# 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 = 16
COLS = 16
START = 1
Reward = 200
```

```
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+2+COLS+i, j*COLS+2-COLS+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)] = 0
  print("Not Hamiltomian Cycle so No Coverage Path!")
}
Q = matrix(0, V, V)
alpha = 0.6
```

```
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()
```

```
## 114.207 sec elapsed
```

```
path
```

```
##   [1]   1  17  33  49  65  81  97 113 129 145 161 177 193 209 225 241 242
##  [18] 243 244 245 246 247 248 249 250 251 252 253 254 255 256 240 224 208
##  [35] 192 176 160 144 128 112  96  80  64  48  32  16  15  31  47  63  79
##  [52]  95 111 127 143 159 175 191 207 223 239 238 222 206 190 174 158 142
##  [69] 126 110  94  78  62  46  30  14  13  29  45  61  77  93 109 125 141
##  [86] 157 173 189 205 221 237 236 220 204 188 172 156 140 124 108  92  76
## [103]  60  44  28  12  11  27  43  59  75  91 107 123 139 155 171 187 203
## [120] 219 235 234 218 202 186 170 154 138 122 106  90  74  58  42  26  10
## [137]   9  25  41  57  73  89 105 121 137 153 169 185 201 217 233 232 216
## [154] 200 184 168 152 136 120 104  88  72  56  40  24   8   7  23  39  55
## [171]  71  87 103 119 135 151 167 183 199 215 231 230 214 198 182 166 150
## [188] 134 118 102  86  70  54  38  22   6   5  21  37  53  69  85 101 117
## [205] 133 149 165 181 197 213 229 228 212 196 180 164 148 132 116 100  84
## [222]  68  52  36  20   4   3  19  35  51  67  83  99 115 131 147 163 179
## [239] 195 211 227 226 210 194 178 162 146 130 114  98  82  66  50  34  18
## [256]   2
```