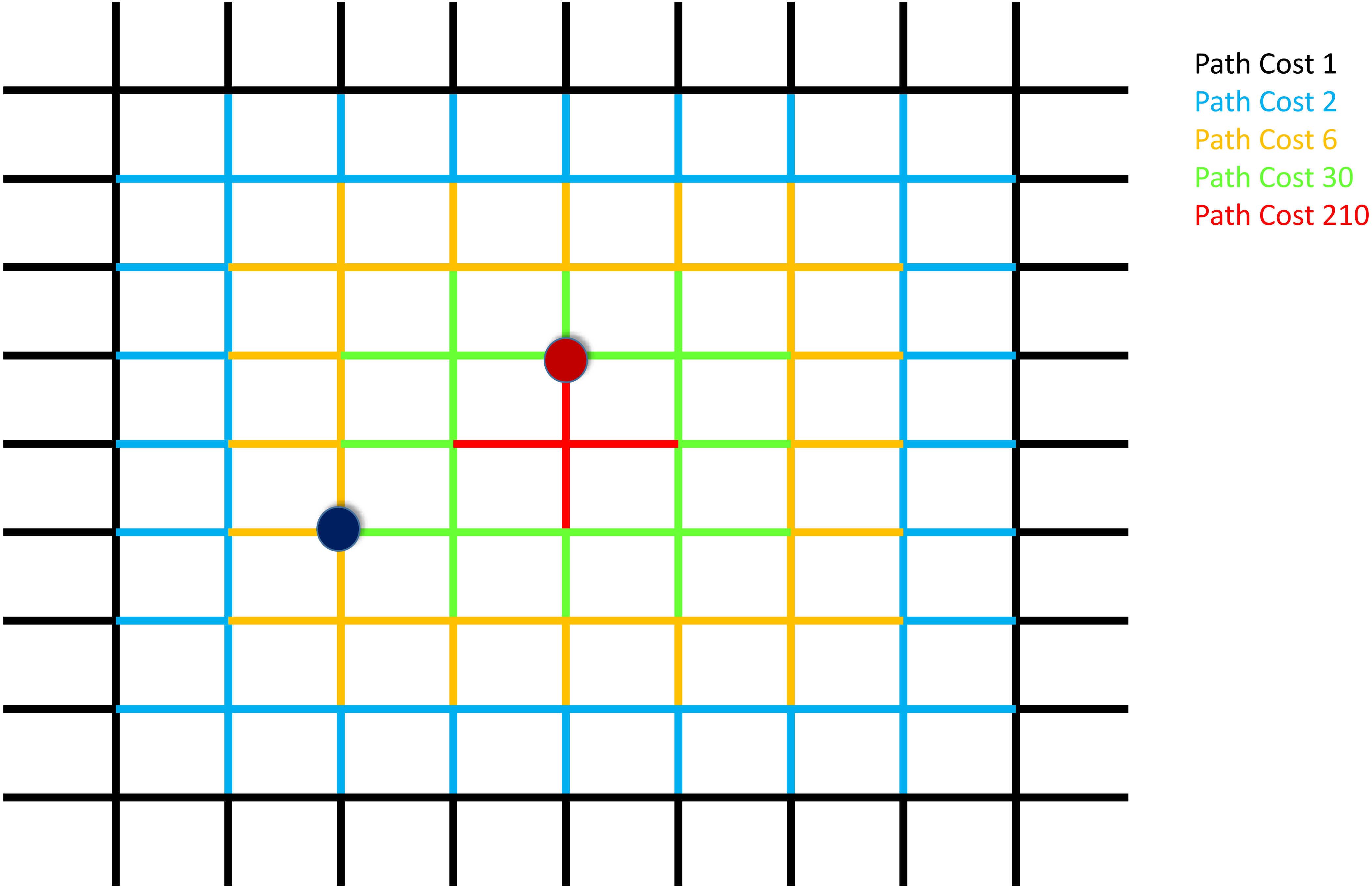
Search Algorithms

Uniform Cost Search

```
node = start
frontier = heap({node})
explored = {}
while not empty(frontier):
    node = frontier.pop()
    if IS_GOAL(node): return SOLUTION(node)
    explored.add(node)
    for action in node.get_actions():
        child = APPLY(node, action)
        if child not in union(frontier, explored):
            frontier.add(child)
        else if child in frontier:
            frontier.decide_and_replace(child)
```



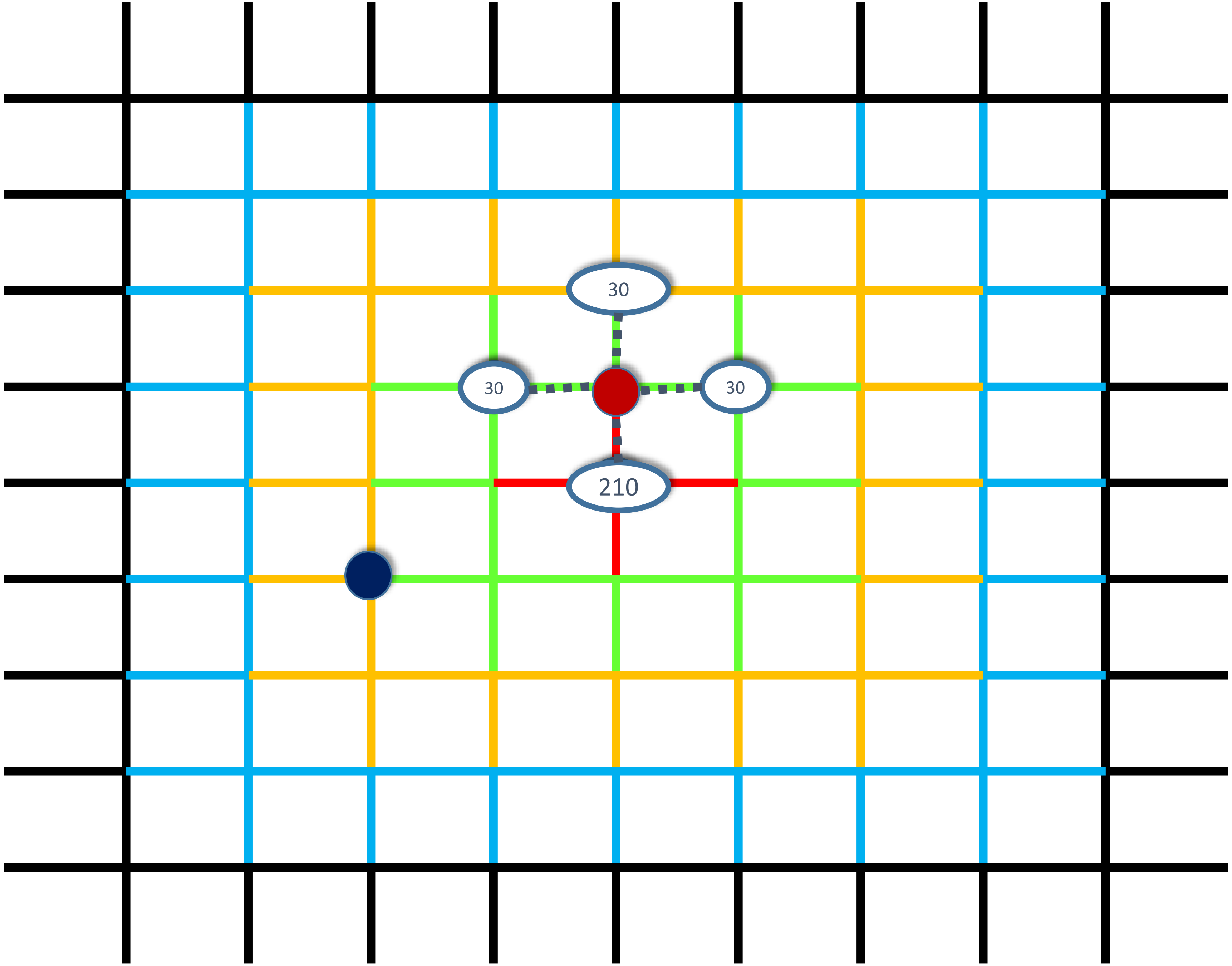
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



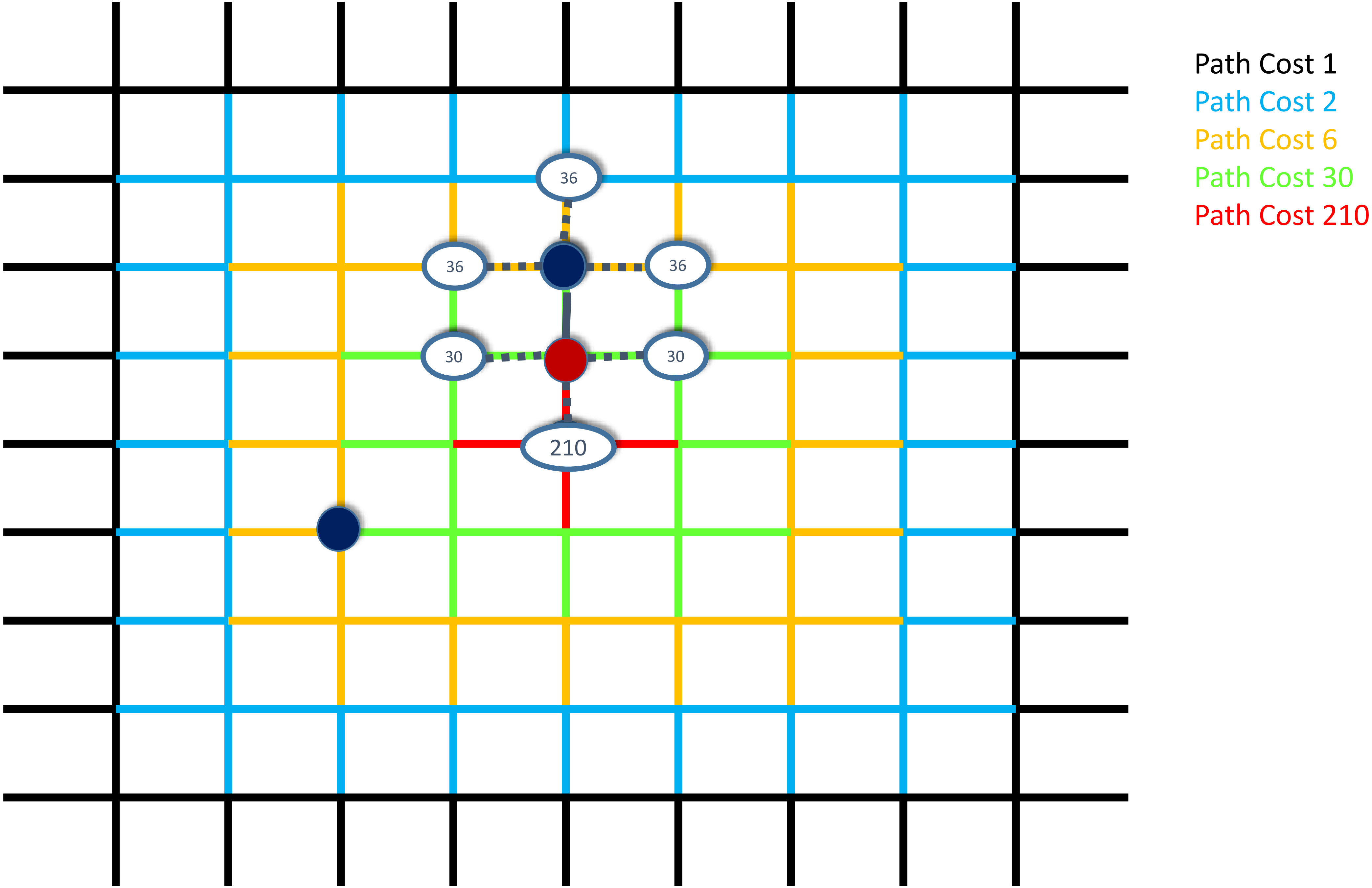
Path Cost 1

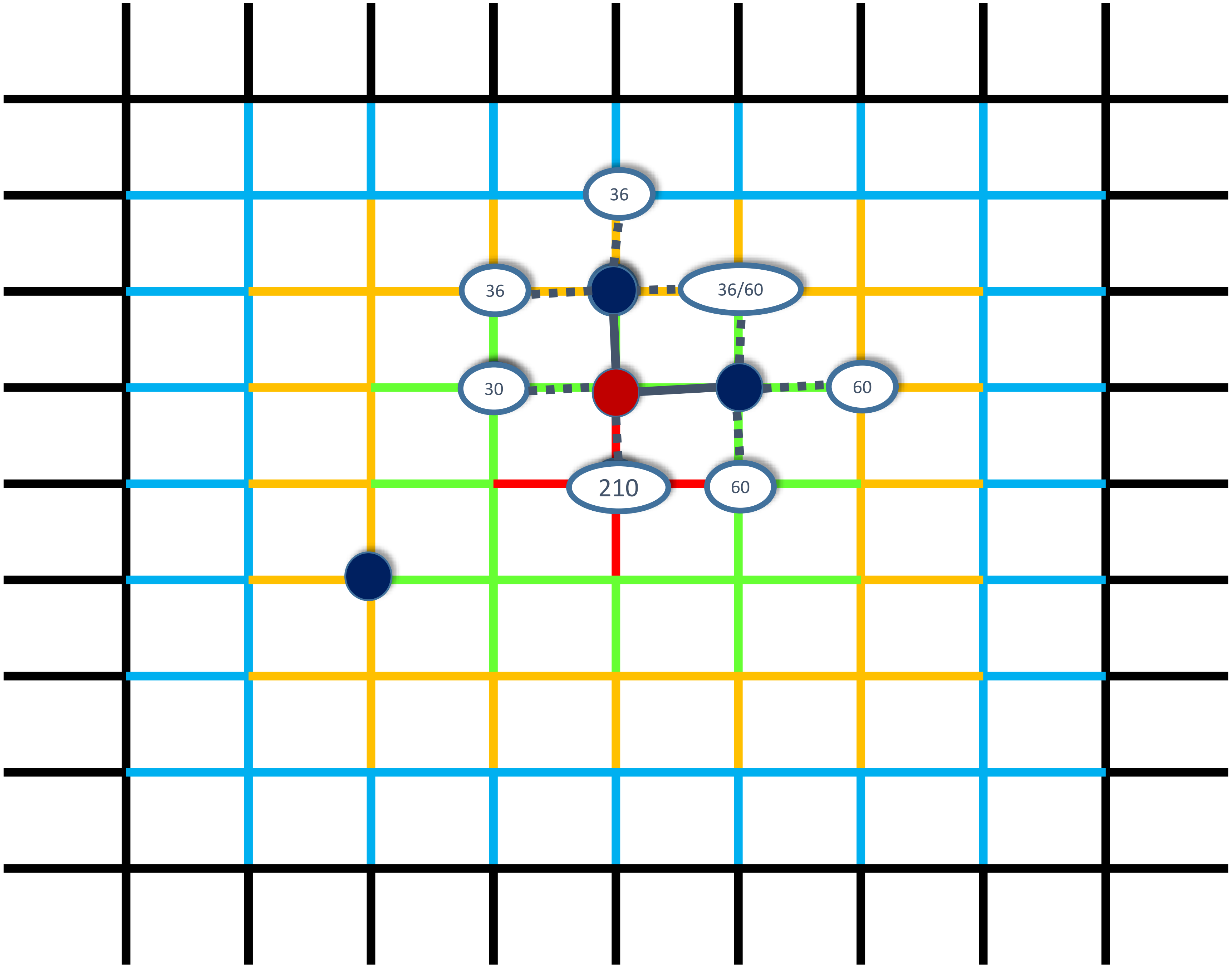
Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210


36
36
36
30
30
210
Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210



Path Cost 1

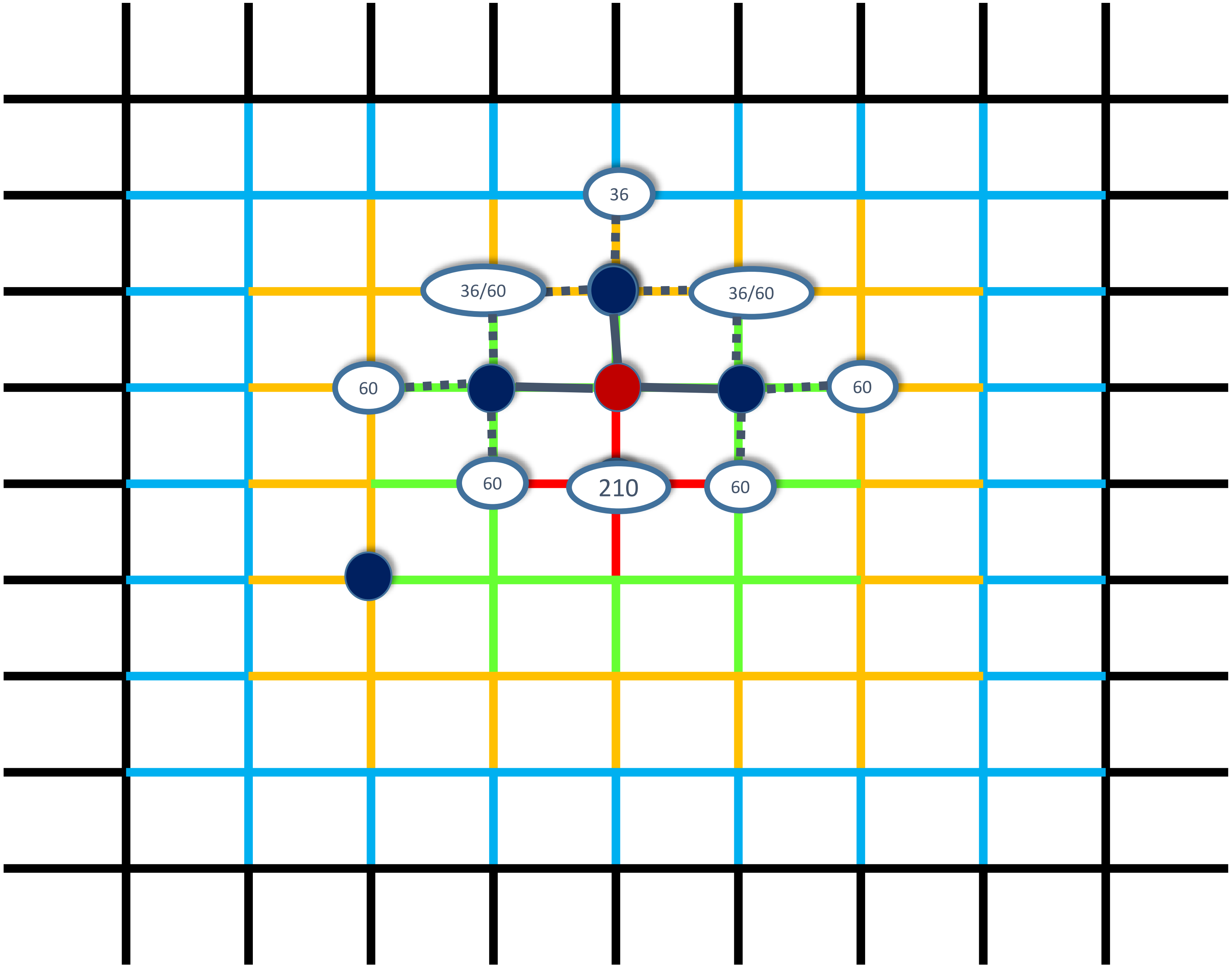
Path Cost 2

Path Cost 6

Path Cost 30

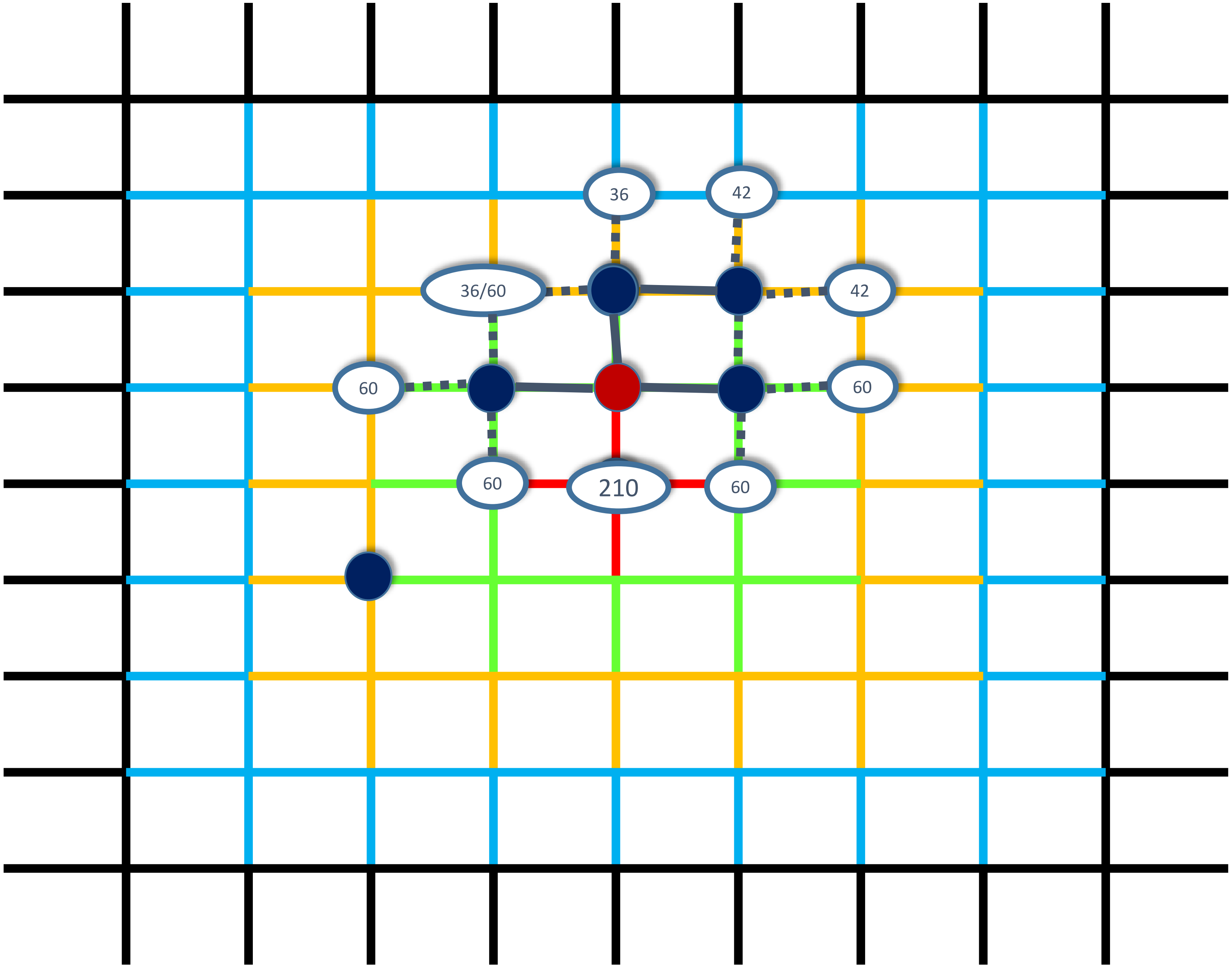
Path Cost 210

Convention:
Clockwise if tie



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210

Convention:
Clockwise if tie



Path Cost 1

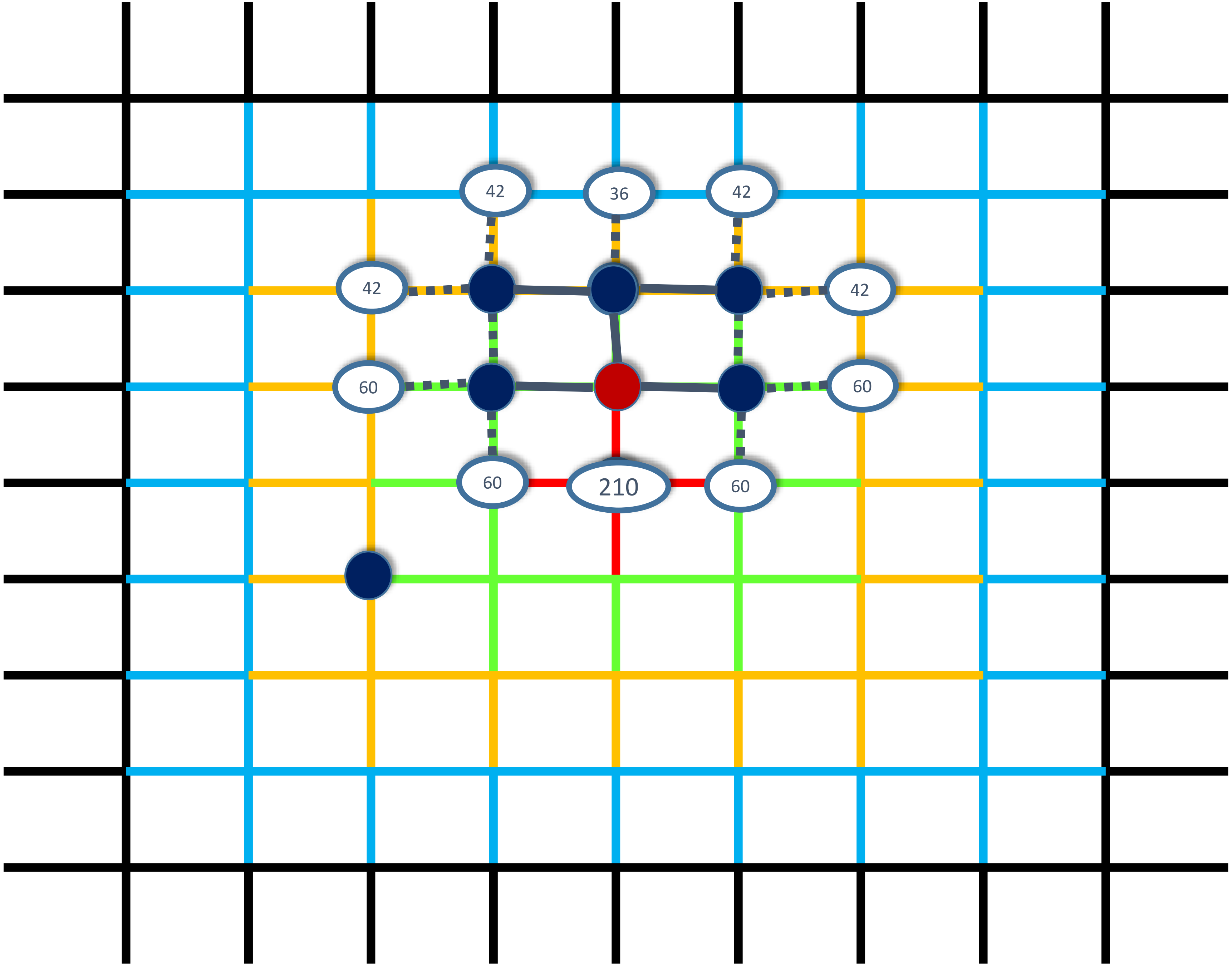
Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210

Convention:
Clockwise if tie



Path Cost 1

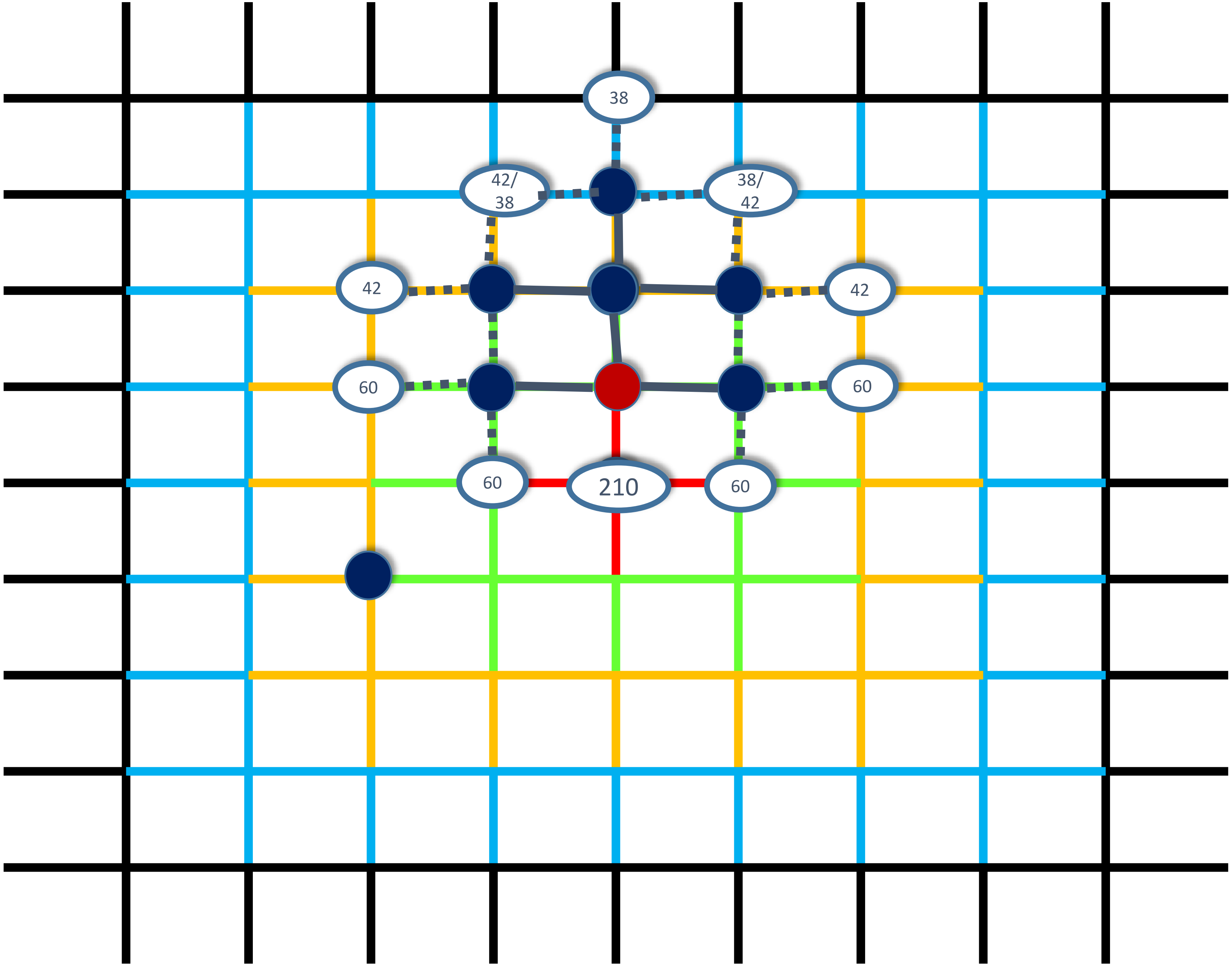
Path Cost 2

Path Cost 6

Path Cost 30

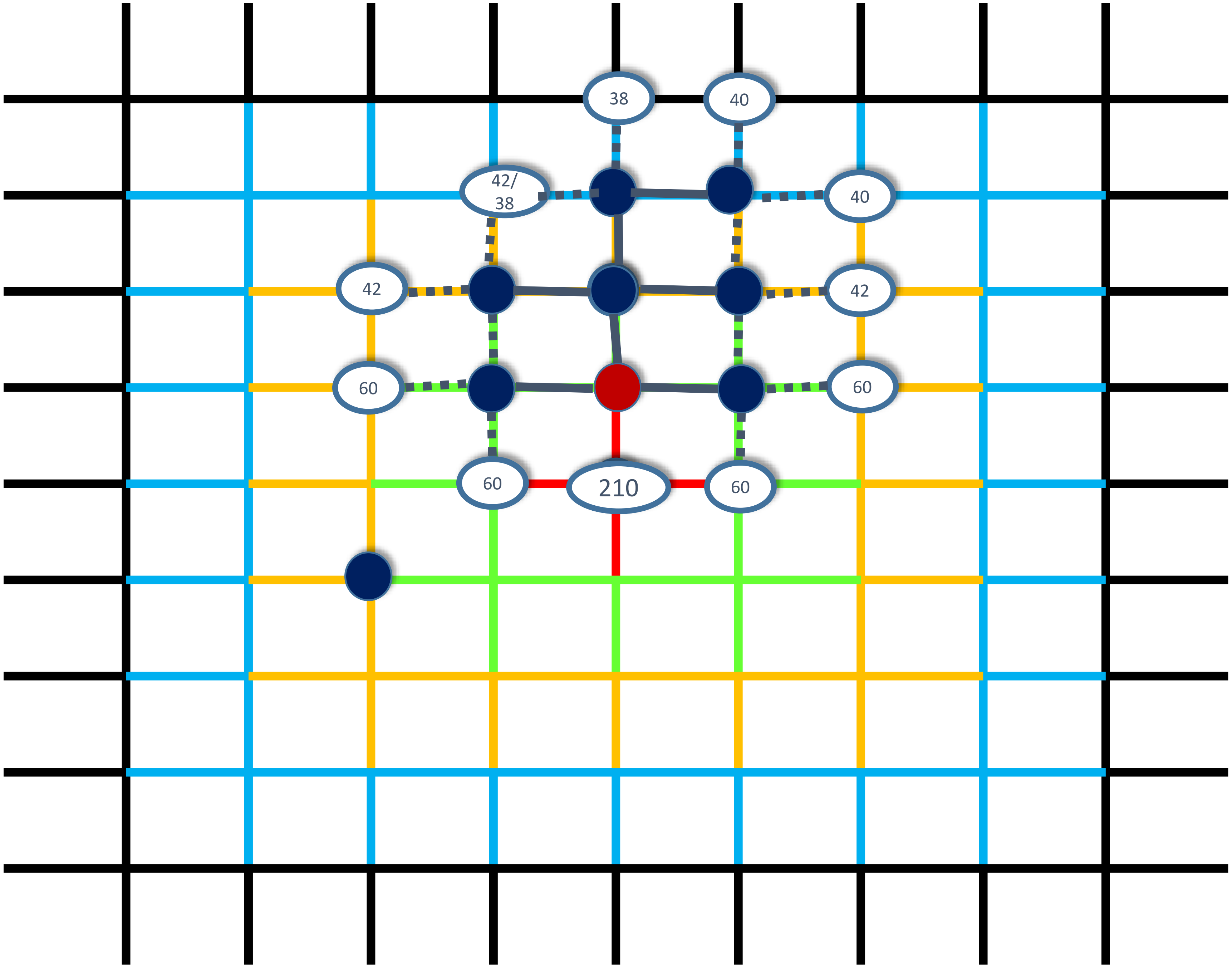
Path Cost 210

Convention:
Clockwise if tie



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210

Convention:
Clockwise if tie



Path Cost 1

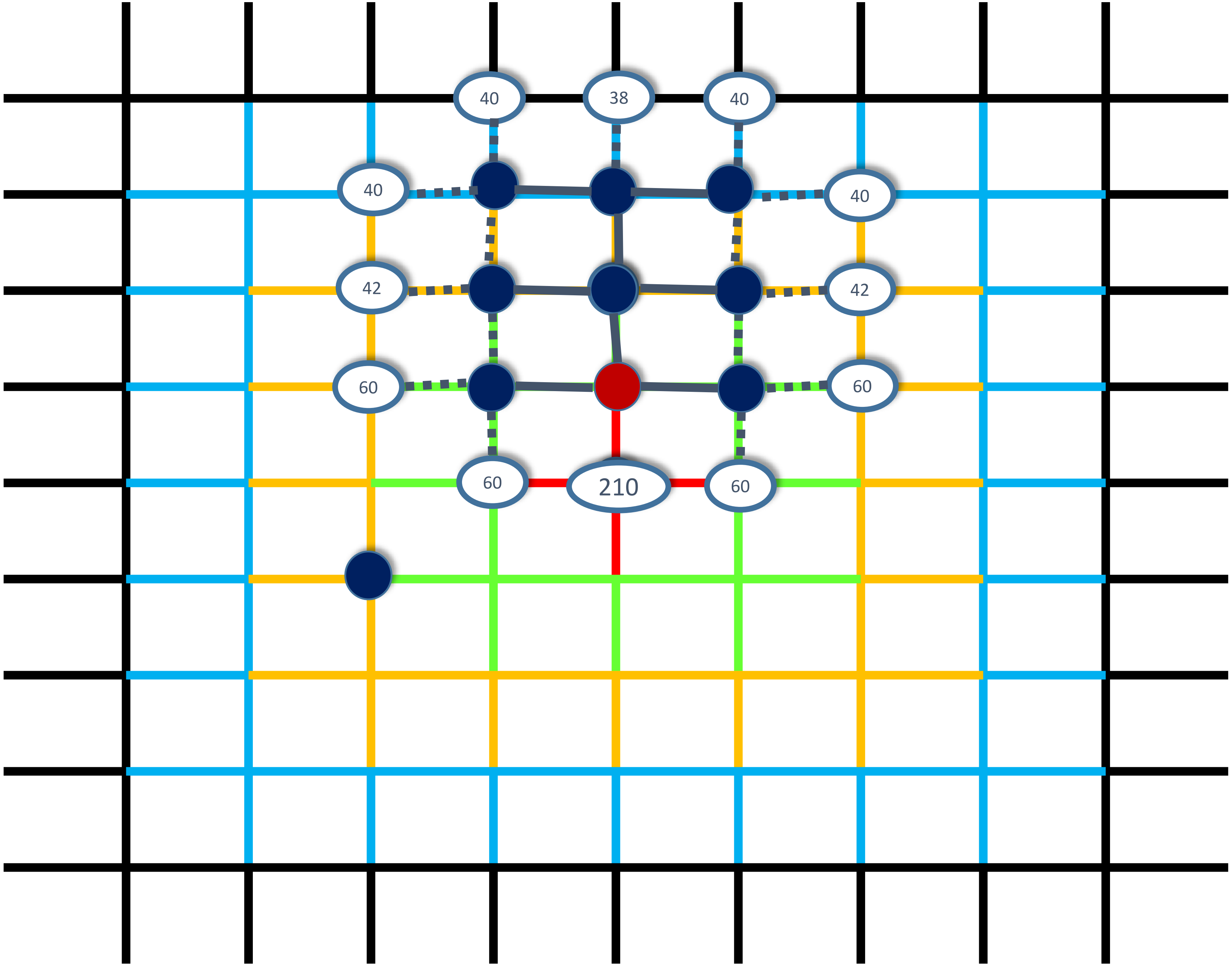
Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210

Convention:
Clockwise if tie



Path Cost 1

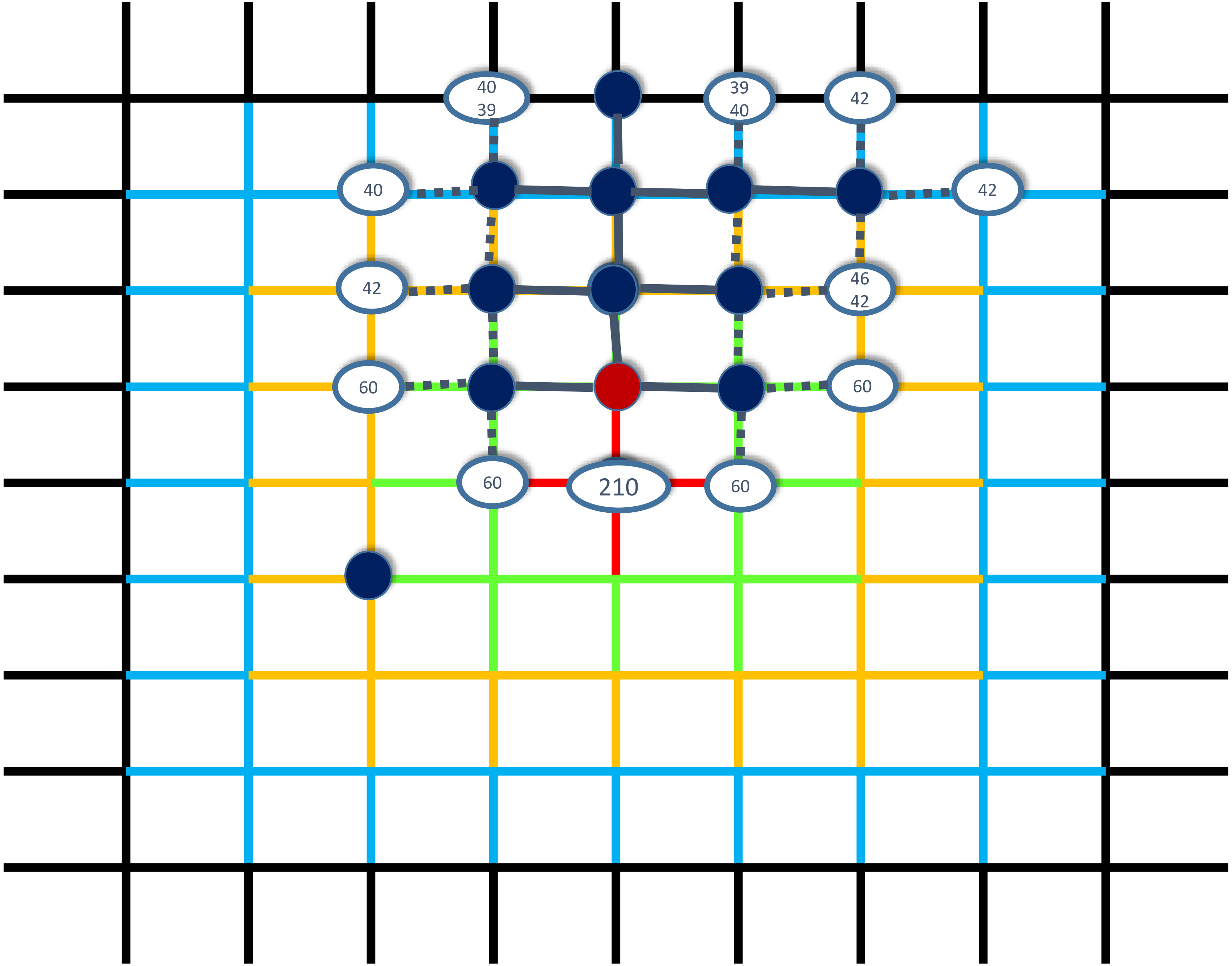
Path Cost 2

Path Cost 6

Path Cost 30

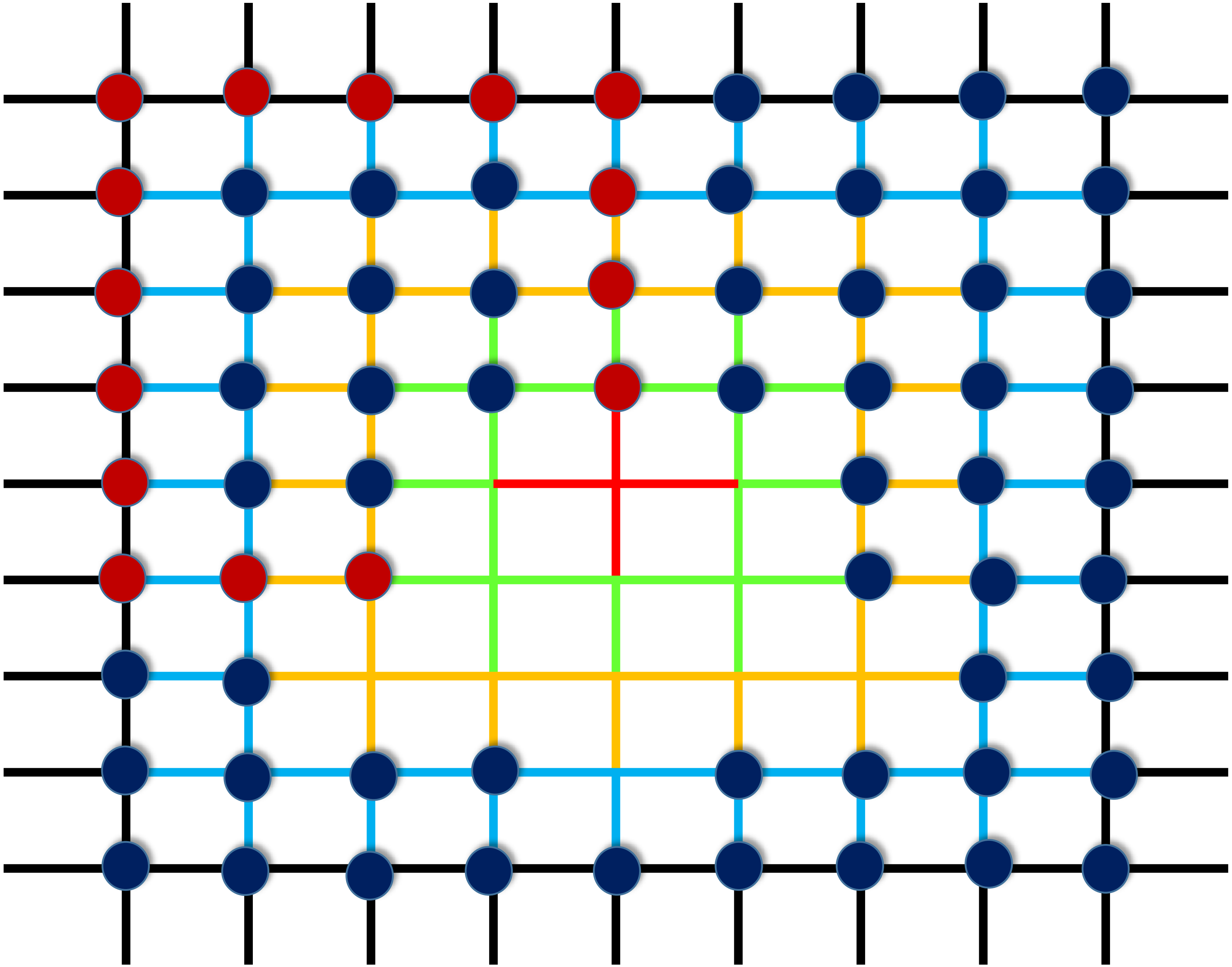
Path Cost 210

Convention:
Clockwise if tie



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210

Convention:
Clockwise if tie



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210

Convention:
Clockwise if tie

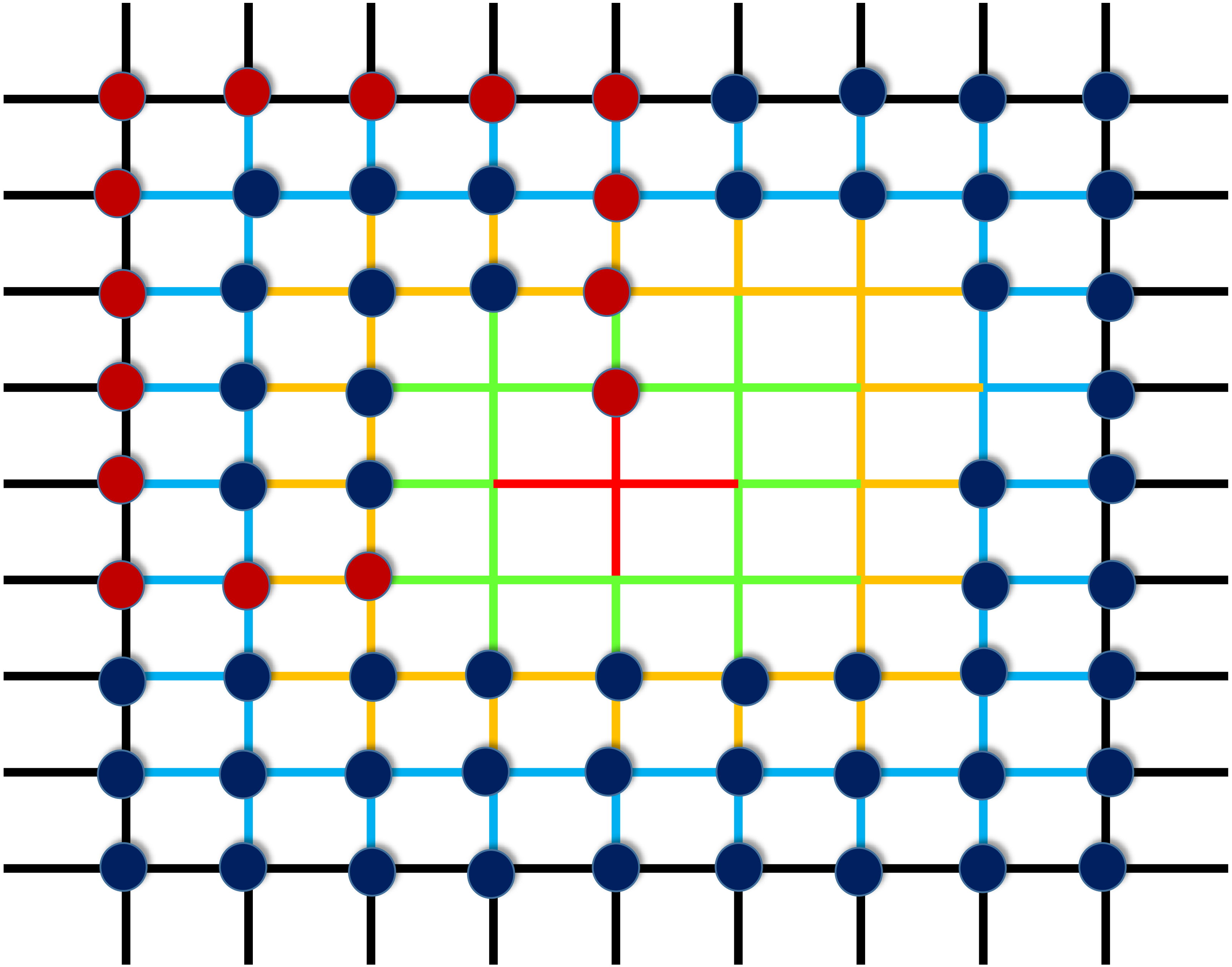
Number Explored: 69
Shortest Cost: 55

Bidirectional Uniform Cost Search

- Any manner of expanding frontiers is OK
  - Alternating both frontiers – good for parallel computing
  - Taking the min – good in weighted graphs where hubs have high cost

- Stopping criterion:
  - `min(forward) + min(reverse) > shortest_path_in_graph`

  - Note: intersection of explored sets, means you check for your stopping criterion when you POP from the queue.



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210

Number Explored: 67
Shortest Cost: 55

Notes:

- Expanding min of frontiers
- Clockwise tie breaks

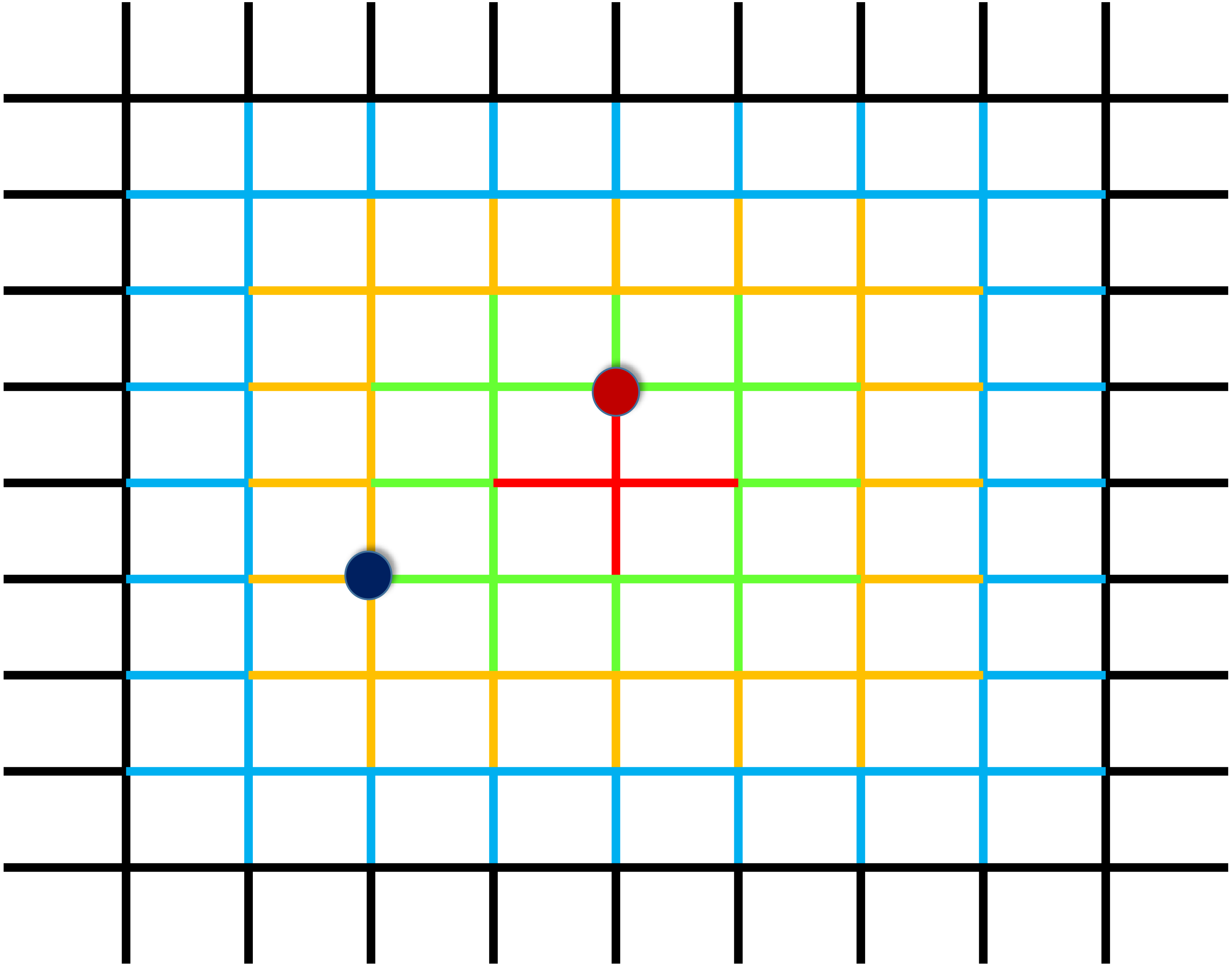
A* Search

- Change the heap sort to include a heuristic function

```
    f(state) = h(state) + g(state)
```

- Choice of a good heuristic:
  - Admissible: underestimates
  - Consistent (strict): monotonic

- The better the heuristic, the quicker the search



Path Cost 1

Path Cost 2

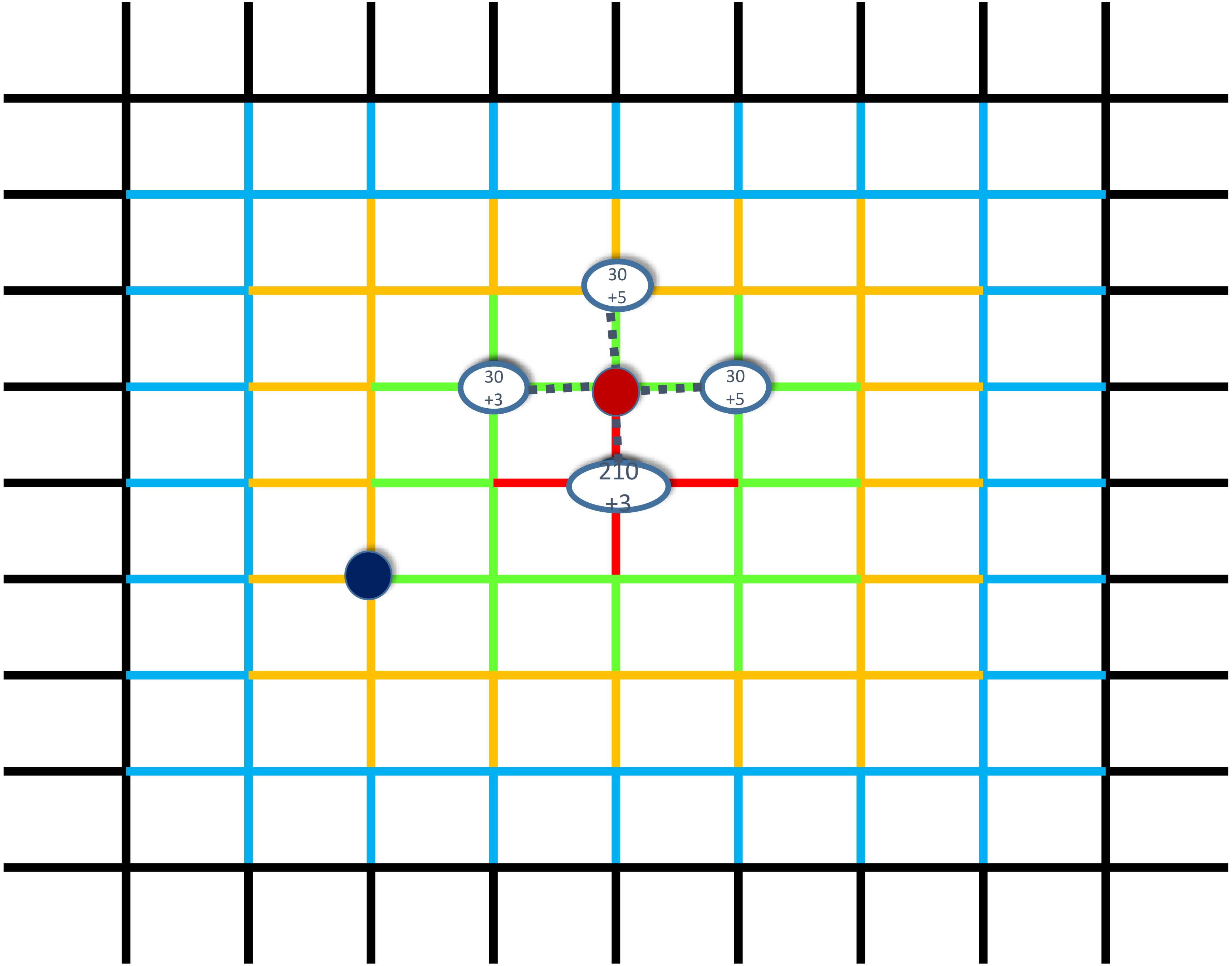
Path Cost 6

Path Cost 30

Path Cost 210

Notes:

- Manhattan Distance Heuristic



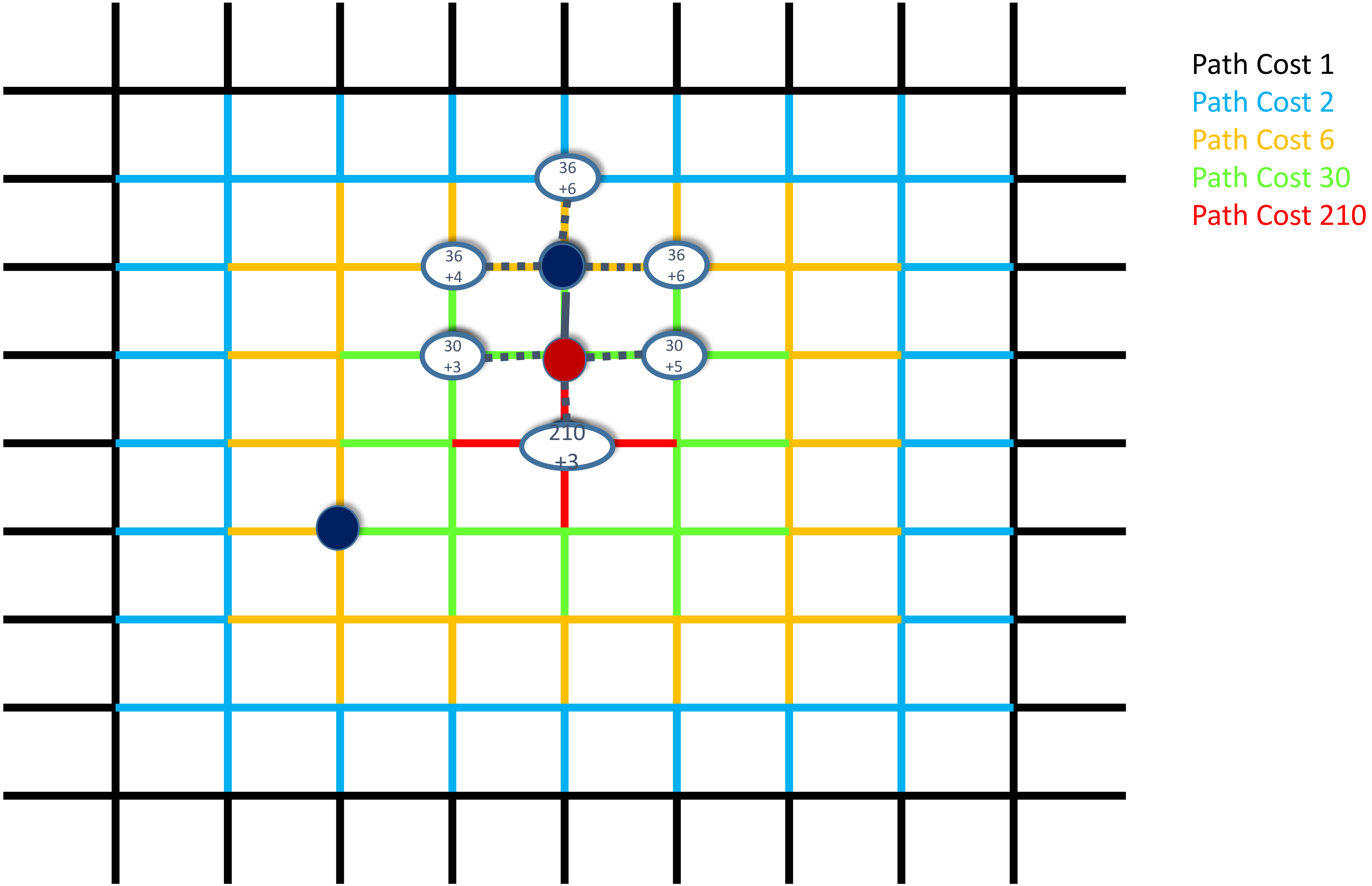
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



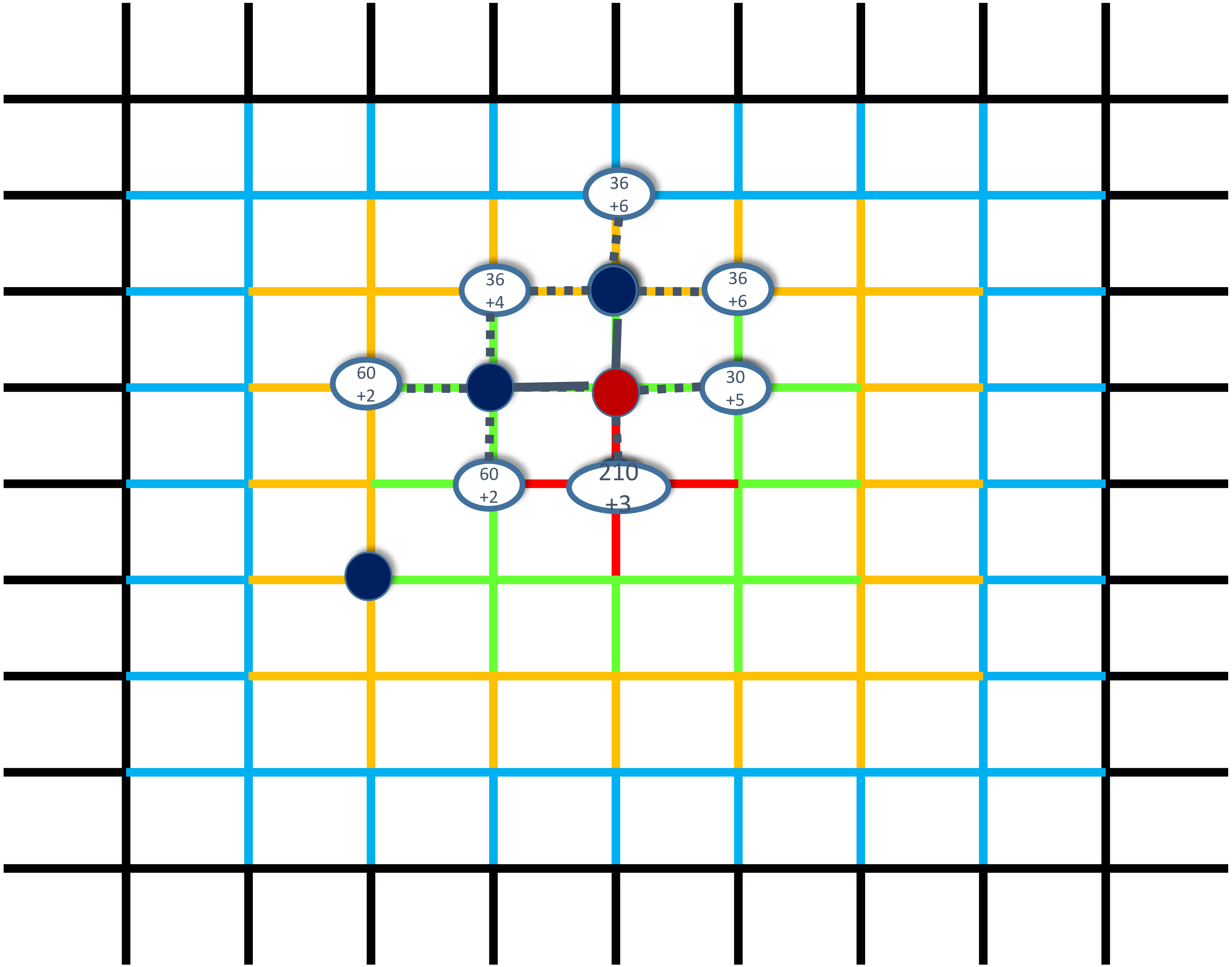
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



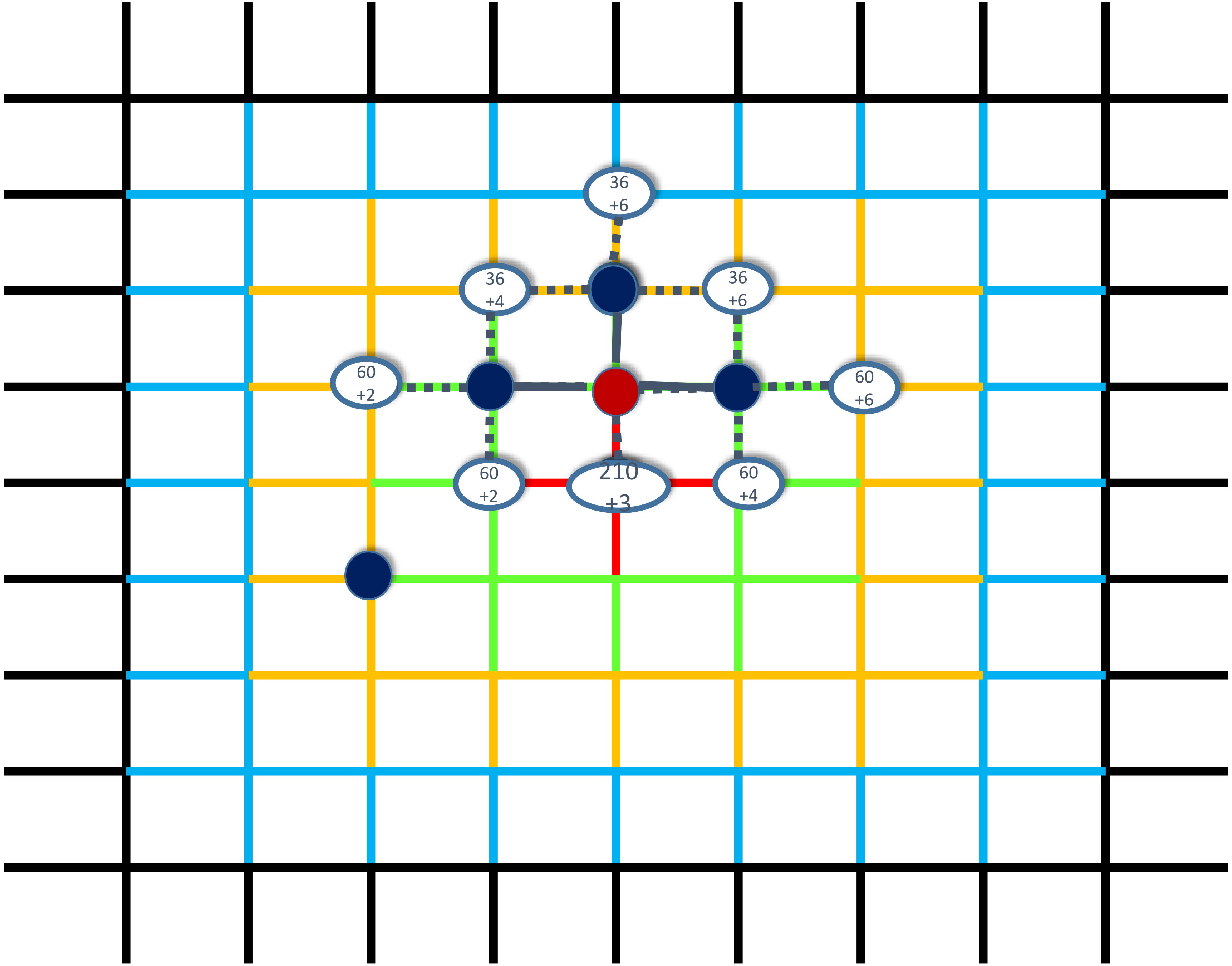
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



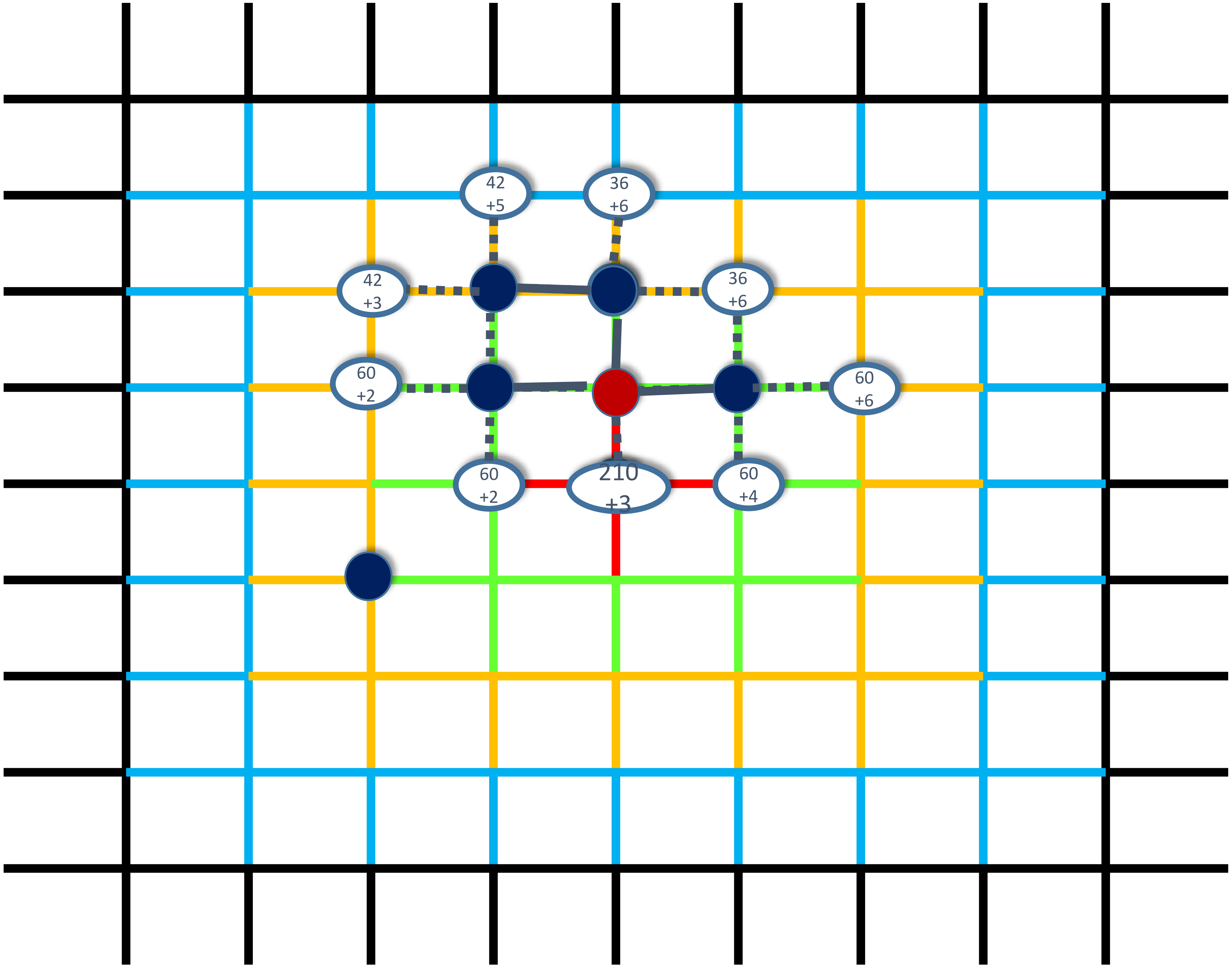
Path Cost 1

Path Cost 2

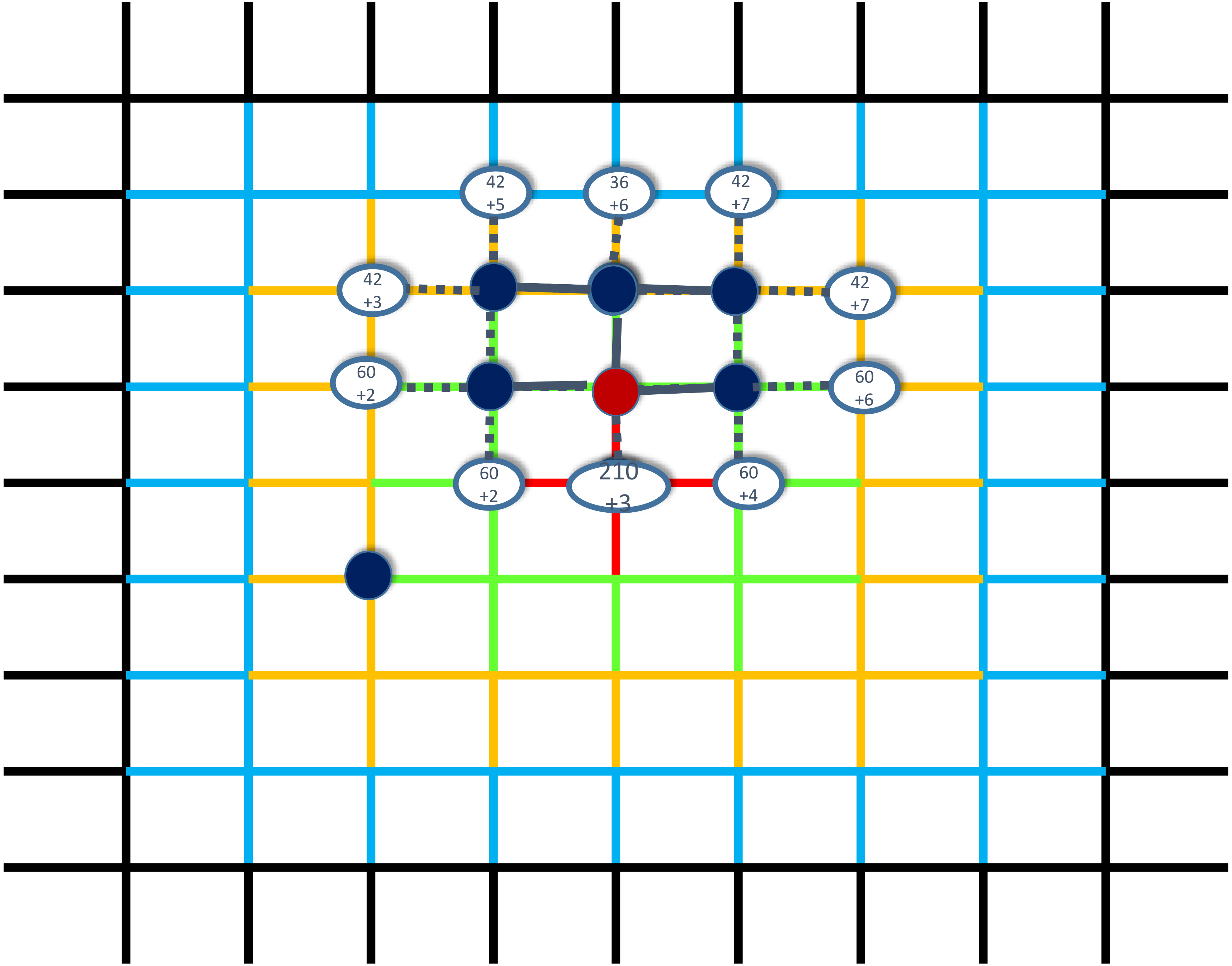
Path Cost 6

Path Cost 30

Path Cost 210



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210



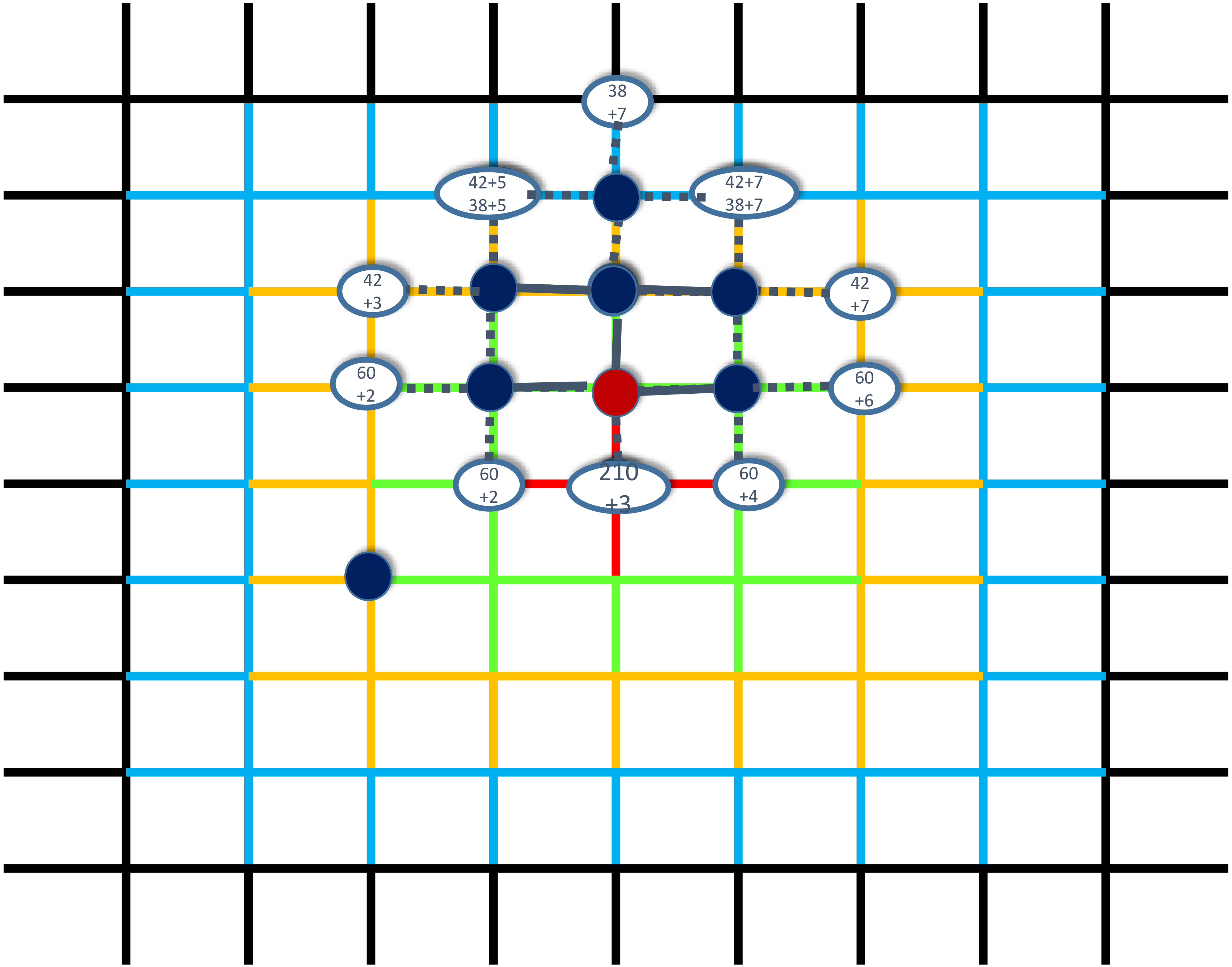
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



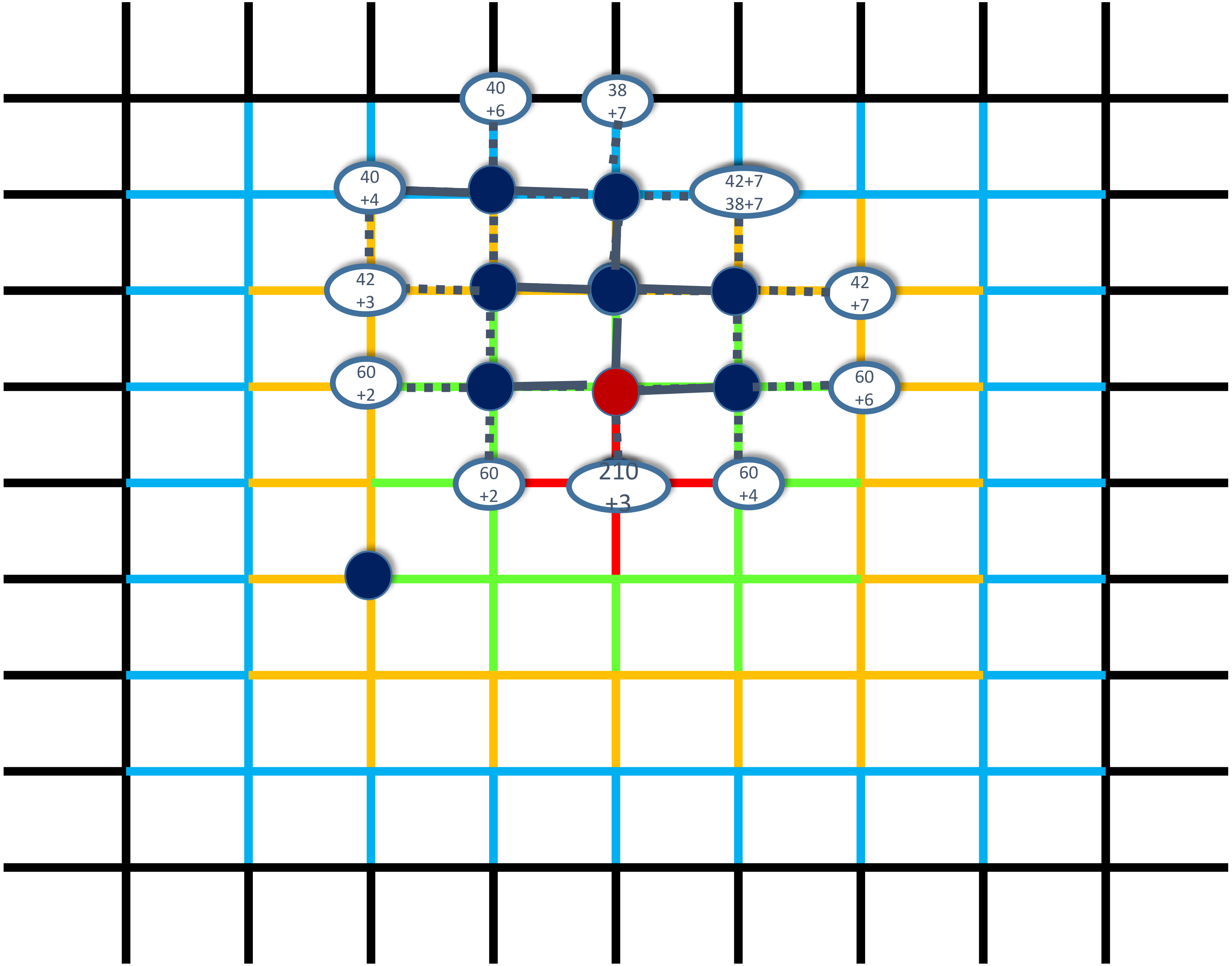
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



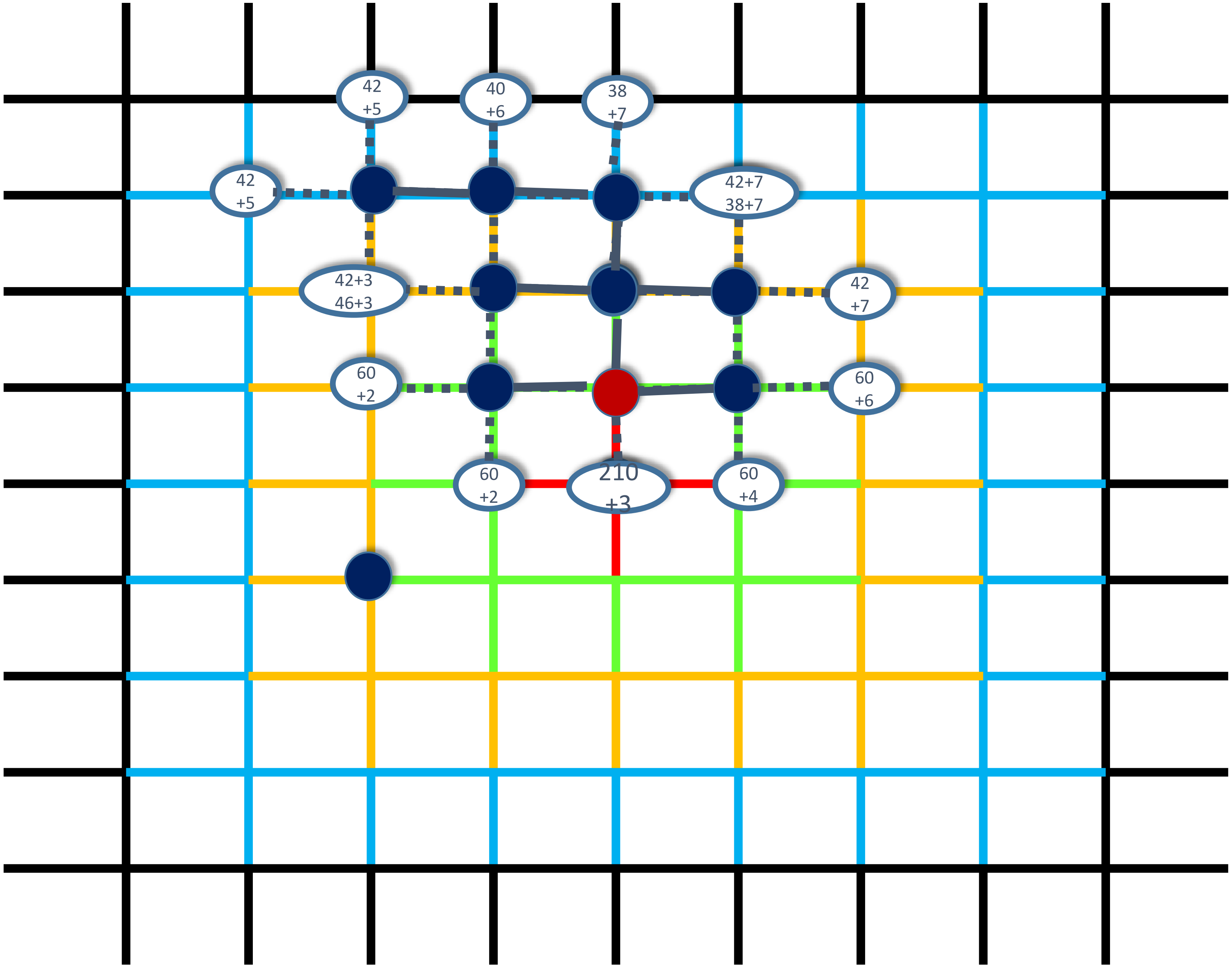
Path Cost 1

Path Cost 2

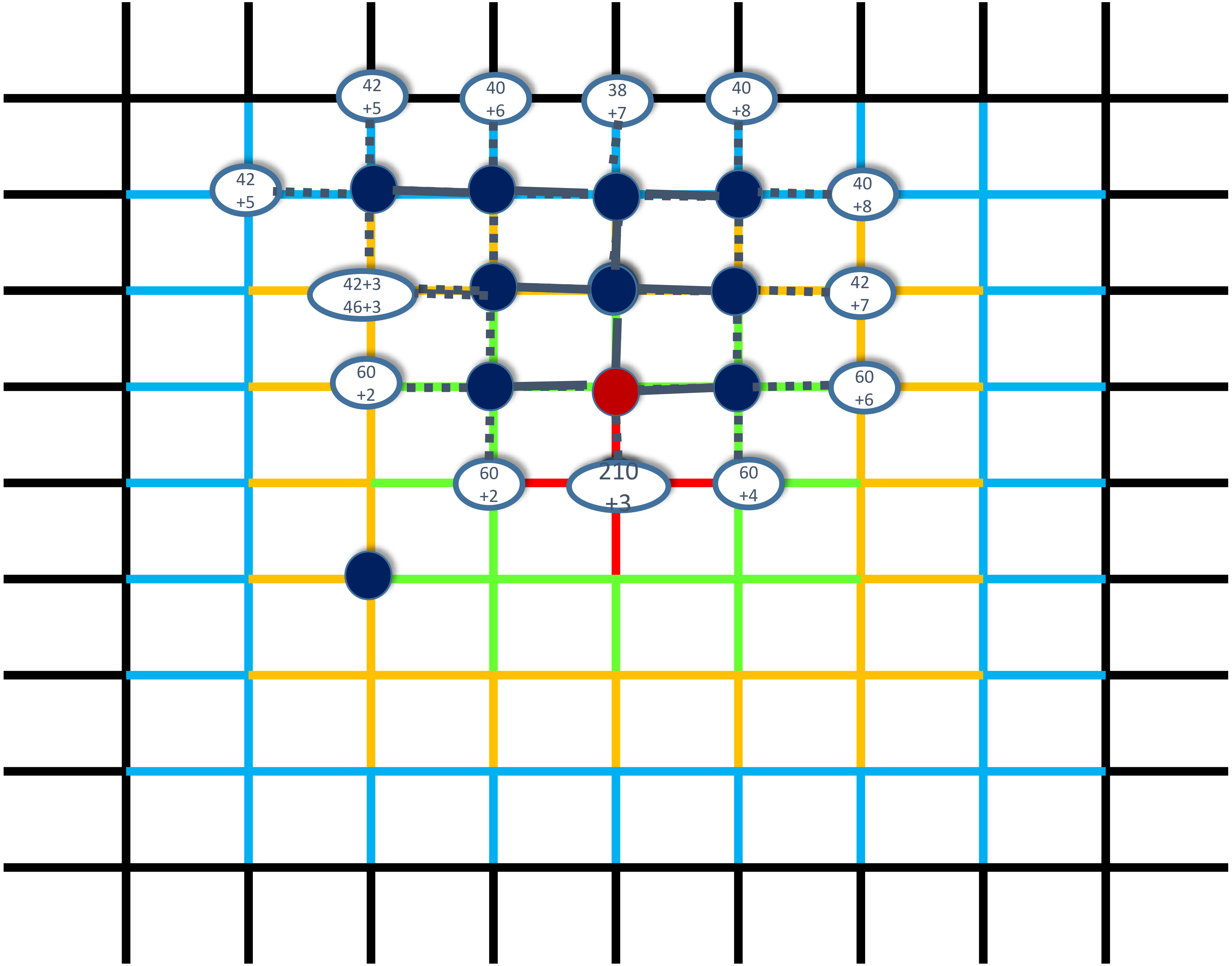
Path Cost 6

Path Cost 30

Path Cost 210



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210



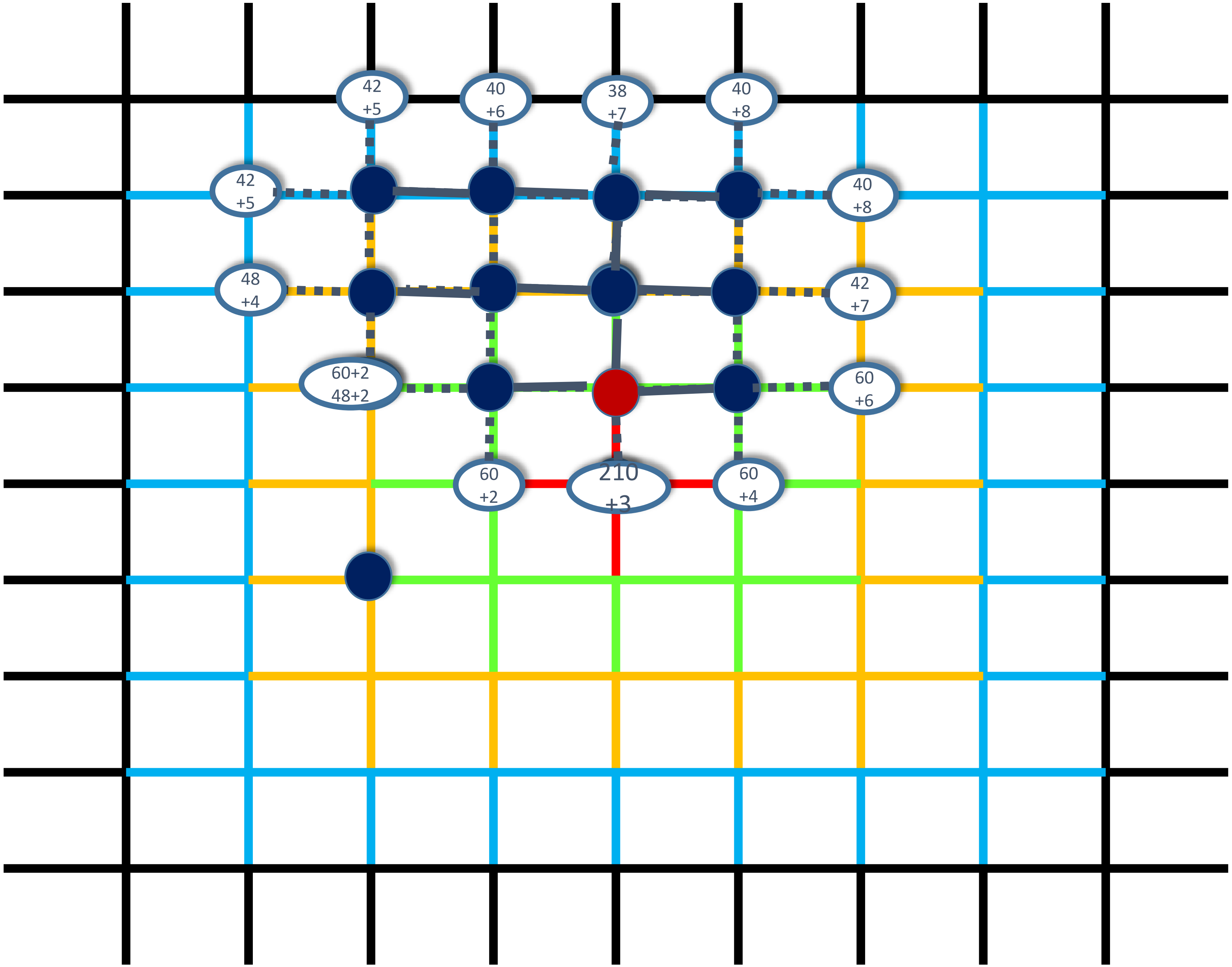
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



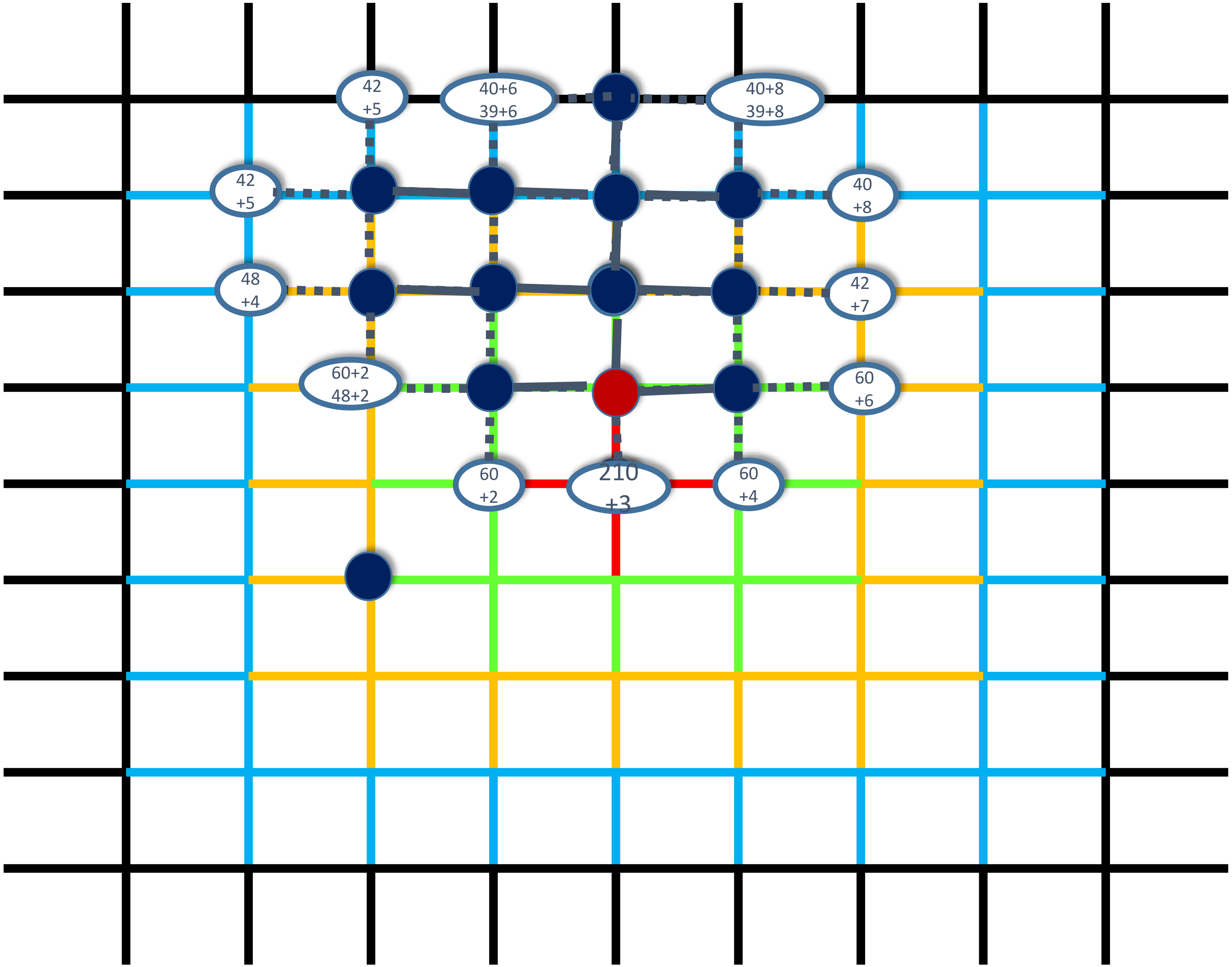
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



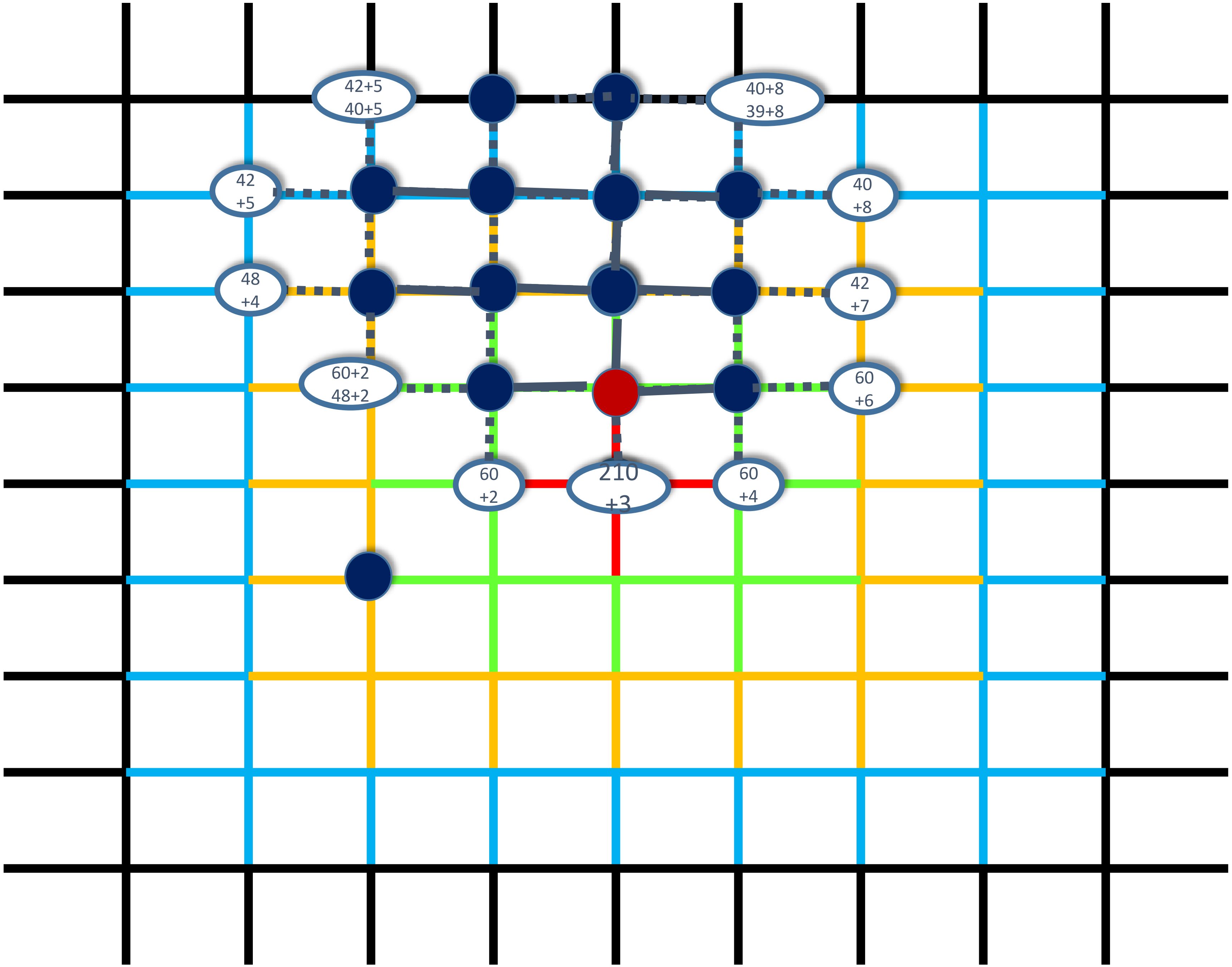
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



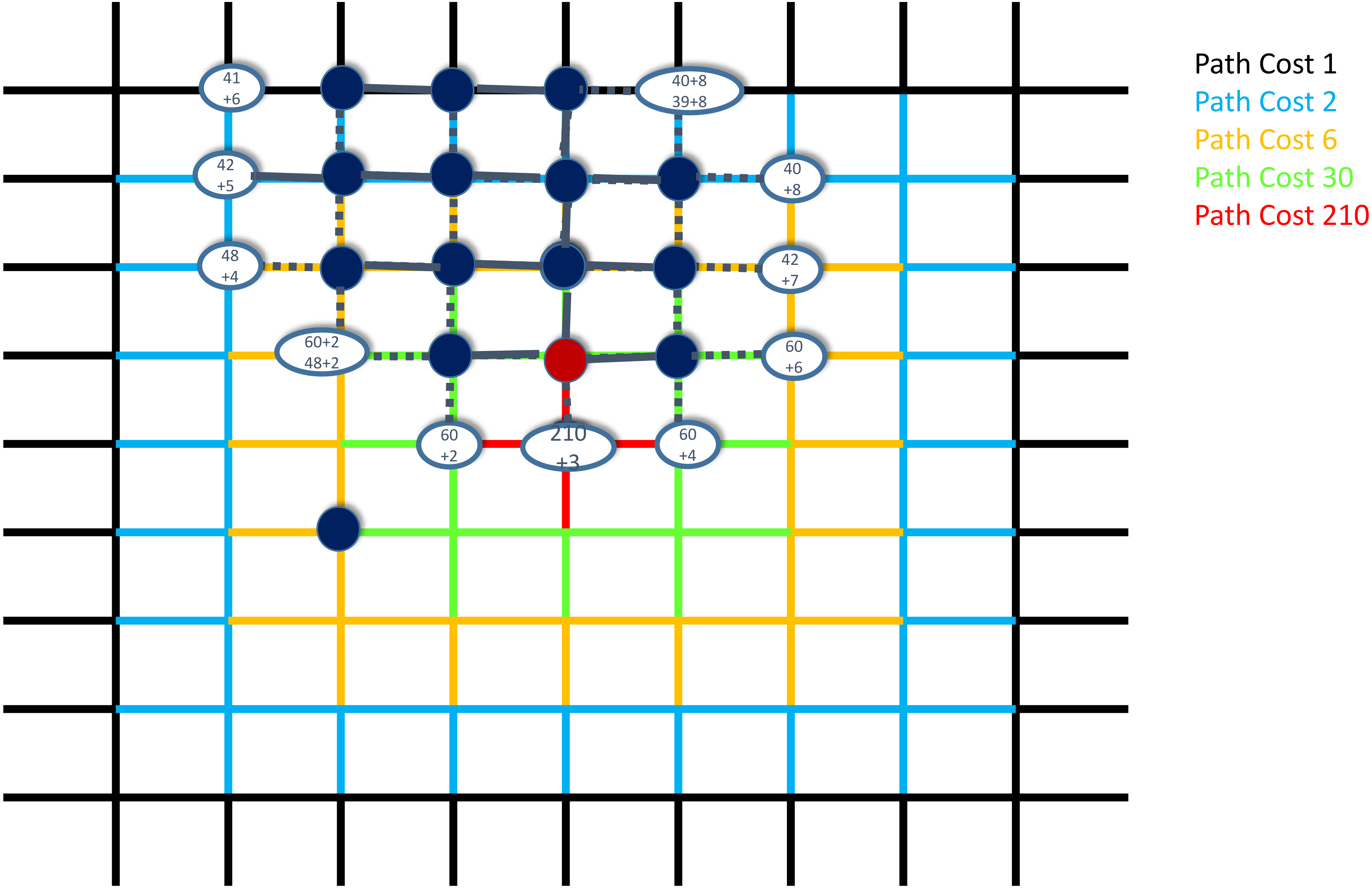
Path Cost 1

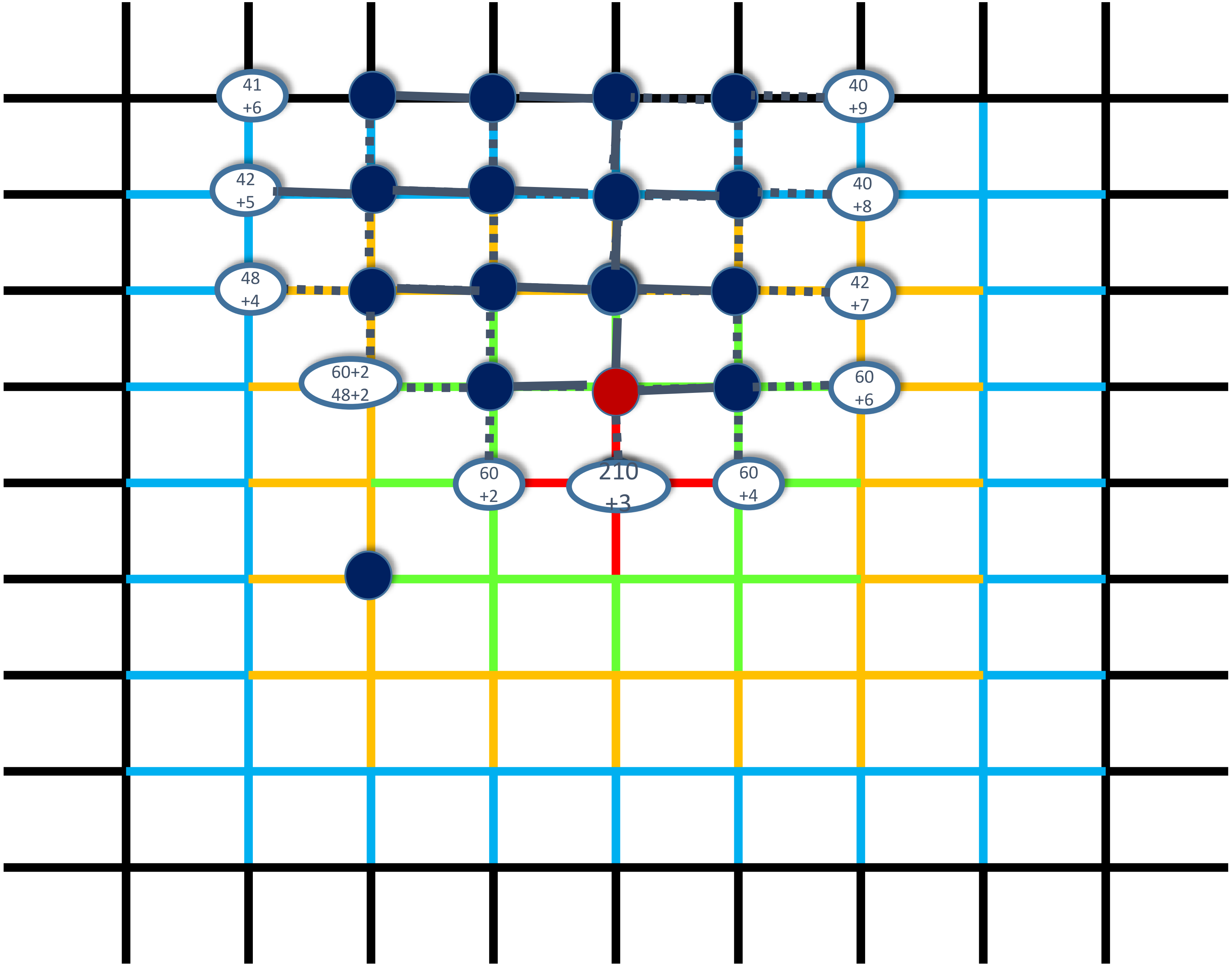
Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210


41
+6
40+8
39+8
42
+5
40
+8
48
+4
42
+7
60+2
48+2
60
+6
60
+2
210
+3
60
+4
Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210



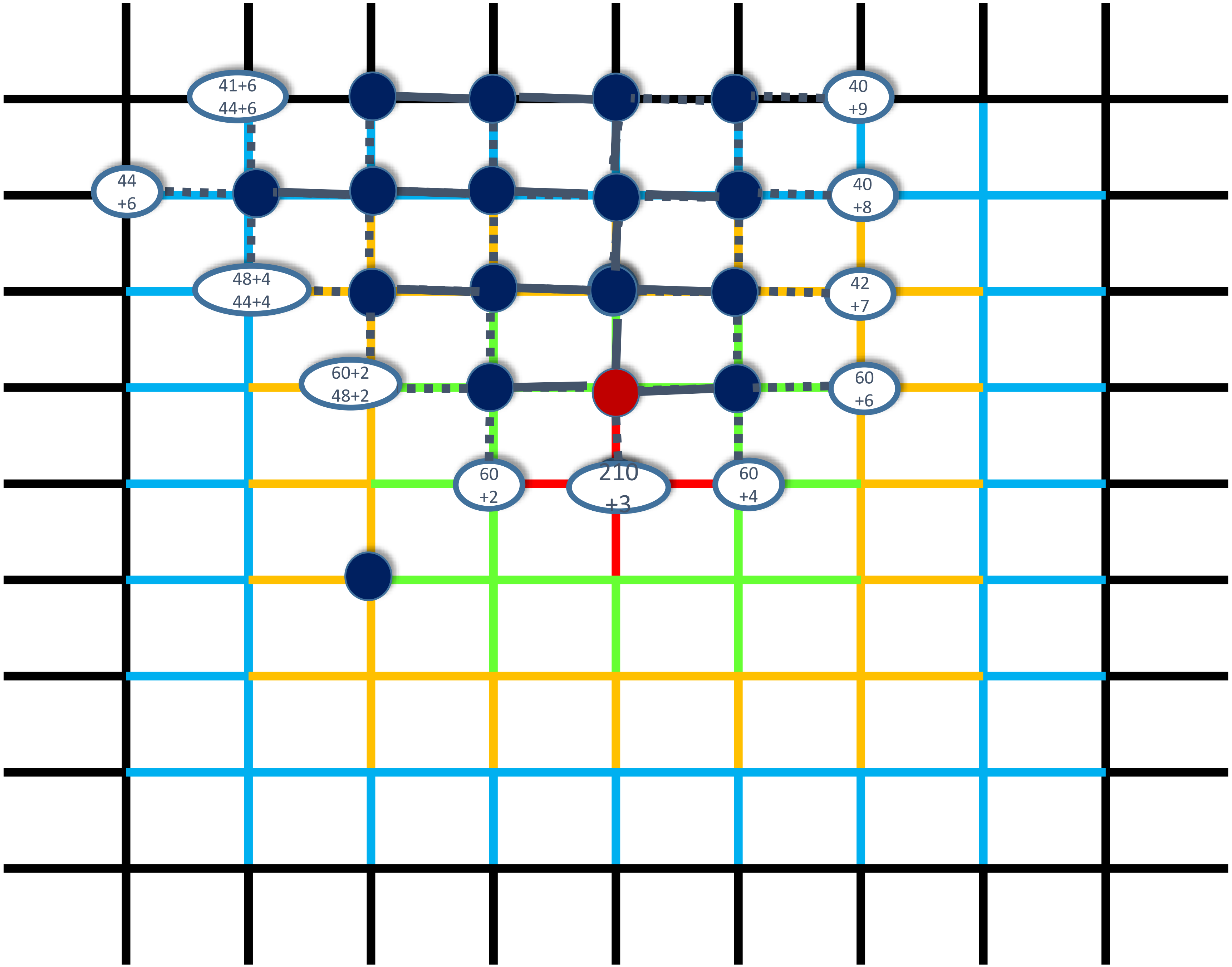
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



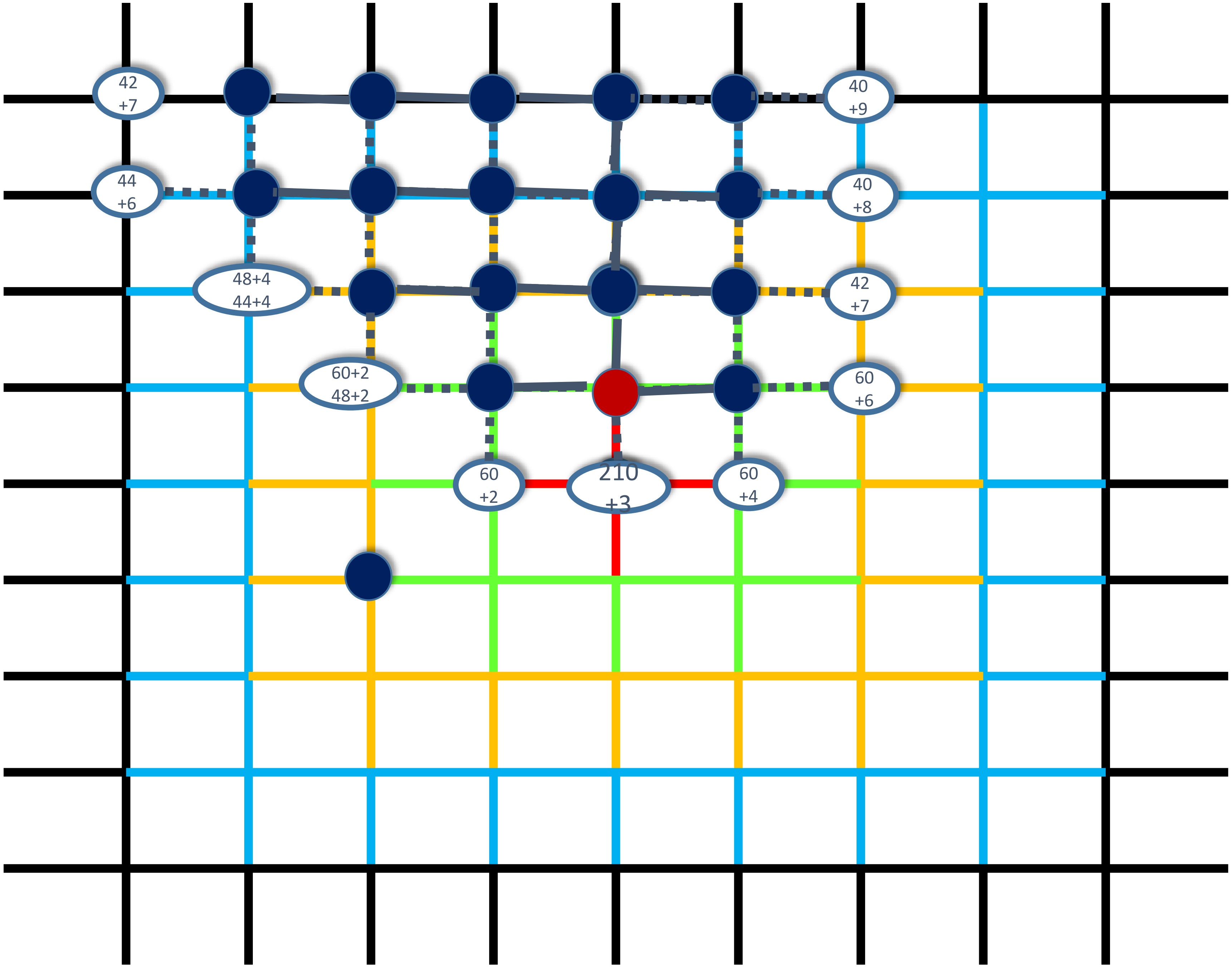
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

Path Cost 210



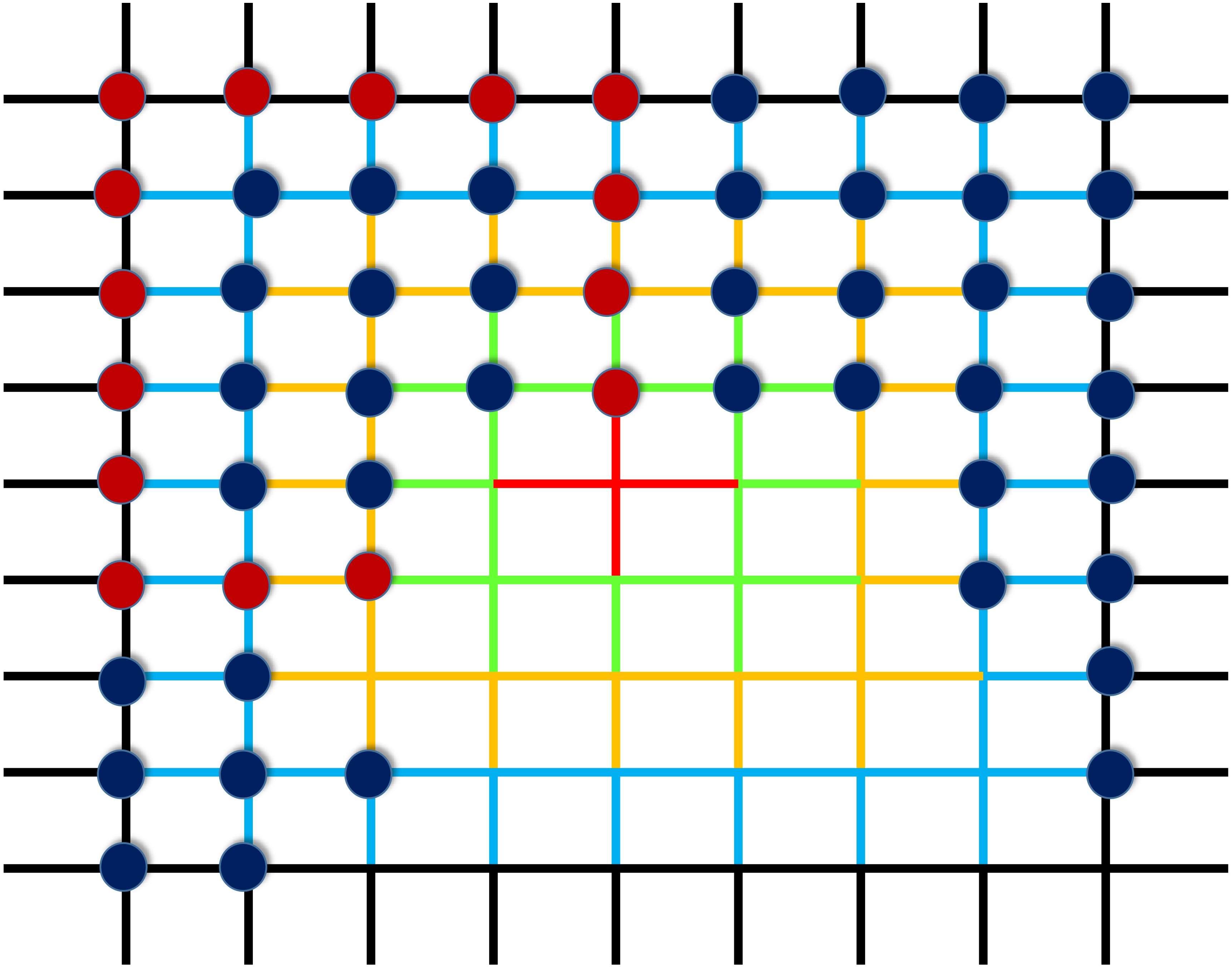
Path Cost 1

Path Cost 2

Path Cost 6

Path Cost 30

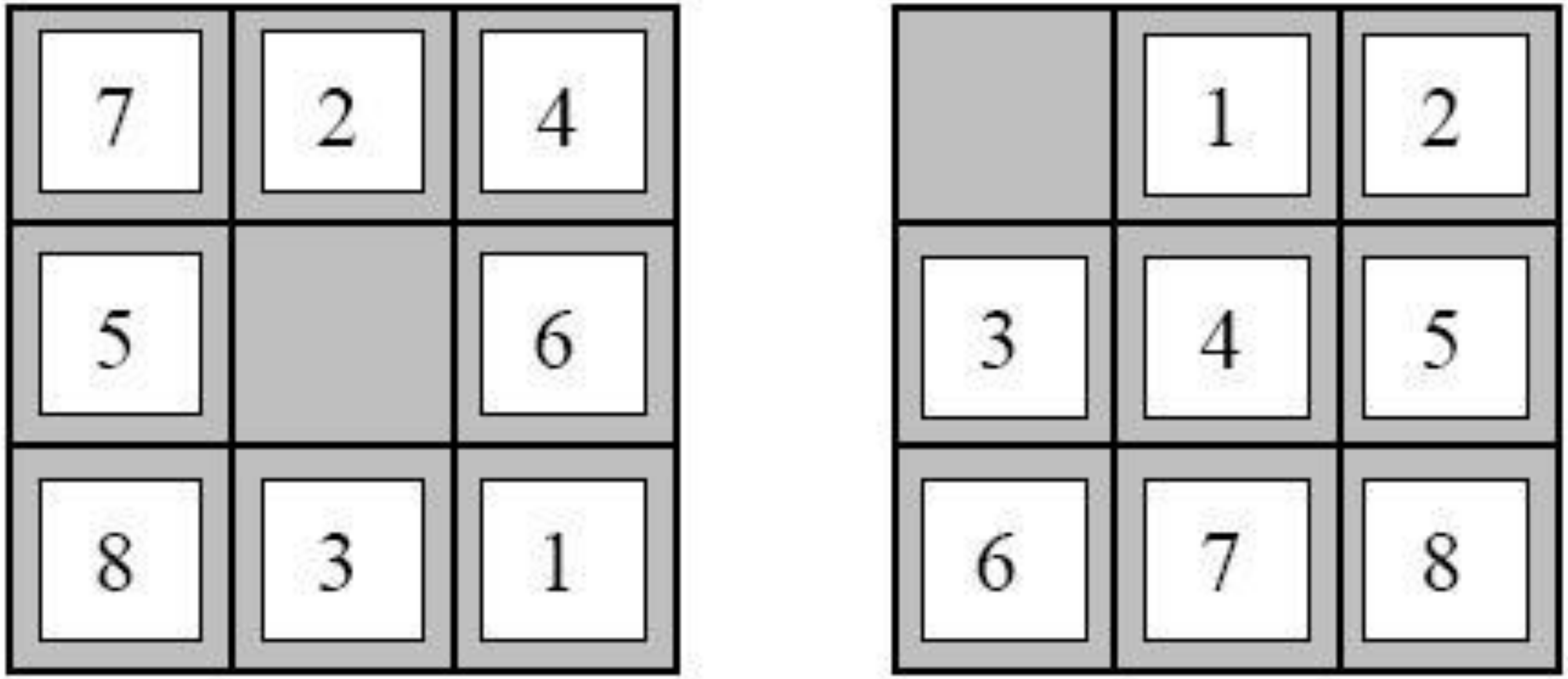
Path Cost 210



Path Cost 1
Path Cost 2
Path Cost 6
Path Cost 30
Path Cost 210

Number Explored: 55
Shortest Cost: 55

Choosing Good Heuristics



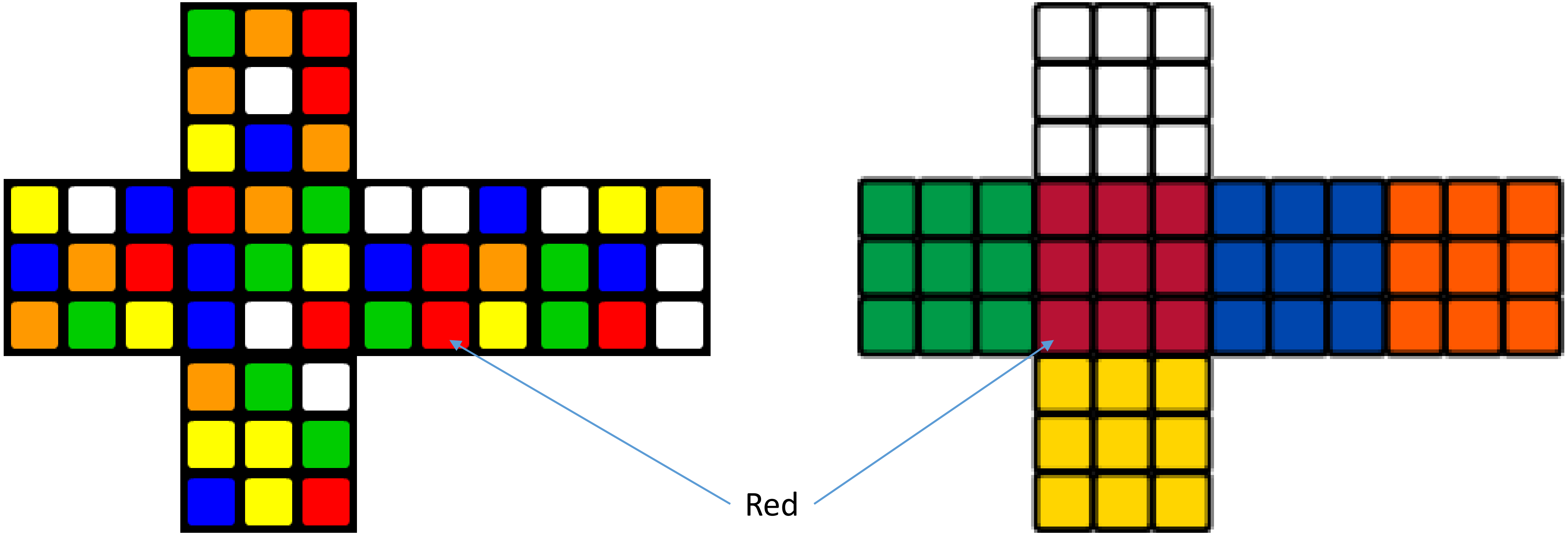
7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Image from AIMA, 3rd Edition



https://www.cs.princeton.edu/courses/archive/fall06/cos402/papers/korfrubik.pdf