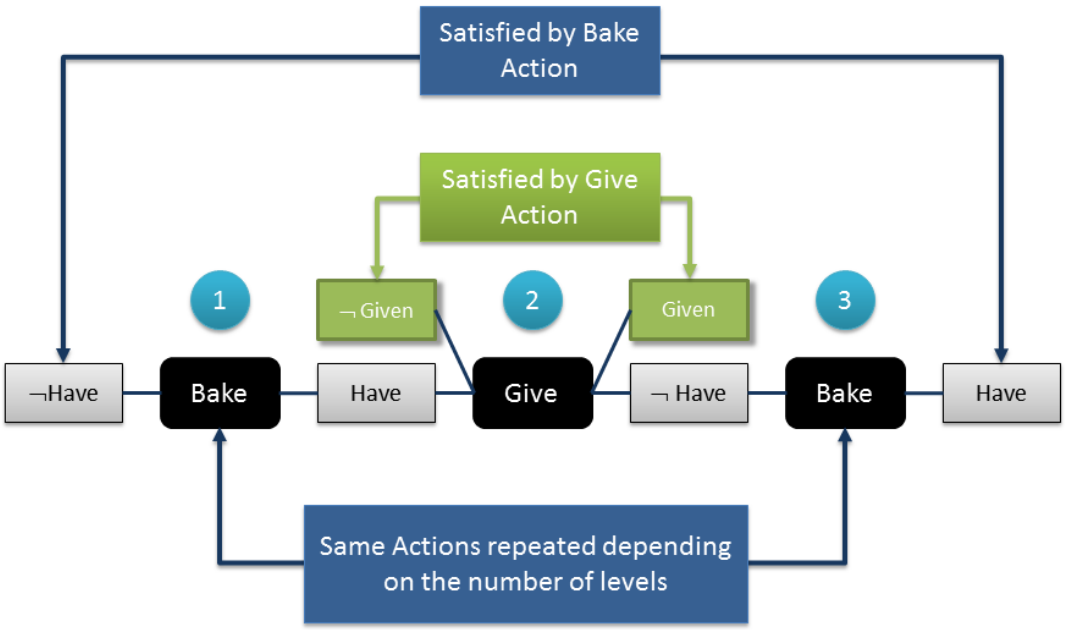

Figure 9. Finding an admissible heuristic

3. DECISION MAKING

One of the main objectives of our research was for the characters in the system to exhibit a natural response. The inclusion of the mental attributes was done for this purpose as it causes the characters to respond to different actions based on their orientation. Once the planning graph has been generated, the character first needs to see if the actions suggested by the graph are available to the character. It then needs to find all those actions that are not according to its orientation. It can be possible that two actions might have the same cost but exhibit opposite orientation. Since the A* algorithm is cost dependent, we needed to find a way by which one action should be selected over the other one. How this selection is brought about is by adding some additional cost to the action with opposite orientation, thus making the selection of that action difficult as shown in Figure 10. In this figure, the green highlighted paths show the

possible routes that a character can take to reach its objectives. Both Action 3 and 7 are bad actions hence an extra cost is added to them. So the A* algorithm will select Action 4 rather than 3 as the cost for doing Action 4 is less. After performing Action 4, the character has no choice but to pick Action 7 as that is the only action available. There is a willingness attribute for every action that determines if the character is willing to do that action or not. This willingness is dependent on the number of attributes that are according to a person's orientation. If the ratio of good attributes to that bad attributes is 6 to 4, it means that there is a 60% chance that the character will be willing to perform a good action. The amount of attributes influences the decision making which exhibits a kind of an emotional response from the character. Even though the aim of the character is to achieve its goals, but if an action is against the character's orientation, he might decide not to do the action thus rejecting a sound plan. This makes the behavior of the characters very natural as in real life the good people find it very hard to do anything bad and sometimes choose inaction rather than doing a bad action.

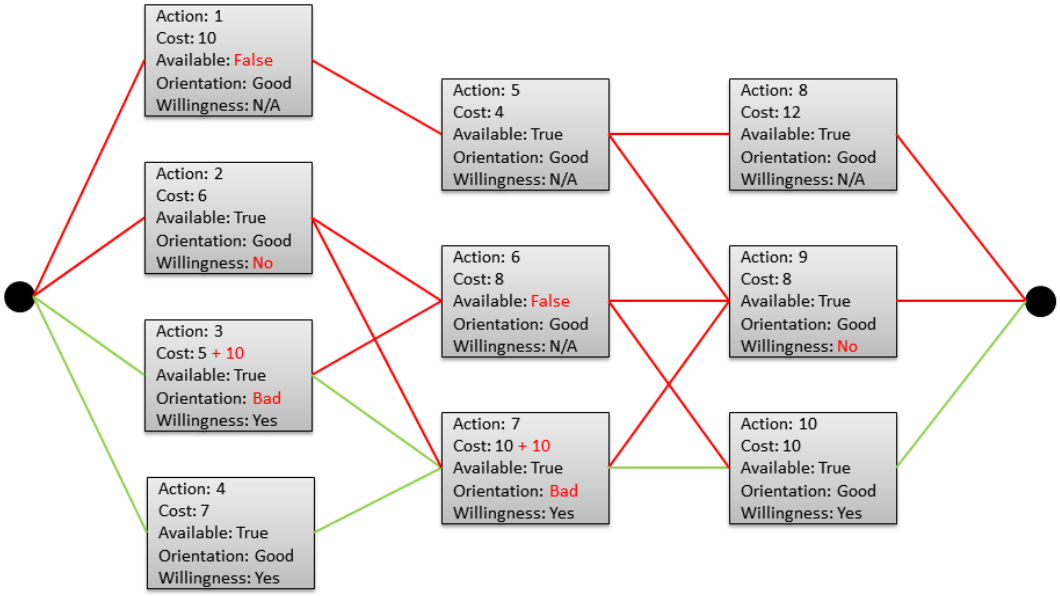

Figure 10. Soliciting an emotional response from the character

4. CONCLUSION

The aim of any Role-Playing Game is always to provide the user with an environment where the actions done by the user affect the overall game environment. That game environment is populated by many Non-Playable Characters that normally follow some scripted events and do not exhibit a proper character that feels or performs its tasks due to a certain alignment. It does not exhibit the feeling of doubt in performing an action or perform something which is out of character as real people do in real life. In our system, the orientation of a character is not absolute good or bad but stays in a rather gray area. This is made possible by defining the

orientation on the number of good or bad attributes the character contains. The more attributes are according to a certain orientation, the more the character makes its decisions aligned to that orientation.

When choosing to obtain its goals with respect to an item, the character is given a choice of decision paths that it can take in order to perform certain actions. While traversing these several decision paths, the character makes a choice about what can and cannot be done by the character. The probability of doing something that is not according to the character's orientation is improbable, but not impossible. We believe that this uncertainty regarding the character's response makes for a good research topic where the decisions are based on the characteristics of agents. If the character fails to achieve the objectives in the first try, then it can try again to see if the action that was previously rejected can be accepted this time around. This unpredictability of agent behavior ensures that if the simulation is given the same parameters and run repeatedly, it is likely to produce different outcome every time thus enhancing the experience of the users.

REFERENCES

Ayesh, A.; Stokes, J. & Edwards, R. (2007), Fuzzy Individual Model (FIM) for Realistic Crowd Simulation: Preliminary Results., in 'FUZZ-IEEE' , IEEE, , pp. 1-5 .

Blum, A.L. and Furst M.L., (1997). Fast planning through planning graph analysis, *Artificial Intelligence*, 90 (1–2) , pp. 279–298.

Dignum, F. Westra, J. W. van Doesburg A., and M. Harbers. (2009). Games and Agents: Designing Intelligent Gameplay, *International Journal of Computer Games Technology*, vol. 2009, Article ID 837095, 18 pages, 2009.

Fikes, R. E. and Nilsson, N. J., (1971) STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208.

Gazen, B., and Knoblock, C. (1997). Combining the ex-pressivity of UCPOP with the efficiency of graphplan. *In Proceedings 4th Euro. Conf. on Planning.*

Hong, J. (2000). Goal Graph construction and analysis as a paradigm for plan recognition. *In Proceedings of AAAI-2000*, pp. 774–779.

Khan. U. A. and Okada. Y., (2013). Character Generation using Interactive Genetic Algorithm, *Proceedings of GameOn 2013*. Brussels, Belgium.

Koehler, J.; Nebel, B.; Hoffmann, J.; and Dimopoulos, Y. 1997. Extending planing graphs to an adl subset. In Proceedings of 4th Euro. Conf. on Planning, 273–285.

Matthews J, (2002). Basic A* Pathfinding Made Simple, in Rabin S, *AI Game Programming Wisdom*, Hingham, Massachusetts:Charles River Media, pp 105-113

Orkin, J., (2003). Applying goal-oriented action planning to games, *AI Game Programming Wisdom 2*, Charles River Media, Brookline, Mass, USA.

Orkin, J., (2006). Three states and a plan: the AI of F.E.A.R., P*roceedings of the Game Developers Conference (GDC '06)*, San Jose, Calif, USA.

Pollack, M. E. and Horty, J. F., (1999). "There's more to life than making plans: plan management in dynamic, multiagent environments," *AI Magazine*, vol. 20, no. 4, pp. 71–83.

Schwab, B., (2009). *AI Game Engine Programming, 2e*. Charles River Media, Inc., Rockland, MA, USA.