PRODUCTION

SYSTEM OF

TOWER OF HANOI

1. Abstract
2. Introduction
3. Proposed Methodology
4. Representation of Tower of Hanoi
5. Proposed Rules for Tower of Hanoi
6. Result & Discussion
7. Conclusion
8. Reference
9. Program for Tower of Hanoi
10. Contribution of Members

* Abstract

In this project, the planning intelligence technique has been developed and applied to solve the “Tower of Hanoi” puzzle through the construction of the rule based included a set of facts and under certain conditions to describe this problem. The tower problem of Hanoi consists of different size of disks and three pegs. The proposed system is to transfer the disks from initial state to the goal state by using some rules. The based is used from planning to get the goal by applying different operations. The intelligence techniques used are contributed to reduce time and memory (state space) compared with traditional planning depended on human aid which spend more time and memory, since this technique solved a problem in the depended are planning the approach without human aid. The prolog language is used to implement the computer simulation program for the proposed system.

* Introduction

Today we are going to over how the basic usage of production systems.

“A production system is a forward-chaining reasoning system that uses rules of a certain form called production rules as its representation of general knowledge”.

In other words, production system reason in the “forward” direction, that is, from premises to conclusions. This is in contrast to backward-chaining systems (such as prolog) which start from the conclusions and try to reason backward through the premises. Planning began as an effort to design robots that could perform their tasks with some degree of flexibility and responsiveness to the outside world. Briefly, planning assumes a robot that is capable of performing certain atomic actions. It attempts to find a sequence of those actions that will accomplish some higher-level task, such as moving across an obstacle-filled room [1][2][3]. Planning is a difficult problem for a number of reasons, not the least of which is the size of the space of possible sequences of movement. Even an extremely simple robot is capable of generating a vast number of potential move sequences. Imagine, for example, a robot that can move forward, backward, right, or left, and consider how many different ways that robot can possibly move around a room. Assume also that there are obstacles in the room and that the robot must select a path that moves around them in some efficient fashion. Writing a program that can discover the best path under these circumstances, without being overwhelmed by the huge number of possibilities, requires sophisticated techniques for representing spatial knowledge and controlling search through possible environments. Most human activities require some form of planning; thus, it is not surprising that planning is a subject of an interest to the artificial intelligence (AI) community. Solving any complex task requires planning, thus, planning is very important in practice as well. Given a set of actions, the objective of planning is to construct a valid sequence of actions, or a plan, to reach a goal state starting from the current state of the system. An intuitive example of planning is the process of solving a puzzle, given the set of pieces with different geometric shapes, scattered on the floor.

* Proposed Methodology

Planning and Robotics The are two common types of planning systems “domain-independent” and “domain-conﬁgurable”. Domain-conﬁgurable planners can perform very well, but require much hand-tuning for peak performance. Domain-independent planners do not require human aid, but do not perform as well on some problems. Hybrid planners attempt to combine the strengths of these two styles while minimizing their weaknesses. The task of a planner is to find a sequence of actions that allows a problem solver, such as a control system, to accomplish some specific task. Traditional planning is very much knowledge-intensive, since the plan creation requires the organization of pieces of knowledge and partial plans into a solution procedure. Besides robotics applications, planning plays a role in expert systems in reasoning about events occurring over time. Planning has many applications in manufacturing, such as process control. It is also important in natural language understanding, where humans frequently discuss plans, goals and intentions. The actions of the traditional planning are shown below:

pickup(W): Pick up block W from its current location and hold it. It is assumed that the block is clear on the top, the gripper is empty at the time.

putdown(W): Place block W down at some location on the table and record the new location for W; W must be held by the gripper at the time.

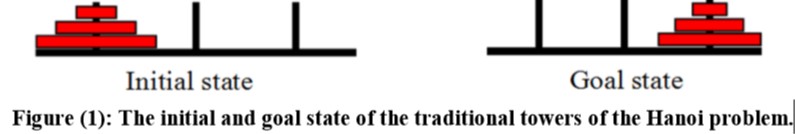
* Rule Based Planning for Solving Hanoi Problem

Stack (U, V): Place block U on the top of block V. The gripper must hold U and the top of V must be clear of other blocks.

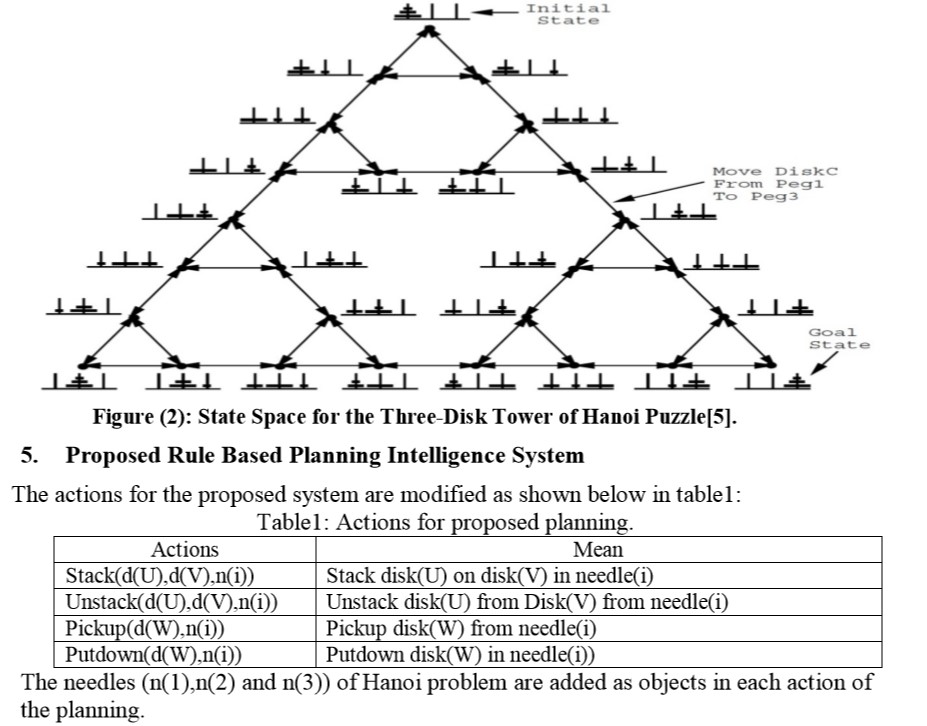
Unstack (U, V): Remove block U from the top of V. U must be clear of other blocks, V must have block U on the top of it, and the hand must be empty before this command can be executed.

* Representation of Tower of Hanoi

The Tower of Hanoi puzzle involves moving a pile of different size disks from one peg to another by using an intermediate peg. Only one disk at a time can be moved, a disk can only be moved if it is the top disk on a pile, and a larger disk can never be placed on a smaller one. Figure (1) shows the initial and goal states of a three-disk problem.



The state space for the three-disk puzzle is shown in Figure (2). Each node represents a state and is labeled with the picture of the state, and each arrow represents an operator that can be applied to reach the adjacent state. A solution to the three-disk problem given above consists of any path through the state space that starts at the initial state and terminates at the goal state. The shortest solution follows the path along the diagonal between the initial and goal states.



The proposed system that is used for planning techniques to solve Hanoi puzzle problem depended on rules base component. The rule base components embedded in the planning contains the following rules:

Rule1: if d(A) in n(1);

then pickup(d(A), n(1)) and putdown(d(A),n(2)).

Rule2: if d(A) in n(1);

then pickup(d(A),n(1)) and putdown(d(A),n(3)).

Rule3: if d(A) in n(2);

then pickup(d(A), n(2)) and putdown(d(A),n(1)).

Rule4: if d(A) in n(2);

then pickup(d(A),n(2)) and putdown(d(A),n(3)).

Rule5: if d(A) in n(3);

then pickup(d(A), n(3)) and putdown(d(A),n(1)).

Rule6: if d(A) in n(3);

then pickup(d(A),n(3)) and putdown(d(A),n(2)).

Rule7: if d(B) in n(1) and not(d(A),table, n(2));

then pickup(d(B),n(1)) and putdown(d(B),n(2)).

Rule8: if d(B) in n(1) and not(d(A),table, n(3));

then pickup(d(B),n(1)) and putdown(d(B),n(3)).

Rule9: if d(B) in n(2),and not(d(A),table, n(1));

then pickup(d(B),n(2)) and d putdown(d(B),n(1)).

Rule10: if d(B) in n(2),and not(d(A),table, n(3));

then pickup(d(B),n(2)) and putdown(d(B),n(3)).

Rule11: if d(B) in n(3) and not(d(A),table, n(1));

then pickup(d(B),n(3)) and putdown(d(B),n(1)).

Rule12: if d(B) in n(3) and not(d(A),table, n(2));

then pickup(d(B),n(3)) and putdown(d(B),n(2)).

Rule13: if d(C) in n(1) and not(d(A),n(2)) and not(d(B),n(2));

then pickup(d(C),n(1)) and putdown(d(C),n(2)).

Rule14: if d(C) in n(1) and not(d(A),n(3)) and not(d(B),n(3));

then pickup(d(C),n(1)) and putdown(d(C),n(3)).

Rule15: if d(C) in n(2) and not(d(A),n(1)) and not(d(B),n(1));

then pickup(d(C),n(2)) and putdown(d(C),n(1)).

Rule16: if d(C) in n(2) and not(d(A),n(3)) and not(d(B),n(3));

then pickup(d(C),n(2)) and putdown(d(C),n(3)).

Rule17: if d(C) in n(3) and not(d(A),n(1)) and not(d(B),n(1));

then pickup(d(C),n(3)) and putdown(d(C),n(1)).

Rule18: if d(C) in n(3) and not(d(A),n(2)) and not(d(B),n(3));

then pickup(d(C),n(3)) and putdown(d(C),n(2)).

The general rules of the stack and unstack explained below:

Rule19: if small disk on large disk in n(i);

then unstack(small disk, large disk, n(i))

Rule20: if small disk is holding by arm of planning

then stack(small disk, large disk, n(i)).

The general rules of the pickup and putdown explained below:

Rule21: if a d(W) alone in n(i)

then pickup(d(W),n(i))

Rule22: if holding(d(W)

then putdown(d(W),n(i))

Whereas:

d(A) represents small blue disk.

d(B) represents large yellow disk.

d(C) represents larger red disk

n(i) represents needle.

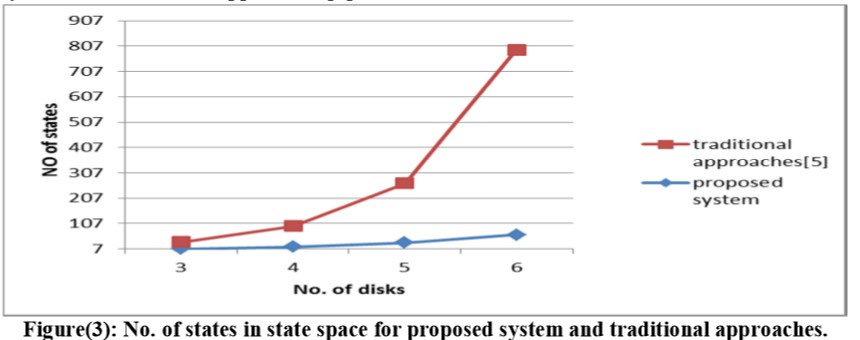
The variable W represents the symbol of disk, if the Hanoi problem deals with five disks then W can be given one of the symbols from A to E, while variable i represents the peg number. The proposed rules in planning can be solve the tower of the Hanoi problem with any number of disks.

Table 2 can explain the number of states in state space of the proposed system compared with the traditional approaches.

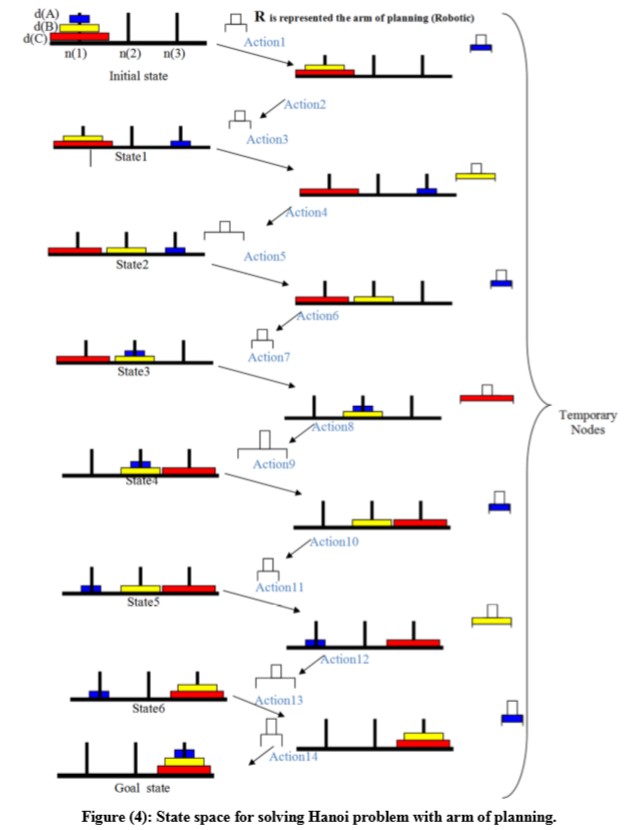
* Result and Discussion

|  |  |  |
| --- | --- | --- |
| No. of Disks | No. of states for proposed System | No. of states for traditional approaches |
| 3 | 7 | 27 |
| 4 | 15 | 81 |
| 5 | 31 | 243 |
| 6 | 63 | 729 |
| 7 | 127 | \_\_\_ |
| 8 | 255 | \_\_\_ |

The minimum number of nodes to reach a goal in the proposed system has been proved to be 2n-1 where n is represented by a number of disks, while the traditional approaches generate all states (the number of states has been computed to be 3n)[5], under consideration the traditional approaches solved Hanoi problem after generated all possible states, while the proposed system generating only necessary state to reach the goal depended on the rules. Figure (3) illustrates the number of states in the proposed system and traditional approaches.



When the above rules are applied to solve Hanoi problem, the proposed system produces the following actions and state space as shown below in Figure (4):



In Figure (4) R symbol represents the arm of planning (Robotic) and each two actions produced one state for Hanoi problem solving.

The following actions representation is implemented by the whole solution to get the goal of the Hanoi puzzle problem.

Action(1)

Unstack(d(A),d(B),n(1)) d(A),d(B),d(C),on(d(C),table,n(1)),on(d(B),d(C),n(1)),clear(d(B)), HOLDING(d(A))

Action(2)

Putdown(d(A),n(3)) d(A),d(B),d(C),on(d(C),table,n(1)),on(d(B),d(C),n(1)),on(d(A),table,n(3)), clear(d(A)),clear(d(B)),HANDEMPTY

Action(3)

Unstack(d(B),d(C),n(1)) d(A),d(B),d(C),on(d(C),table,n(1)),on(d(A),table,n(3)),clear(d(A)), clear(d(C)),HOLDING(d(B)) Action(4) Putdown(d(B),n(2)) d(A),d(B),d(C),on(d(A),table,n(3)),on(d(B),table,n(2)),on(d(C),table,n(1)), clear(d(A)),clear(d(B)), HANDEMPTY

Action(5)

Pickup(d(A),n(3)) d(A),d(B),d(C),on(d(B),table,n(2)),on(d(C),table,n(1)),clear(d(B)), HOLDING(d(A))

Action(6)

Stack(d(A),d(B),n(2)) d(A),d(B),d(C),on(d(B),table,n(2)),on(d(C),table,n(1)),on(d(A),d(B),n(2)),clear(d(A)), clear(d(C)), HANDEMPTY.

Action(7)

Pickup(d(C),n(1)) d(A),d(B),d(C),on(d(B),table,n(2)),on(d(A),d(B),n(2)),clear(d(A)), HOLDING(d(C)).

Action(8)

Putdown(d(C),n(3)) d(A),d(B),d(C),on(d(B),table,n(2)),on(d(A),d(B),n(2)),on(d(C),table,n(3)),clear(d(A)), clear(d(C)),HANDEMPTY.

Action(9)

Unstack(d(A),d(B),n(2)) d(A),d(B),d(C),on(d(B),table,n(2)),on(d(C),table,n(3)),clear(d(B)), clear(d(C)),HOLDING(d(A)).

Action(10)

Putdown(d(A),n(1)) d(A),d(B),d(C),on(d(A),table,n(1)),on(d(B),table,n(2)),on(d(C),table,n(3)),clear(d(A)), clear(d(B)), clear(d(C)),HANDEMPTY.

Action(11)

Pickup(d(B),n(2)) d(A),d(B),d(C),on(d(A),table,n(1)),on(d(C),table,n(3)),clear(d(A)), clear(d(C)),HOLDING(d(B)).

Action(12)

Stack(d(B),d(C),n(3)) d(A),d(B),d(C),on(d(A),table,n(1)),on(d(B),d(C),n(3)),clear(d(A)), clear(d(B)),HANDEMPTY.

Action(13)

Pickup(d(A),n(1))

d(A),d(B),d(C),on(d(B),d(C),n(3)),clear(d(B)),HOLDING(A).

Action(14)

Stack(d(A)),n(3)) d(A),d(B),d(C),on(d(A),d(B),n(3)),on(d(B),d(C),n(3)),HANDEMPTY.

Whereas:

Clear(d(W)) Block W has nothing on top of it.

HANDEMPTY the robot arm is empty

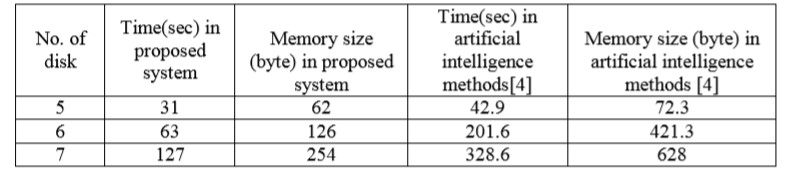
Holding(d(W)) the robot arm is holding disk d(W)

The above actions represented the steps of the proposed independent planning when started with the initial state and applied the rules from 1 to 22 for getting the goal state of the Hanoi problem in fully automation manner. Finally, the proposed system reduces the memory and time to reach the goal, when the comparison of the no. of nodes in the state spaces between the traditional approach and the proposed system is done (see Table1 and Figure3)

* Conclusion

There are several techniques to solve the problem of the tower of the Hanoi problem by using traditional approaches or depending on artificial intelligence such as (neural networks and genetic algorithms). These techniques can be classified depending on the planning (i.e. the solutions of any problem need human aid), whereas the suggested work, in this paper, has depended on the rule based on the embedded planning to solve the Hanoi problem and produce fully automatically approach. This technique can be classified into independent planning without human aid and contribution reduces time and less memory when is compared with the traditional planning. The Table 3 can explain time and memory size of the proposed system compared with the artificial intelligence methods to solve the tower problem of Hanoi puzzle.

Table3: Time and memory size that are needed for the proposed system and artificial intelligence methods to solve the tower problem of Hanoi puzzle.



* References

[1] George F Luger, “Artificial Intelligence Structures and Strategies for Complex Problem Solving”, 6th Edition, 2009.

[2] Stuart Russell and Peter Norvig, “Artificial Intelligence A Modern Approach”, 3th Edition, 2010

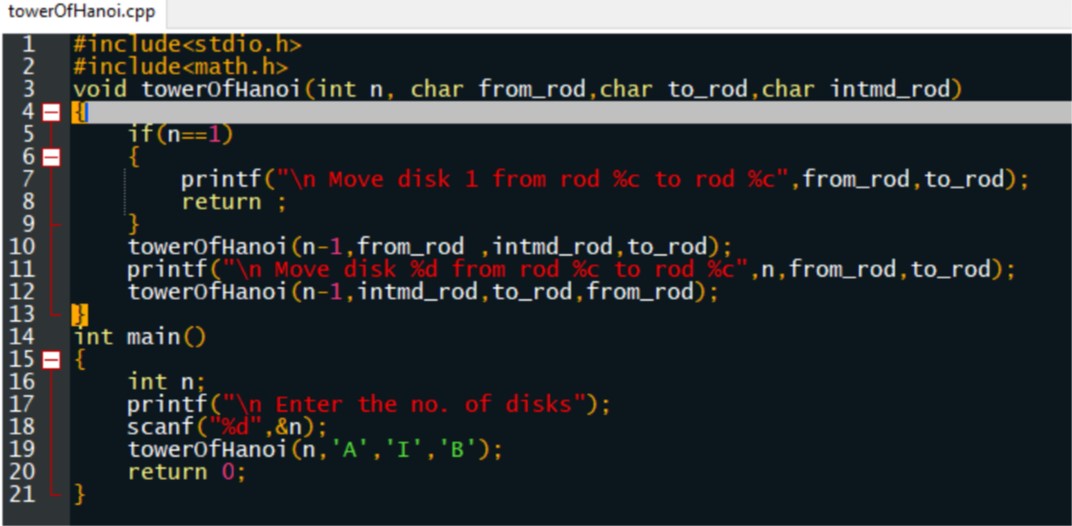
[3] Neeta Deshpande, “Artificial Intelligence”, 2th Edition,2008

[4] Han Yu, Dan C. Marinescu and Annie S. Wu ,“A Genetic Approach to Planning in Heterogeneous Computing Environments”,2002

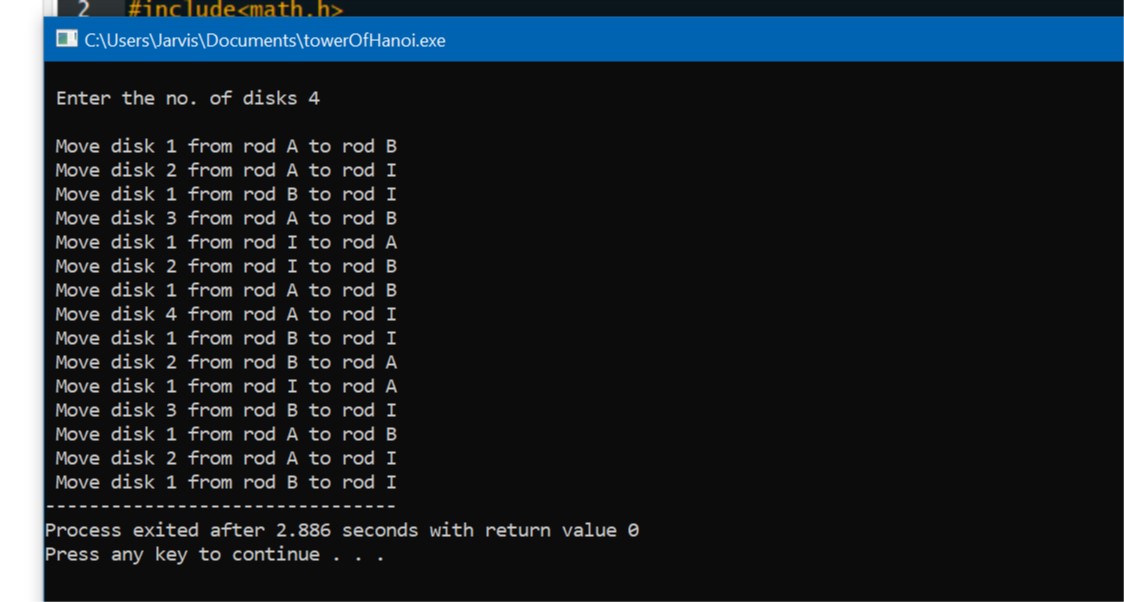
[5] Craig A. Knoblock, “Abstraction the Tower of Hanoi”,1990

* Program for Tower of Hanoi

This program is coded in C language which can solve the puzzle of Tower of Hanoi up to n number of disks.



Output:



* Contribution of Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Reg. No. | Roll No. | Section | Work |
| Asrar Ahmad | 11811292 | 55 | K18PB | b, c, e, f |
| Piyush Katyal | 11807547 | 54 | K18PB | d, i |
| Akhilesh S | 11804318 | 53 | K18PB | a, d, g |
| Sagar Singh | 11809632 | 56 | K18PB | Report making, h, g |