2nd Lt David Crow

ENG/20M

Assignment #3 – Rush Hour

Artificial Intelligence – CSCE 523

In solving Rush Hour Problem 1 and Rush Hour Problem 2 using Graphplan, I found that both problems can be solved in 0.03 seconds. Although this is the first time we’ve solved Rush Hour Problem 2 with any method, we’ve seen Problem 1 before. According to the Homework 1 document, MATLAB also solves Problem 1 in 0.03 seconds. However, my implementation of breadth-first search solves Problem 1 in 0.016 seconds, and my implementation of A\* Graph Search solves it in 0.003 seconds. Clearly, Graphplan solves Rush Hour much, much slower than do other search algorithms.

Although I certainly prefer to solve Rush Hour in Java, I find that Graphplan allows a layman to implement a Rush Hour solver much more easily than does high-level programming. In other words, I imagine the Graphplan implementation is easier to understand than most (all?) Java implementations for those who have limited/no knowledge of either. Personally, I grew frustrated with Graphplan because the parser is nonintuitive; initially, I labeled all grid squares l00-l55, and, for whatever reason, Graphplan wouldn’t accept those names – it also wouldn’t give me any error reports. After debugging this issue, however (I now label grid squares aa-ff), Graphplan proved relatively easy to use. Without a doubt, though, I prefer standard programming simply because I have much more experience with those tools.

Hypothetically, though, if I were required to implement a Rush Hour solver from scratch, I would likely use Graphplan. Much of the ease-of-use in Homework 1 was because Dr. Peterson provided so much of the Rush Hour implementation – I would not want to implement all of that when Graphplan requires so little work to operate. On the other hand, if I already have that backend work completed, standard programming is the way to go. Encoding a Rush Hour problem into Java with a pre-existing framework is much easier than encoding the same problem in Graphplan; in Java, we don’t need an entirely new facts file for each problem.

I think the choice between Graphplan and Java primarily concerns the problem set and the need for efficiency. For large problems or those with a complex solution – or those for which a fast solution is required – I think Java makes more sense. For all other problems, I think Graphplan is the way to go.