**《 智能系统 》实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学号** | | **姓名** | **承担任务** | | | **贡献度** | **得分** |
| **20194053** | | **王志鹏** | **完成可信度推理机代码、报告撰写** | | |  |  |
| **20194051** | | **王良厅** | **完成可信度推理机代码** | | |  |  |
| **20184199** | | **王赛宇** | **完成模糊推理代码、报告** | | |  |  |
|  | |  |  | | |  |  |
|  | |  |  | | |  |  |
|  | |  |  | | |  |  |
|  | |  |  | | |  |  |
| **实验题目** | **推理机设计** | | | | | | |
| **实验时间** |  | | | **实验地点** |  | | |
| **实验成绩** |  | | | **实验性质** | **□验证性 □设计性 ☑综合性** | | |
| 教师评价：  □算法/实验过程正确 □源程序/实验内容提交  □程序结构/实验步骤合理 □实验结果正确  □语法、语义正确 □报告规范  其他：  评价教师签名： | | | | | | | |
| 1. 实验目的   为实现十字路口红绿灯智能控制，本次实验的目的是：  （1）了解推理原理与机制  （2）掌握可信度推理方法  （3）掌握模糊推理方法  （4）设计并实现推理机 | | | | | | | |
| 1. 实验项目内容   1、设计并实现可信度推理机  （1）设计可信度推理机；  （2）实现可信度推理机；  （3）完成可信度推理机测试：测试“传感器采集——传输——数据库——知识库——推理机——数据库——传输——传感器响应”的全过程；  （4）对可信度推理机设计与实现给出合理的解释或说明。  2、设计并实现模糊推理机  （1）设计模糊推理机；  （2）实现模糊推理机；  （3）完成模糊推理机测试：测试“传感器采集——传输——数据库——知识库——推理机——数据库——传输——传感器响应”的全过程；  （4）对模糊推理机设计与实现给出合理的解释。  3、设计并实现相关软件界面。 | | | | | | | |
| 三、实验过程或算法（代码）   1. 可信度知识推理：   流程图如下：    由于在程序当中需要将初始值控制在[0,1]的范围之内，因此我们按照如下定义来处理：  可信度推理机实现代码如下：   1. def confident\_infer(rain, snow, wind, lightTime): 2. print("chushi:",lightTime) 3. rain = max(0, min(100, rain)) 4. snow = max(0, min(100, snow)) 5. wind = max(0, min(100, wind)) 6. cf\_p = dict() 7. cf\_p['下雨'] = rain / 100 8. cf\_p['下雪'] = snow / 100 9. cf\_p['大风'] = wind / 100 10. newdict = {} 11. entitys = dbUtil.findKLList(db) 12. for index, element in enumerate(entitys): 13. premise = element.E 14. conclusion = element.H 15. val = cf\_p[premise] 16. if val >= element.CFE: 17. newdict[conclusion] = val \* element.CFHE 18. print(newdict) 19. maxkey="" 20. for key, value in newdict.items(): 21. if (value == max(newdict.values())): 22. maxkey = key 23. if (maxkey == "绿灯时间+3s"): 24. print(maxkey) 25. for i in range(len(lightTime)): 26. lightTime[i] += 3 27. elif (maxkey == "绿灯时间+5s"): 28. print(maxkey) 29. for i in range(len(lightTime)): 30. lightTime[i] += 5 31. elif (maxkey == "绿灯时间+6s"): 32. print(maxkey) 33. for i in range(len(lightTime)): 34. lightTime[i] += 6 35. print("zuihou",lightTime) 36. return lightTime   在上述代码中，我们首先初始化了3类前提的阈值，然后将输入的条件与阈值相比较，只有当条件可能性大于阈值，我们才考虑进行可信度的推理。随后我们遍历整个知识库，将每条知识都做相应的可信度推理，并将其结果存入一个字典。结束知识遍历之后，我们从字典当中选择值最大的，也就是可能性最高的结论作为我们可信度推理的最终结论，并将修改后的红绿灯时长传送给前台。   1. 模糊推理   模糊推理的基本方法是：根据车道上的车辆的数量，进行模糊权重估计，根据权重来分配一个总的通勤时常。为了避免通勤时常过短，还加入了一个基础的通勤时常。总体代码如下图所示；   1. *# 计90s为一次通行周期，根据权重分配通行时间.* 2. *# 基础通行时间为5s，在5s基础上累加.* 3. total\_time = 60 4. basePassTime = 5 5. fuzzy\_handler = fuzzy.get\_default\_fuzzy() 6. topRightWeight = fuzzy\_handler.get\_result\_by\_input(get\_fuzzy\_name\_by\_car\_number(topRight)) 7. eastLeftWeight = fuzzy\_handler.get\_result\_by\_input(get\_fuzzy\_name\_by\_car\_number(eastLeft)) 8. eastRightWeight = fuzzy\_handler.get\_result\_by\_input(get\_fuzzy\_name\_by\_car\_number(eastRight)) 9. topLeftWeight = fuzzy\_handler.get\_result\_by\_input(get\_fuzzy\_name\_by\_car\_number(topLeft)) 10. total\_weight = topLeftWeight + topRightWeight + eastLeftWeight + eastRightWeight 11. pass\_times = [ 12. basePassTime + (total\_time \* topRightWeight / total\_weight), 13. basePassTime + (total\_time \* eastLeftWeight / total\_weight), 14. basePassTime + (total\_time \* eastRightWeight / total\_weight), 15. basePassTime + (total\_time \* topLeftWeight / total\_weight), 16. ]   其中推理过程如下方代码所示；   1. def compute\_fuzzy\_result(input\_membership\_vec, matrix\_r): 2. """ 3. compute fuzzy result of fuzzy reasoning. 4. example: 5. input\_membership\_vec: [0.8, 0.2, 0, 0] 6. matrix\_r: [ 7. [0.8, 0.2, 0, 0], 8. [0.2, 0.5, 0.5, 0], 9. [0, 0.5, 0.5, 0.2], 10. [0, 0, 0.2, 0.8] 11. ] 12. return: 0.2 = 0.8 \* 0 + 0.2 \* 1 + 0 \* 2 + 0 \* 3 13. """ 14. res = 0 15. for inx\_vec, val\_vec in enumerate(input\_membership\_vec): 16. min\_result = [min(val\_vec, row[inx\_vec]) for row in matrix\_r] 17. res += inx\_vec \* max(min\_result) 18. return res 19. class Fuzzy: 20. def \_\_init\_\_(self, map\_input\_2\_membership\_vec, matrix\_r): 21. """ 22. init create a fuzzy object from orm.Fuzzy items 23. """ 24. self.\_\_input2vec = json.loads(map\_input\_2\_membership\_vec) 25. self.\_\_r = json.loads(matrix\_r) 26. def get\_result\_by\_input(self, in\_name): 27. """ 28. get fuzzy result by input name. 29. example: 30. :in\_name: "small" 31. :return: 0.2 32. """ 33. if self.\_\_input2vec.get(in\_name) is None: 34. raise "input name not found in map\_input\_2\_membership\_vec" 35. return compute\_fuzzy\_result(self.\_\_input2vec[in\_name], self.\_\_r) | | | | | | | |
| 1. 实验结果及分析 2. 可信度知识推理：            1. 模糊推理   本测试在假设天气良好的时候进行，即：可信推理结果恒定， 不会对最终结果产生影响，最终结果就是模糊推理的输出。  最初状态下，各个车道的车流量相似，推理时间也相同：    接下来，两车道的车辆数明显不同，车辆的通行时间也有了明显差别： | | | | | | | |
| 1. 完成时间   （1）实验时间：2022.4.29，2021.5.8  （2）检查时间：2022.5.8  （3）2022年5月10日23:59之前提交实验报告 | | | | | | | |