Dpto. Sistemas Informáticos y Computación
Escuela Técnica Superior de Ingeniería Informática
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

INTELLIGENT SYSTEMS

LAB WORK 1

DESIGN, IMPLEMENTATION AND EVALUATION OF A RULE-BASED SYSTEM

- Problem to solve -

September 2016

We have a robot, several lampposts with blown bulbs and a bulb warehouse, lying on a grid as shown in Figure 1. Each lamppost may have several blown bulbs. The objective is that the robot loads bulbs at the warehouse to replace blown bulbs. The robot must not carry bulbs when all blown bulbs have been replaced.

An example of this initial state is shown in Figure 1. The robot and the warehouse can be located anywhere on the grid; the number of lampposts, their location as well as the number of blown bulbs in each lamppost can be different for a given initial state. The size of the grid can be defined for a given initial state.

The solution to this problem does not need to represent the number of bulbs in each lamppost, only those that have blown. In Figure 1, there are three lampposts with 3, 2 and 2 blown bulbs.

The robot can only move horizontally and vertically, at each movement the robot moves from one cell to the next. There is a maximum number of bulbs the robot can carry. That maximum number of bulbs is the maximum number of blown bulbs in a lamppost at the initial state. For example in Figure 1, the maximum number of bulbs that the robot can carry is 3.

When the robot is at the warehouse, it can load several bulbs without exceeding the maximum number of bulbs. The loading operation must be performed with a single rule loading all bulbs needed at the same time. Bulbs cannot be loaded one by one.

When the robot arrives to a lamppost with blown bulbs, it will be able to fix the lamppost if it is carrying at least as many bulbs as blown bulbs. The replacement of all the blown bulbs in a lamppost is performed with a single rule. It is not possible to replace only some of the blown bulbs in a lamppost, all of them must be replaced at the same time. For example, in Figure 1, the lamppost in cell (4,2) has 2 blown bulbs, so if the robot is carrying 2 or 3 bulbs, then it will replace all blown bulbs with a single rule.

The cost of each movement over the grid, and loading/replacing all bulbs is 1.

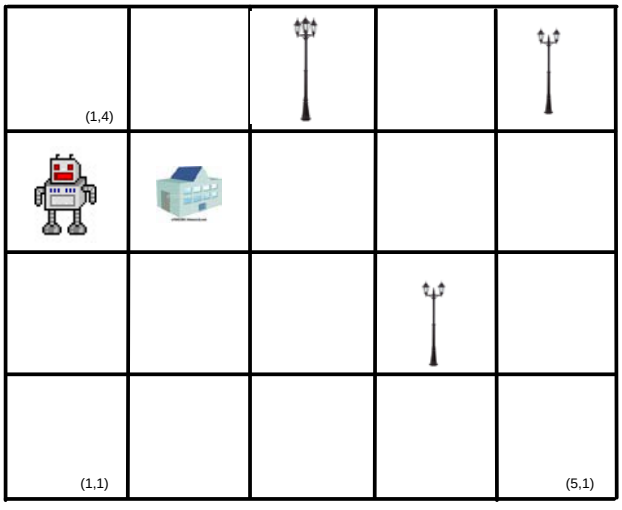

Fig. 1. Example of initial state

SECTION A: RULE-BASED SYSTEM IN CLIPS

1) Design a rule-based system (graph exploration) that solves the previous problem. Describe the domain of the fact list using patterns, specifying the structure of the facts that will be employed, and the set of rules according to the representation selected. Do not assign priority to rules except for the objective rule.

2) Perform different evaluation tests to compare the results obtained using breadth and depth search. The evaluation tests must consider different number of lampposts, number of blown bulbs, and location of the robot and lampposts. For each evaluation test, the number of generated nodes and the depth at which the solution is found must be reported. One of these evaluation tests must be that of Figure 1.

Aspects to take into account in the design:

a) We assume that the cost of each operation is 1.

b) Each rule performs a single operation, that is, it is not possible to combine a movement rule over the grid with an operation to load/replace bulbs. In addition, bulbs are loaded/replaced in a single operation. The rule to load/replace bulbs must be general enough to handle any number of bulbs in a single operation.

c) The system must return the depth level at which the solution is found. In this way, the breadth strategy must return the shortest solution, that is, that with the minimum number of operations. For example, for the initial state in Figure 1 in which the robot is not carrying bulbs, the optimal solution requires 21 operations.

It is not needed to compute the sequence of operations to reach the final state, ONLY THE NUMBER OF OPERATIONS TO REACH THE FINAL STATE AND THE NUMBER OF GENERATED NODES.

d) The knowledge representation and the rule design must be **GENERAL** in order to be modified easily. **The rules must be as general as possible**. More precisely:

   1) The size of the grid can be modified, although that size will not change over the execution of an instance of the problem. That is, the size of the grid can be modified at the initial state, but the rules must remain the same.
   2) Do not use procedural code on the right-hand side of rules, except for those commands to print out messages on the terminal, counting the number of generated nodes or halting the execution.

e) It is not necessary to implement a function to request the user information on the number of lampposts, maximum number of bulbs to carry, size of the grid, etc. That information can be directly provided as static facts.

f) The program must request the user to provide the maximum depth in the execution (see the solution to the puzzle problem for further details).

SECTION B: IMPLEMENTATION OF A HEURISTIC FUNCTION

Let us implement a heuristic function for the problem described in Section A using an A algorithm f(n) = g(n)+h(n), where:

1) g(n) is the number of operations performed from the initial state to reach state n. The operations that are considered are: movements over the grid and load/replace bulbs.
2) h(n) is the estimation of the number of operations to reach the final state from state n.

Modify the design performed in Section A to incorporate a heuristic function that computes f(n).

How to compute the heuristic function h(n):

If the robot is not carrying bulbs, the heuristic function assumes that the robot will go to the warehouse to load bulbs. Next, the robot would go to the first lamppost to replace the blown bulbs and would go back to the warehouse to load bulbs. So on and so forth until the last lamppost, that after replacing the bulbs, the robot does not need to go back to the warehouse.

In this way, the cost of the heuristic function would sum the following values:

1. The distance from the robot to the warehouse.
2. For each lamppost with blown bulbs (except for the last one), the distance from this lamppost to the warehouse and back.
3. For the last lamppost, the distance from the warehouse to this lamppost.
4. The cost of every loading/replacing operation. The cost of a single operation is 1.

You do not need to take into account if the number of bulbs loaded or replaced is appropriated.

If the robot is carrying bulbs, then the sum defined above must exclude the first item, since the robot will directly go the lamppost.

The distance between two objects A and B is the Manhattan distance:

$$d(A,B) = |A_x - B_x| + |A_y - B_y|$$

In Figure 1, assuming that lampposts are visited in the following order (4,2), (3,4) and (5,4), the value of the heuristic function h(n) would be:

- Distance from the robot to the warehouse: 1.
- Loading bulbs: 1
- Distance from the warehouse to the first lamppost (4,2), replace bulbs, back to the warehouse and load bulbs: 3 + 1 + 3 + 1
- Distance from the warehouse to the second lamppost (3,4), replace bulbs, back to the warehouse and load bulbs: 2 + 1 + 2 + 1
- Distance from the warehouse to the third lamppost (5,4) and replace bulbs: 4 +1

Total h(n)= 21

Compare the depth of the solution and the number of generated nodes in Section A with breadth and depth strategies to those obtained with the heuristic function. Discuss the properties of the heuristic function h(n).