Report for Programming Assignment 2 (Fifteen Puzzle)

Matthew Callahan **Suphalerk Lortaraprasert**

Abstract

This will be finished after we have the results.

1 Introduction

In this paper, we examine the performance of two search algorithms, Compassion A* and RBFS, using a 4x4 puzzle as a representation. The environment is randomized to ensure optimal exploration. To enhance efficiency, we introduce a random algorithm designed to avoid repeating the same steps.

2 Environment Descriptions

2.1 Heuristic function

2.2 A* search

Algorithm 1 Programmatic Description of Simple Reflex Agent

```
1: for timestep $t$ until termination do
2:     if on dirty tile then
3:         clean tile
4:     if  not on dirty tile and facing wall then
5:         turn clockwise
6:     if not facing wall and not on dirty tile then
7:         move forward
```

2.3 RBFS search

??.

3 Experimental setup

4 Results

5 Discussion

1. Is there a clear preference ordering among the heuristics you tested considering the number of nodes searched and the total CPU time taken to solve the problems for the two algorithms?

2. Can a small sacrifice in optimality give a large reduction in the number of nodes expanded? What about CPU time?

Algorithm 2 Programatic Description of Simple Reflex Agent

```
for timestep $t$ until termination do
    if on dirty tile then
        clean tile
    random = (random action of agent) //move forward with 85 possibility and turning with 15 possibility
    if random equal move forward then
        if not facing wall and not on dirty tile then
            move forward
        else
            turn clockwise or counterclockwise //random with 50:50
    else random equal move forward
        if  not on dirty tile and facing wall then
            turn clockwise or counterclockwise //random with 50:50
```

3. Is the time spent on evaluating the heuristic a significant fraction of the total problem-solving time for any heuristic and algorithm you tested?

4. How did you come up with your heuristic evaluation function?

5. How do the two algorithms compare in the amount of search involved and the cpu-time?