Max O’Cull
0028687211
CS 471
Homework 1
01/23/19

1. Give a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.
   a. Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.

      States: A state description specifies a planar map containing *n* regions placed adjacent to each other. All regions can have one of 5 values: red, green, blue, yellow, or uncolored (colors are provided for sake of simplicity, not by requirement).

      Initial state: All regions are set to uncolored.

      Actions: ColorRed, ColorGreen, ColorBlue, ColorYellow are all possible actions which color a specified region to it’s aforementioned color.

      Goal test: All regions are colored, with no adjacent regions having the same color anywhere in the map.

      Path cost: Each step costs 1, so the path cost is the number of steps in the path.

   b. A 2-foot-tall giraffe is in a room where some peaches are suspended from the 7-foot ceiling. He would like to get the peaches. The room contains two stackable, movable, climbable 3-foot-high crates.

      States: A state description specifies a room containing peaches, a giraffe, and 2 crates. The crates may be relocated, stacked, and climbed by the giraffe. The peaches are stationary, and the giraffe may move about. Any permutation of these is a valid state.

      Initial state: A room containing only 2 crates that are 3 feet high, some peaches suspended on the 7 foot ceiling, and a 2 foot tall giraffe.

      Actions: WalkTo, ClimbOnCrate, ClimbOffCrate, StackCrate, UnstackCrate, PushCrateTo, EatPeaches are all possible actions which interact with the specified entity (onto the specified object, in the case of all but WalkTo).

      Goal test: The giraffe can reach and eat the peaches.

      Path cost: Each action costs 1, so the path cost is the number of actions taken in the path.

   c. You have a program that outputs the message “*illegal input record”* when fed a certain file of input records. You know that processing of each record is independent of the other records. You want to discover what record is illegal.

      States: A state description specifies a set of input records which are unprocessed, legal, possibly illegal, or illegal. Variable sized batches of records may be processed in a single step. If a batch receives an error message, the whole batch is possibly illegal unless the batch is of size 1 (in which case it’s illegal); otherwise, the records are legal. Any permutation of these settings is a valid state.

Initial state: All records have yet to be processed.

Actions: SplitRecords which takes all records and splits them into two evenly sized groups, and ProcessGroup which takes a specified group and processes all contained records.

Goal test: While considering a single record, it returns “illegal input record.”

Path cost: Each action costs 1, so the path cost is the number of actions taken in the path.

d. You have three buckets, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You can fill the buckets up or empty them out from one to another or onto the ground. You need to measure out exactly one gallon.

States: A state description specifies a set of buckets of size 12, 8, and 3 gallons. The buckets may hold any amount of water between 0 and their respective capacity. Any permutation of these settings is a valid state.

Initial state: All 3 buckets hold no water.

Actions: FillBucket which maximizes the water contained within a bucket, EmptyBucket which sets the water contained in the bucket to 0, and PourFromTo. PourFromTo transfers water from a specified bucket to another. The amount of water transferred is equal to the minimum of the water held in total by both buckets and the receiving bucket’s capacity. The pouring bucket’s amount of water is subtracted by the amount transferred.

Goal test: One of the 3 buckets measures exactly 1 gallon. The other buckets need not be empty.

Path cost: Each action costs 1, so the path cost is the number of actions taken in the path.

2. Three dogs and three cats are on one side of a river, along with a boat that can hold one or two animals. Find a way to get everyone to the other side without ever leaving a group of cats in one place outnumbered by the dogs in that place.

a. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.

States: A state description specifies a two shores. The shores can hold an unlimited amount of animals. An animal must be present on the boat to move the boat. Cats must not be outnumbered by dogs in either shore (except when there

Max O’Cull
0028687211
CS 471
Homework 1
01/23/19

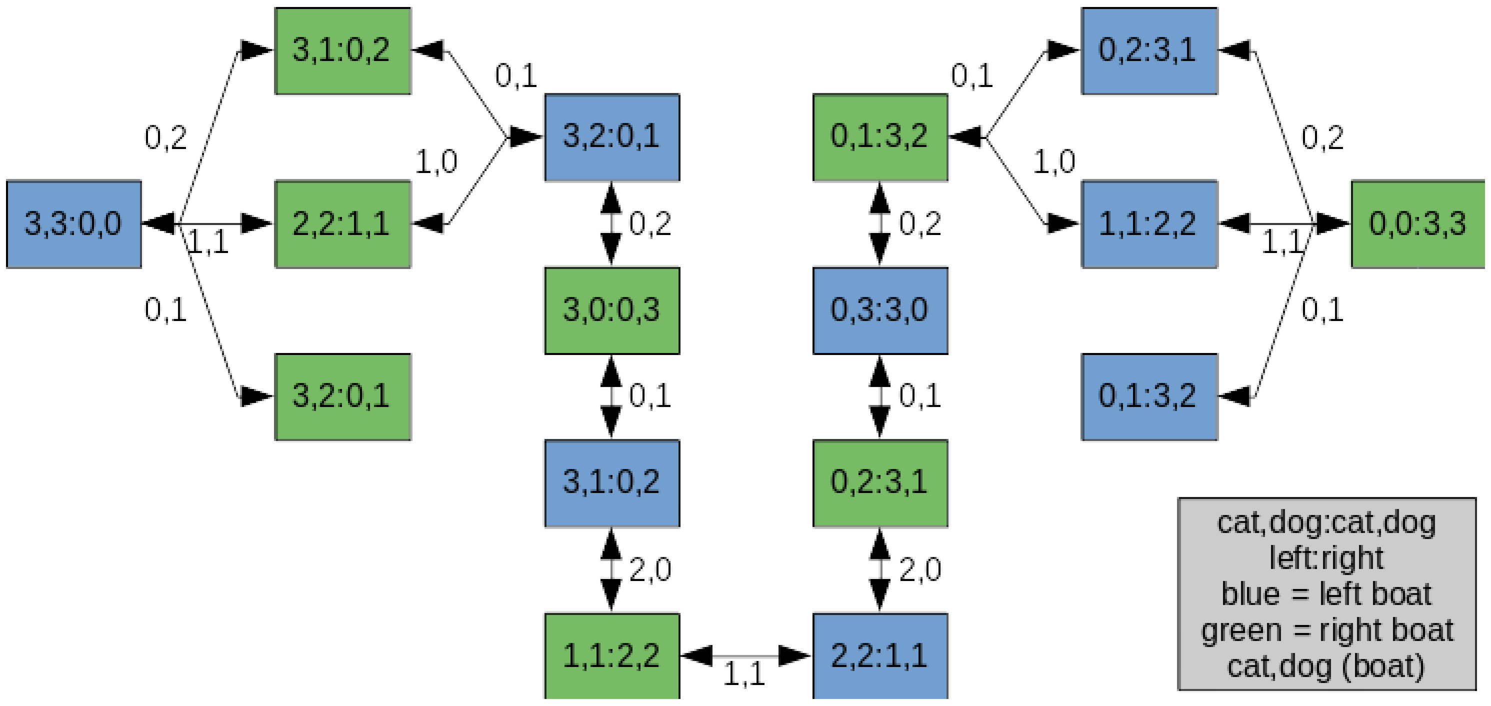
are no cats on that shore). Any permutation of these settings is a valid state. We do not need to consider the boat other than what side it is on in our states because there are no invalid (combination of animals in) boat states, and because we assume no-ops do not occur.

Initial state: All dogs and cats are on the left side of the river, and the boat between the land masses is empty.

Actions: RowRight which moves a specified 1 or 2 (but not 0) animals on the boat to the right shore as long as the boat is on the left shore, RowLeft which moves a specified 1 or 2 (but not 0) animals on the boat to the left shore as long as the boat is on the right shore.

Goal test: All 3 dogs and 3 cats are on the right shore.

Path cost: Each action costs 1, so the path cost is the number of actions taken in the path.



b. When implementing a solution to this problem, is it a good idea to check for repeated states?

Yes, because it will greatly reduce the size of the search.

c. Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

Max O'Cull
0028687211
CS 471
Homework 1
01/23/19

It is difficult to realize what parts are necessary and which are not. It is also not trivial to realize that many states are repeated or lead to invalid scenarios, and as such may be excluded from evaluation.

3. Answer the following questions regarding local search.
    a. Explain the difference between local search and greedy best-first search on a graph.

       Local search algorithms aim to solve optimization in wide spaces by intelligently analyzing neighbors, whereas greedy best-first search algorithms aim to efficiently determine optimal paths in graphs by selecting promising nodes
    b. Give an example, providing a specific graph, where local search is better or explain why this is impossible.

       Local search would tend to do well in a graph that is a billion nodes wide (where adjacent nodes are similar in value) but 3 nodes deep. Analyzing neighbors will be beneficial to the local search, but in best-first search it will struggle because there are many candidates that must be evaluated to determine which is ideal for moving forward with.
    c. Give an example, providing a specific graph, where local search is worse or explain why this is impossible.

       In a graph that is a million wide and a million deep that represents a geographical map, best-first search will likely compute faster because not all paths that run depth-wise will need to be examined. Local search will spend much time analyzing the first few layers by proximity to neighbors.
    d. Give an example where local search is better than A* search. Explain why this is possible even though A* is optimal.

       As in 3.b., a local search would perform better than A* because A* would need to evaluate most, if not all the nodes and compute heuristics for each. While A* will achieve an optimal solution, it will be much slower than a local search.
4. Implement and solve problem 2 using iterative deepening depth-first search and also use A* with a heuristic of your choice. The initial state is defined as 3 cats and 3 dogs are on one side of the river, and they need to go to to the other side. By the end of each algorithm you need to print in the console the states from the beginning to the goal (state graph path).