Group Homework 1: Classical Planning

October 6, 2014

a. Formulate a HTN planning problem for the Towers of Hanoi: HTN planning for tower of hanoi can be formulated as a recursive algorithm. At each level of this algorithm, it decomposes moving a tower of size k from source to destination into the problem of moving a tower of size $k-1$ and the moving the k^{th} disk. It can be easily visualized if we consider the tower of size $k-1$ as one big disk and the other k^{th} disk and the task is to move the k^{th} disk from source to destination. After this moving the tower of size $k-1$ can be again sub divided into the problem of moving the tower of size $k-2$ and moving the $(k-1)^{th}$ disk and so on. The recursion ends when the size of $k=1$. Algorithm 1 summarizes the algorithm.

Algorithm 1 Move-Tower($disk, source, dest, spare$)

1: **if** disk == 0 **then**
2: Move disk from *source* to *dest*
3: **else**
4: $Move-Tower(disk-1, source, spare, dest)$
5: Move disk from *source* to *dest*
6: $Move-Tower(disk-1, spare, dest, source)$
7: **end if**

b. Describe the domain knowledge you encoded and resulting planning domain: The domain knowledge that we encoded is that the towers of hanoi problem can be solved by recursively splitting the main problem into two high level actions and a primitive action. The high level action moves a tower of smaller size followed by a primitive action of moving the bottom most disk. The high level actions are further refined until it reaches a primitive action.

The primitive task is defined in Table 1 and the compound task is defined in Tables 2 and 3. Figure 1 shows the corresponding flowchart.

c. Solve your HTN problem for the cases with 3-12 discs: The code is attached. It can be executed using `python towers_of_hanoi.py <num_disks>`

move(disk, source, destination)
precond: location(disk) == source
effect: location(disk) = destination

Table 1: Primitive Move disk Task

move_tower(disk, source, destination, spare)
• task: move_stack(disk, source, destination, spare)
• precond: disk > 0
• subtasks: <move_tower(disk-1, source, spare, destination),move(disk, source, destination), move_tower(disk - 1, spare, destination , source)>

Table 2: Compound move_stack task

move_tower(disk, source, destination, spare)
• task: move_disk(disk, source, destination, spare)
• precond: disk == 0
• subtasks: move(disk, source, destination)

Table 3: Compound move_disk task

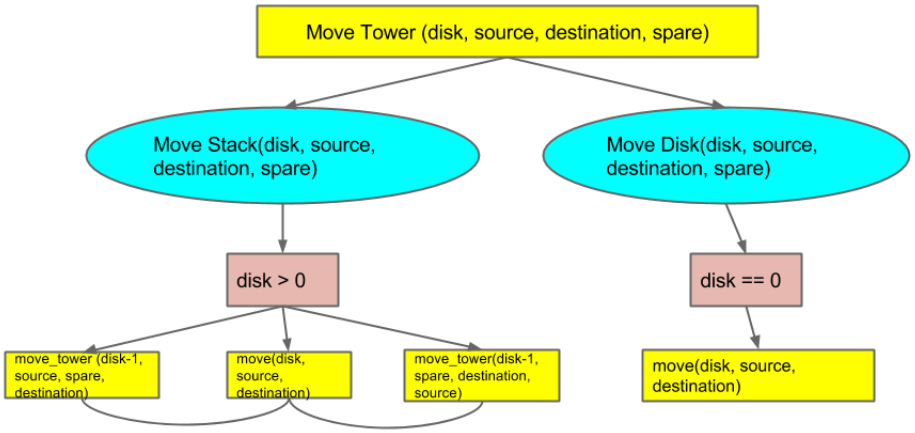

Figure 1: HTN Planning Flowchart

d. Solve the same cases with a non-HTN planner of your choice and compare the results with (c): The code and pddl files are is attached. It can be executed using `./ff -o ../hanoi-domain.pddl -f ../hanoi<num_disks>.pddl`

e. Describe your observations and discuss about them As we can see from Figure 2, 3 and 4, time taken and the number of states explored by the HTN planner is much less than FF planner. The number of steps taken in the final plan is the same for both HTN planner and FF planner.

The reduction in the number of states and the time taken to explore the states is because of the domain knowledge that we encoded in HTN planning framework. This is because during each level of HTN planning it sub divides the problem into a sequence of easier primitive/compound tasks which it can run without failure. As a result the number of states explored which didn't fail are much less for HTN planning as compared to FF planning (or any other non HTN planning). Domain knowledge guides the planner towards the right search direction.

If we reduce the amount of domain knowledge, the planner has to explore more states to find a suitable plan which results in additional time taken to find the plan. However increasing the amount of domain knowledge reduces generalizability of the planner to varied situations. So, there is a trade-off here in the amount of time taken to plan and the situations that planner can handle.

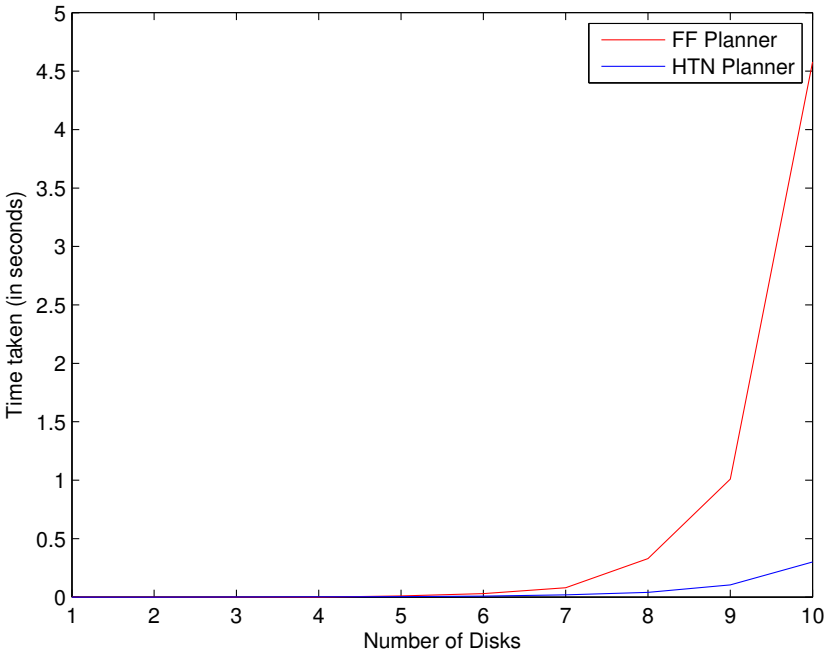

Figure 2: Time taken

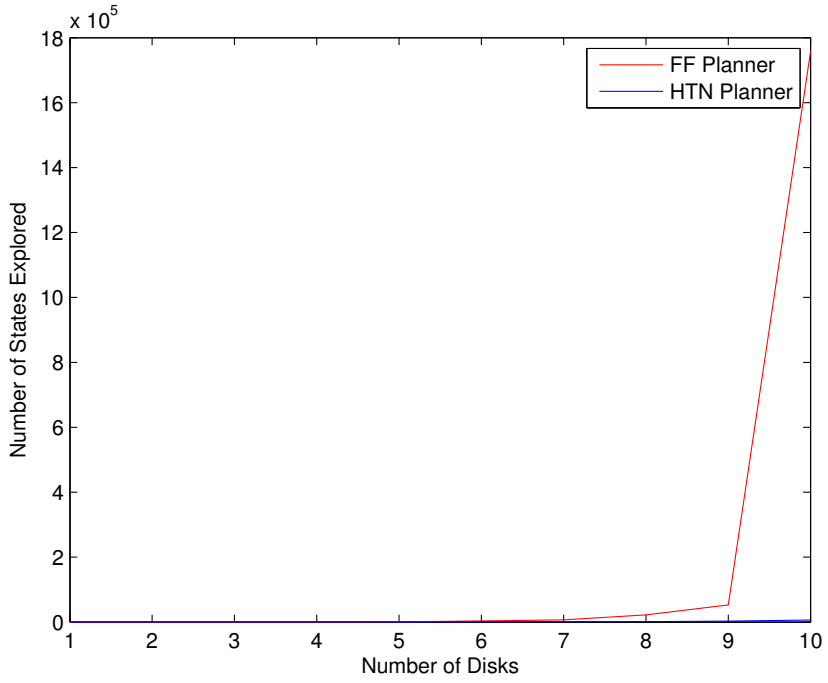

Figure 3: Number of States Explored

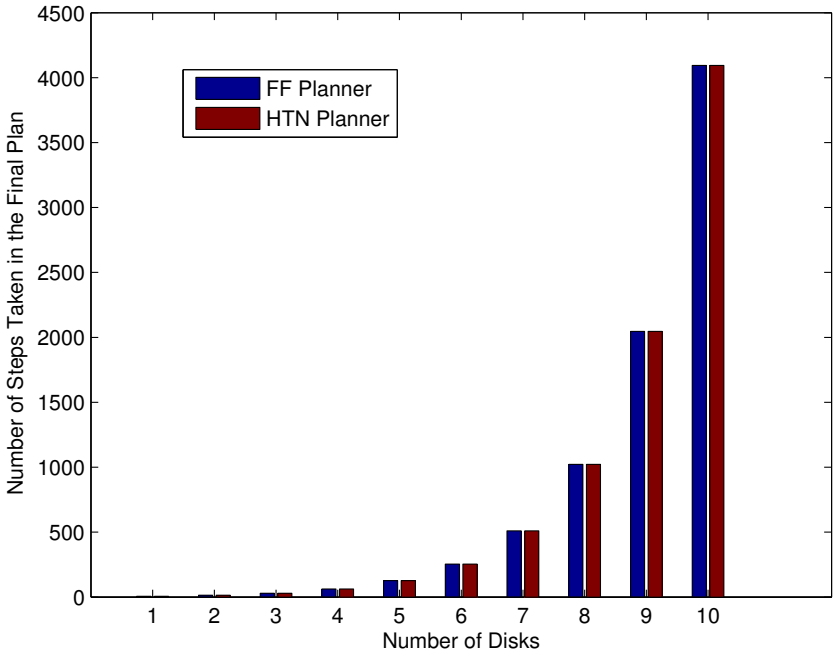

Figure 4: Number of Steps taken in the Final Plan