New Coverage Algorithm

Charles Zhang

July 1 2020

```
options(digits = 6)
library(tictoc)
```

Environment Example(7x8):

"""

1 — 2 —3 — 4 — 5 —6 —7 —8

9 —10—11—12—13—14—15—16

17—18—19—20—21—22—23—24

25—26—27—28—29—30—31—32

33—34—35—36—37—38—39—40

41—42—43—44—45—46—47—48

49—50—51—52—53—54—55—56

"""

Global variable

```
ROWS = 20
COLS = 8
START = 1
Reward = 100
```

```
V = ROWS * COLS
S = 1:V
R = matrix(-1, V, V)
# four cornors
R[1, c(2, 1+COLS)] = 0
R[COLS*(ROWS-1)+1, c(COLS*(ROWS-1)+1-COLS, COLS*(ROWS-1)+2)] = 0
R[COLS, c(COLS-1, 2*COLS)] = 0
R[V, c(V-1, V-COLS)] = 0
# four boundary edges
for(i in 2:(COLS - 1)) {
  R[i, c(i-1, i+1, i+COLS)] = 0     # up edge
  R[V-i+1, c(V-i+2, V-i, V-i+1-COLS)] = 0   # bottom edge
}
for(i in 1:(ROWS-2)) {
  R[i*COLS+1, c(i*COLS+1-COLS, i*COLS+2, i*COLS+1+COLS)] = 0    # left edge
  R[(i+1)*COLS, c(i*COLS, (i+2)*COLS, (i+1)*COLS-1)] = 0    # right edge
}
# inside vertices
for (i in 0:(COLS-3)) {
  for (j in 1:(ROWS-2)) {
    R[j*COLS+2+i, c(j*COLS+1+i, j*COLS+3+i, j*COLS+10+i, j*COLS-6+i)] = 0
  }
```

```
}
# give reward
# if (COLS %% 2 == 0) {
#   END = 2
#   R[2, c(1, COLS+2, 3)] = Reward
# } else if (ROWS %% 2 == 0) {
#   END = COLS+1
#   R[END, c(1, END+1, END+COLS)] = Reward
# } else {
#   END = 2
#   R[2, c(1, COLS+2, 3)] = Reward
#   print("still working on this part!")
# }
END = 2
R[2, c(1,3,12)] = Reward
Q = matrix(0, V, V)
alpha = 0.5
```

```
rounds = 1000
r = 1
get_actions <- function(s) {
  a = c()
  for (i in 1:V) {
    if(R[s,i] != -1) a = c(a, i)
  }
  return(a)
}
```

Core algorithm based on Q learning

```
tic()
while (r <= rounds) {
  s = sample(S, 1)
  while (TRUE) {
    action_space = get_actions(s)
    action <- sample(action_space, 1)
    s_next <- action
    actions_next = get_actions(s_next)
    qs = c()
    for (i in actions_next) qs = c(Q[s_next,i], qs)
    Q[s,action] <- R[s,action] + alpha * max(qs)
    s = s_next
    if (s == END) break
  }
  r <- r+1
}
```

Find Path based on Q table

```
path = c()
state = START
Q[Q == 0] <- 1000
while (length(path) < V)
```

```
{
  pre_state = state
  path = c(path, state)
  state = match((min(Q[state,])), Q[state,])
  Q[pre_state, ] = 1000
  Q[, pre_state] = 1000
}
```

Running Time

```
toc()
```

```
## 42.836 sec elapsed
```

```
path
```

```
##   [1]   1   9  17  25  33  41  49  57  65  73  81  89  97 105 113 121 129
##  [18] 137 145 153 154 155 156 157 158 159 160 152 144 136 128 120 112 104
##  [35]  96  88  80  72  64  56  48  40  32  24  16   8   7  15  23  31  39
##  [52]  47  55  63  71  79  87  95 103 111 119 127 135 143 151 150 142 134
##  [69] 126 118 110 102  94  86  78  70  62  54  46  38  30  22  14   6   5
##  [86]  13  21  29  37  45  53  61  69  77  85  93 101 109 117 125 133 141
## [103] 149 148 140 132 124 116 108 100  92  84  76  68  60  52  44  36  28
## [120]  20  12   4   3  11  19  27  35  43  51  59  67  75  83  91  99 107
## [137] 115 123 131 139 147 146 138 130 122 114 106  98  90  82  74  66  58
## [154]  50  42  34  26  18  10   2
```