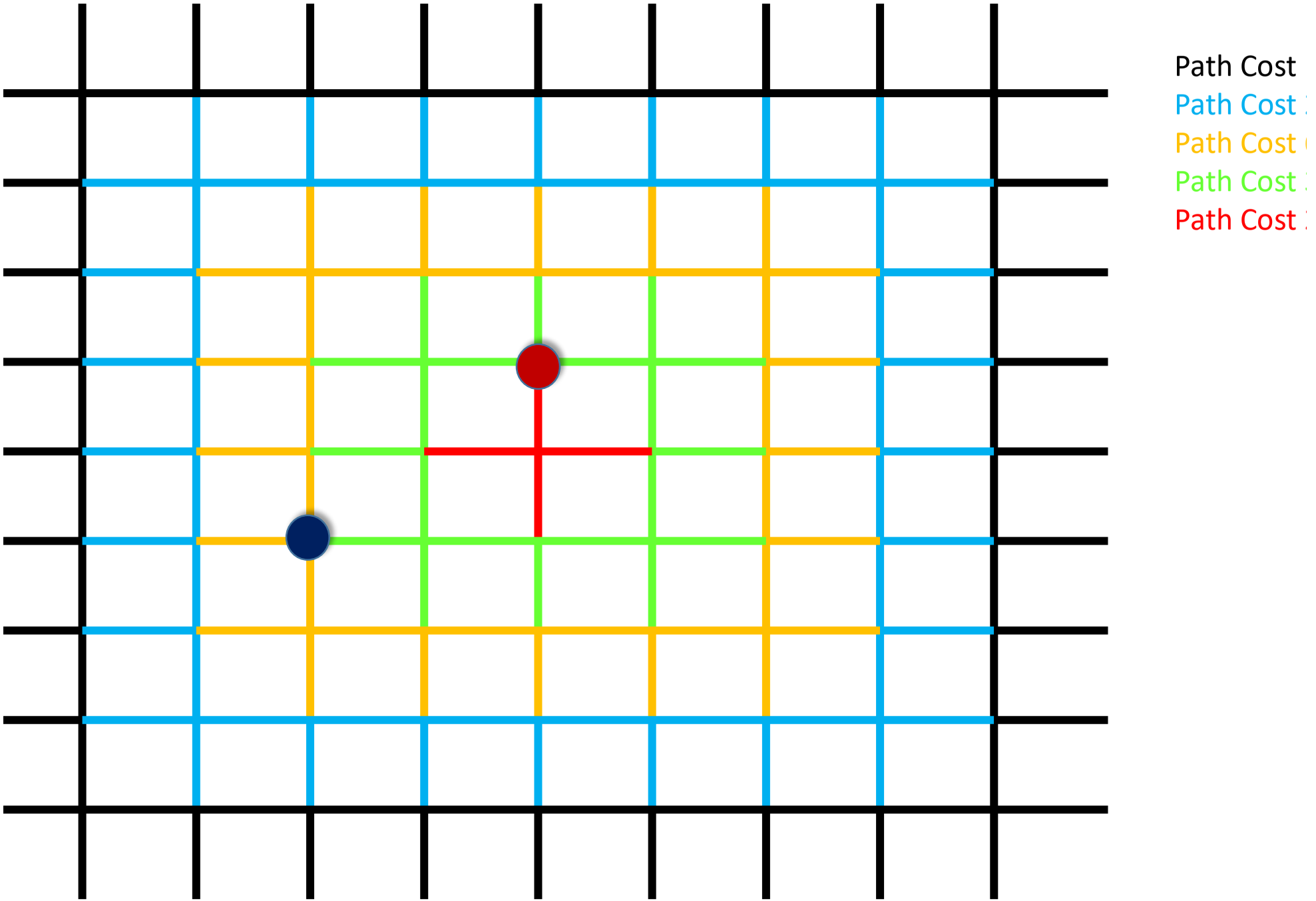
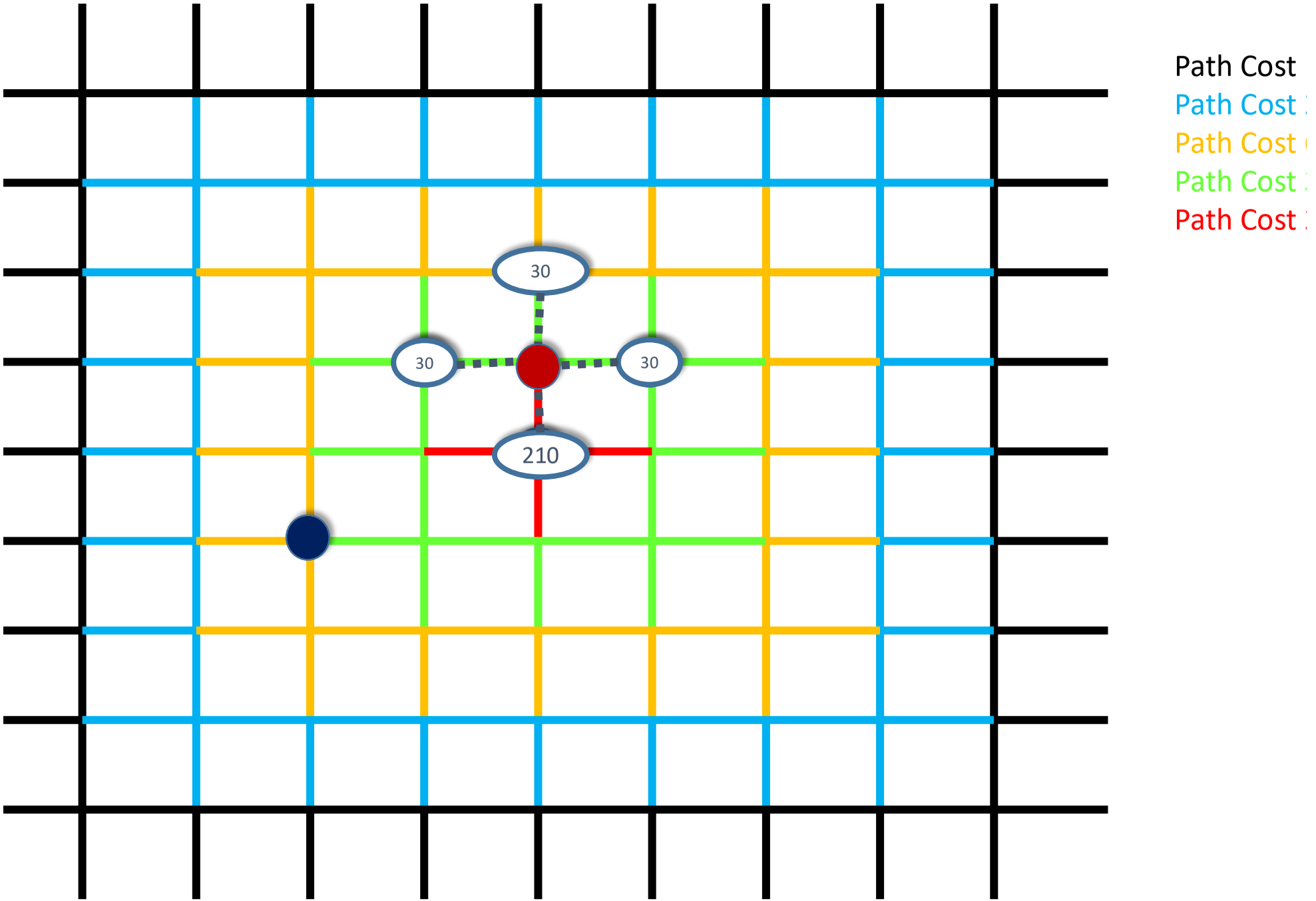
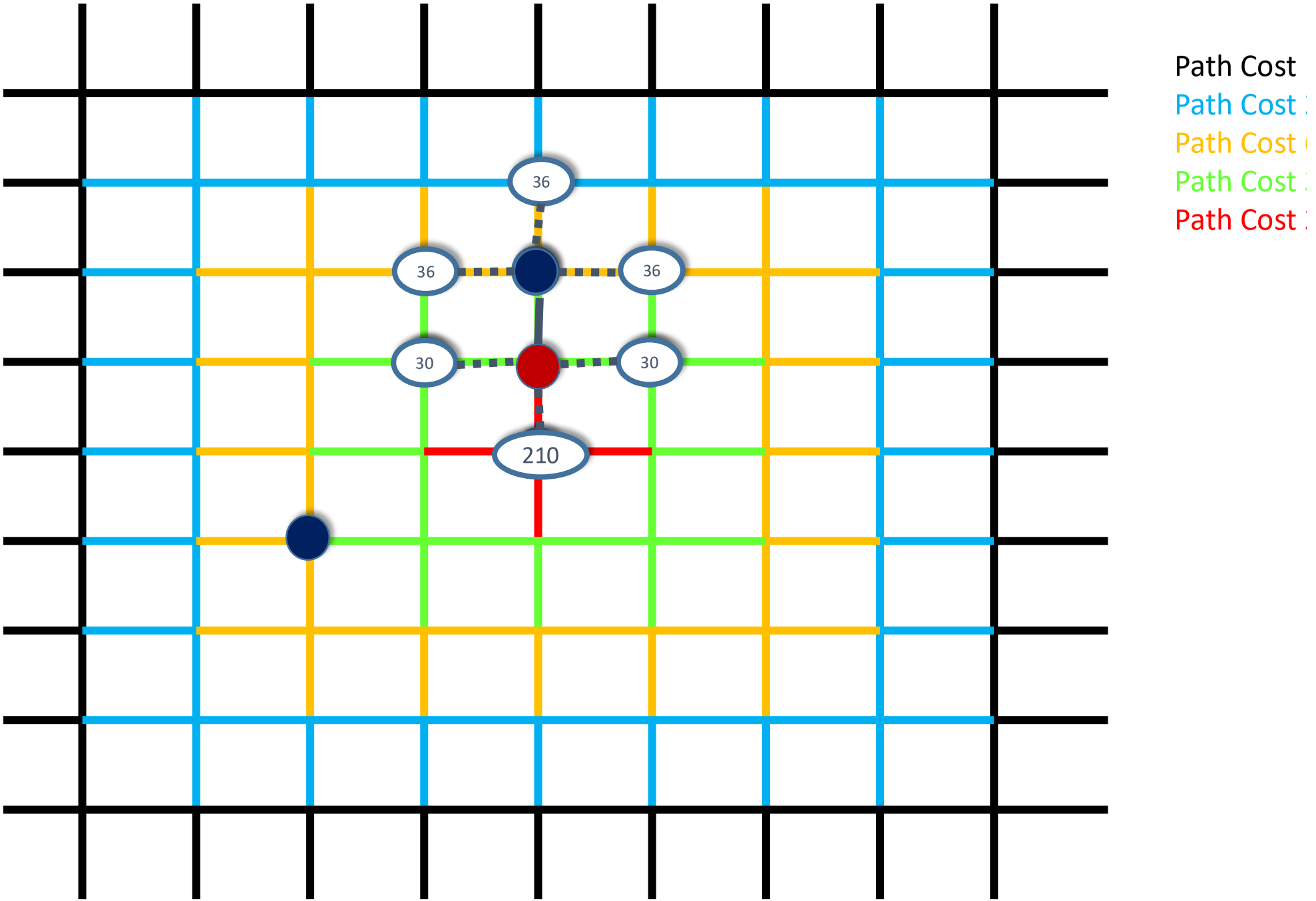
Search Algorithms

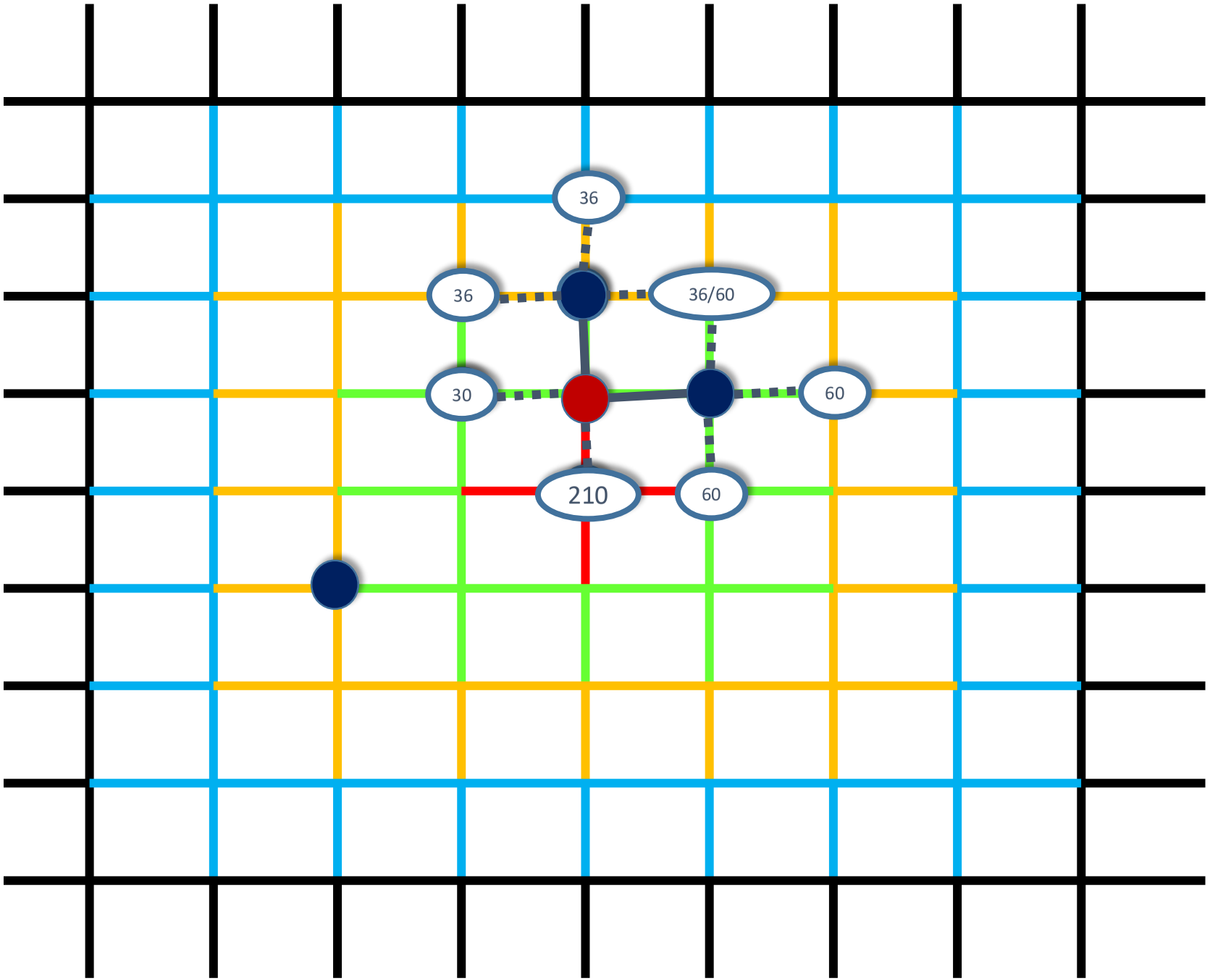
niform Cost Search

```
e = start
ıtier = heap({node})
.ored = {}
.e 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
Path Cost
Path Cost
Path Cost
Path Cost

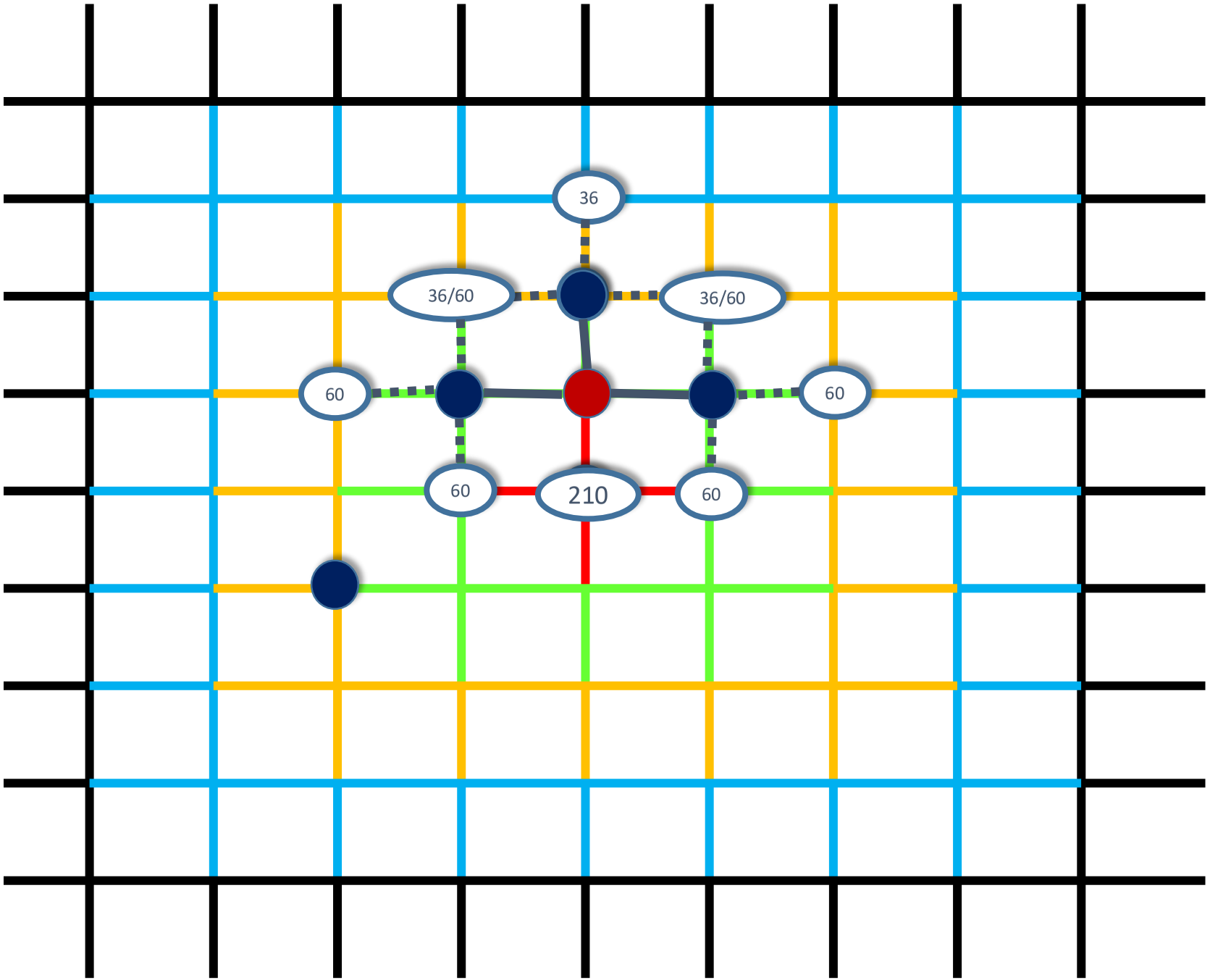

Path Cost
Path Cost
Path Cost
Path Cost
Path Cost
30
30
30
210


Path Cost
Path Cost
Path Cost
Path Cost
Path Cost
36
36
36
30
30
210



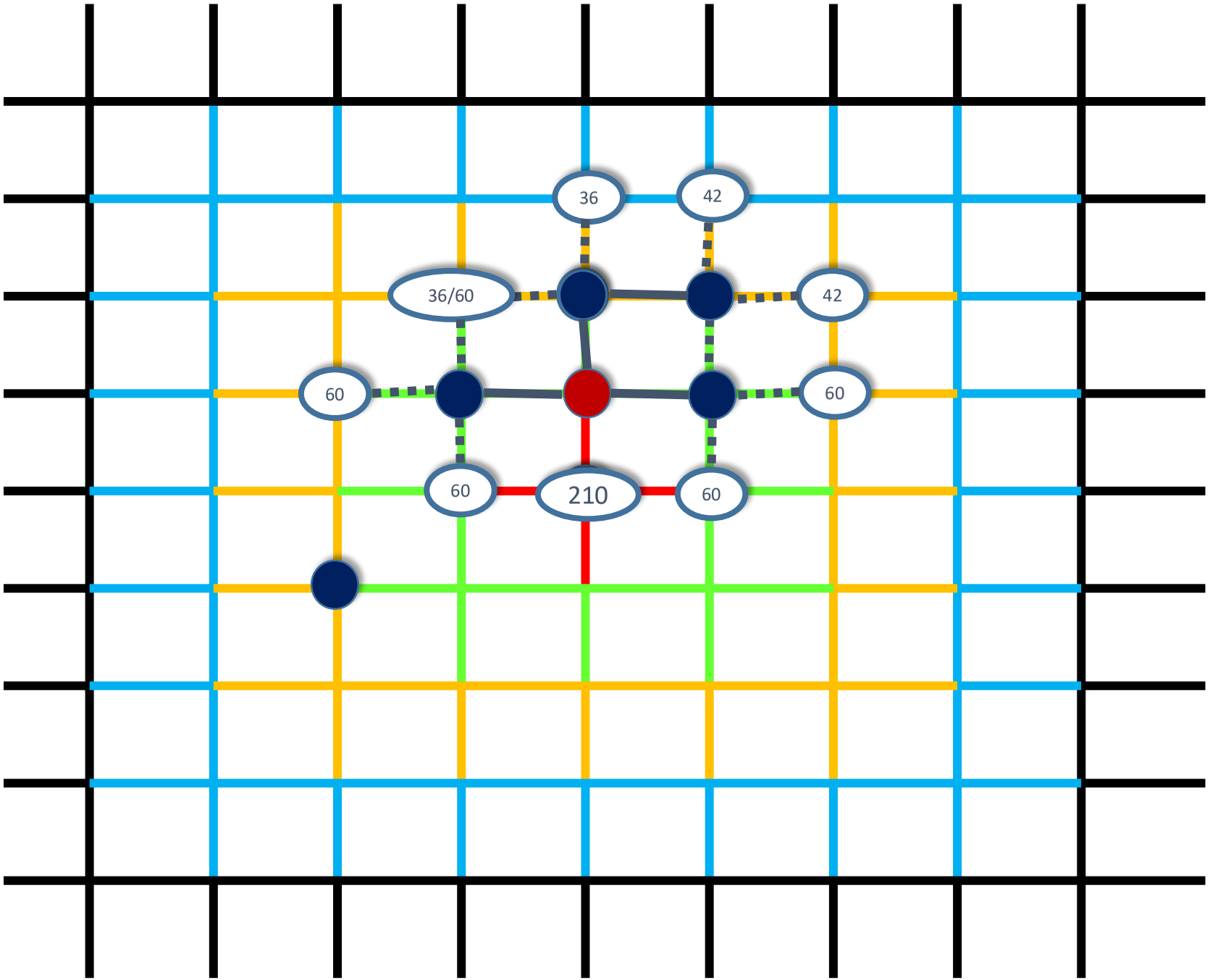
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise



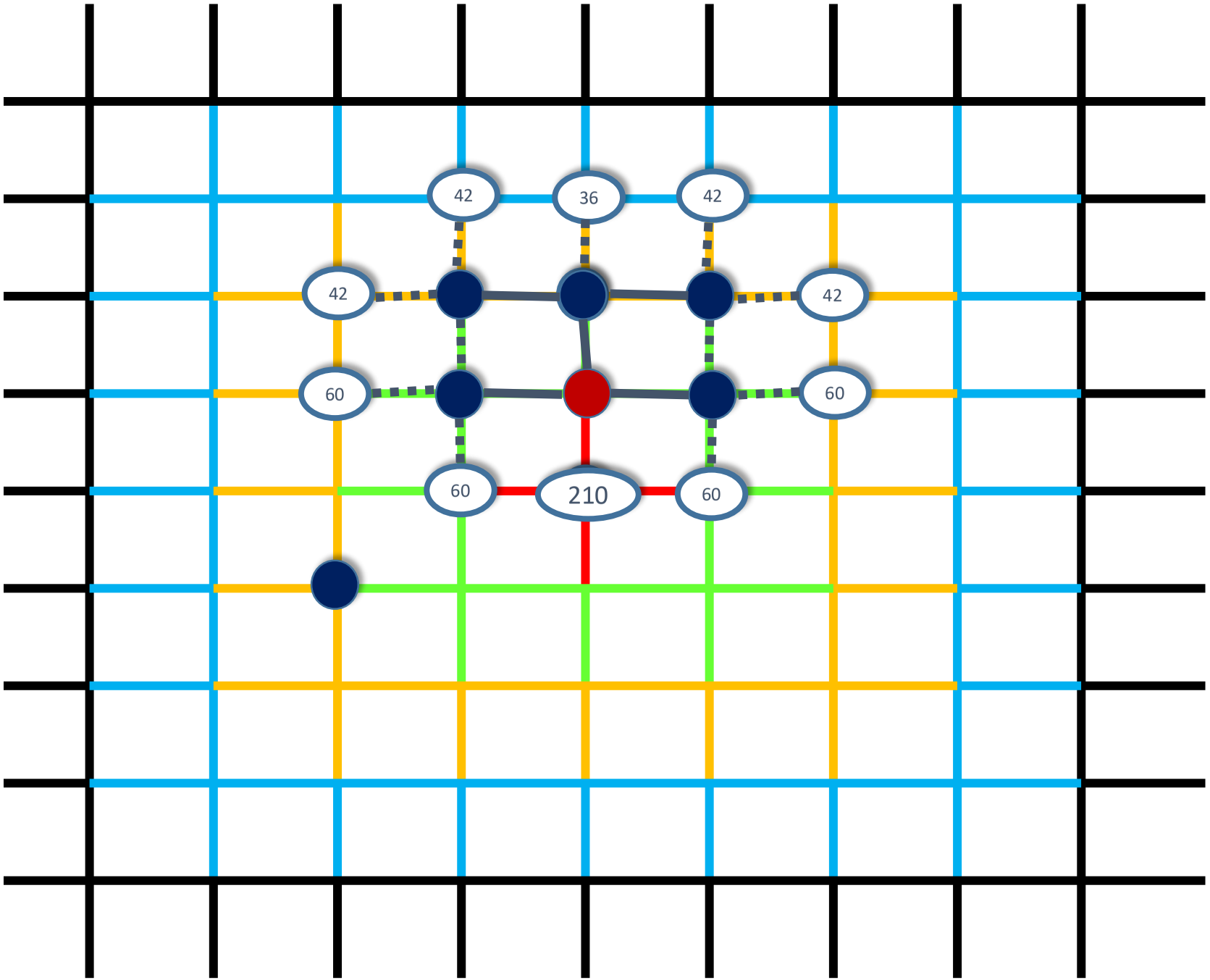
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise



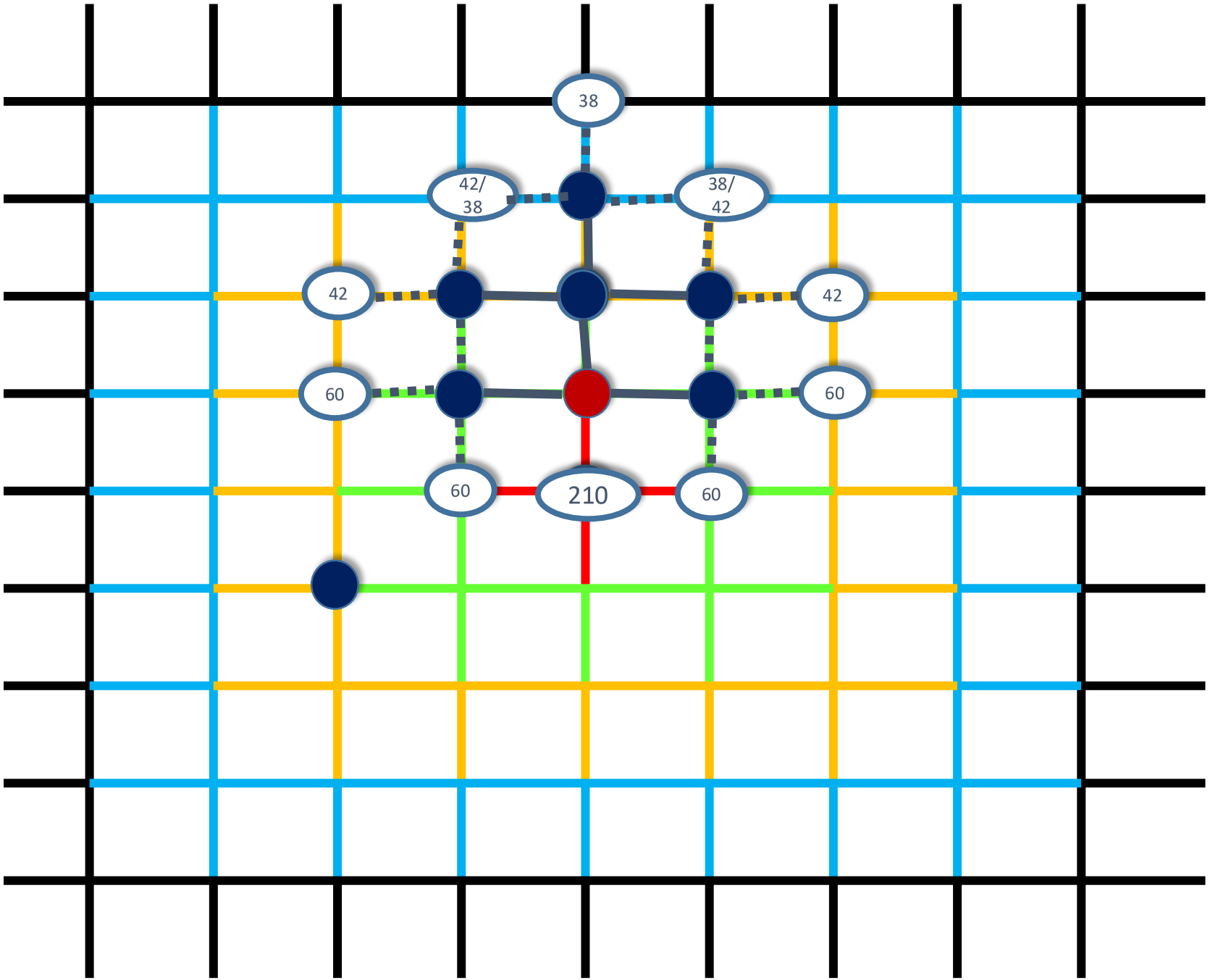
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise



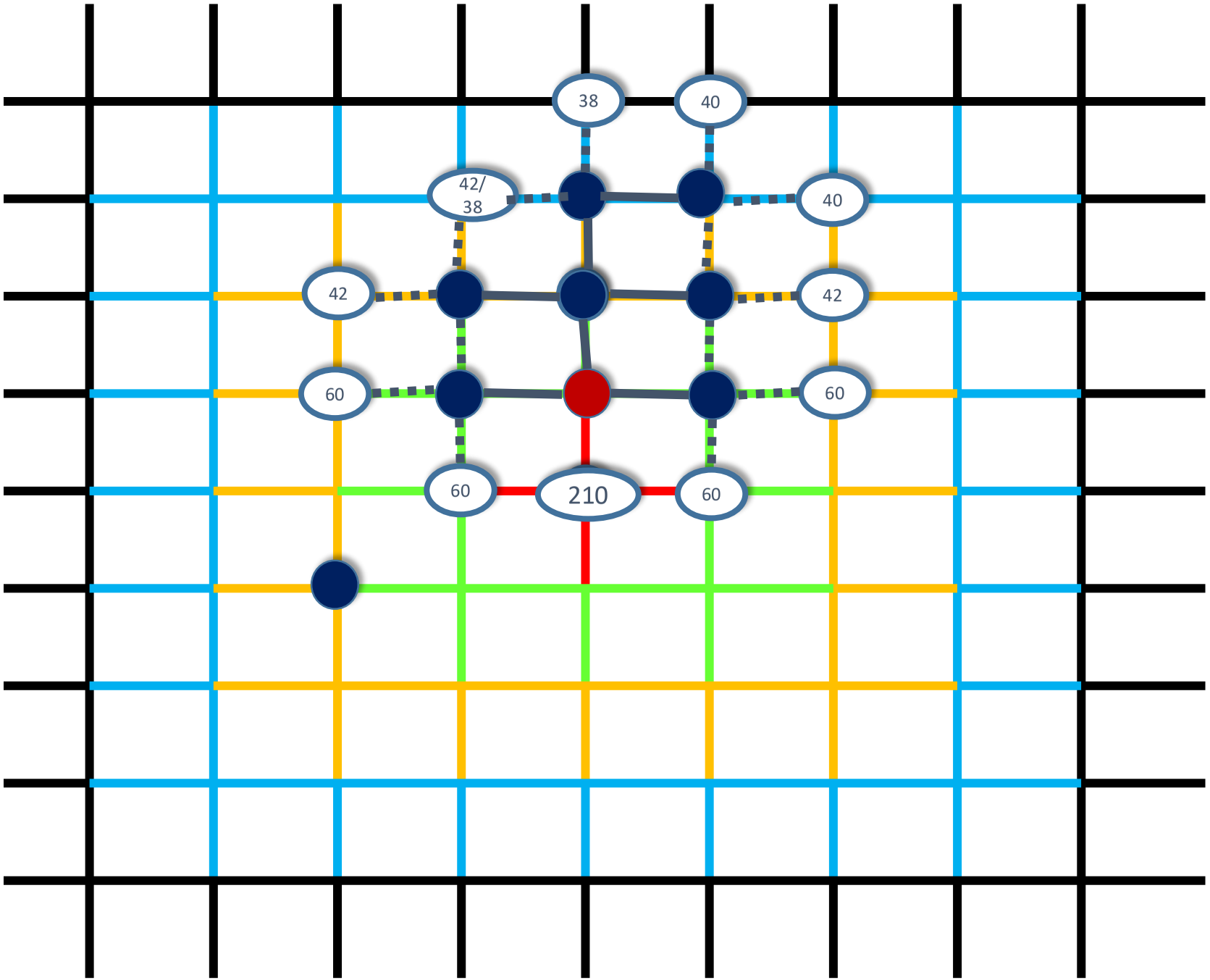
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise



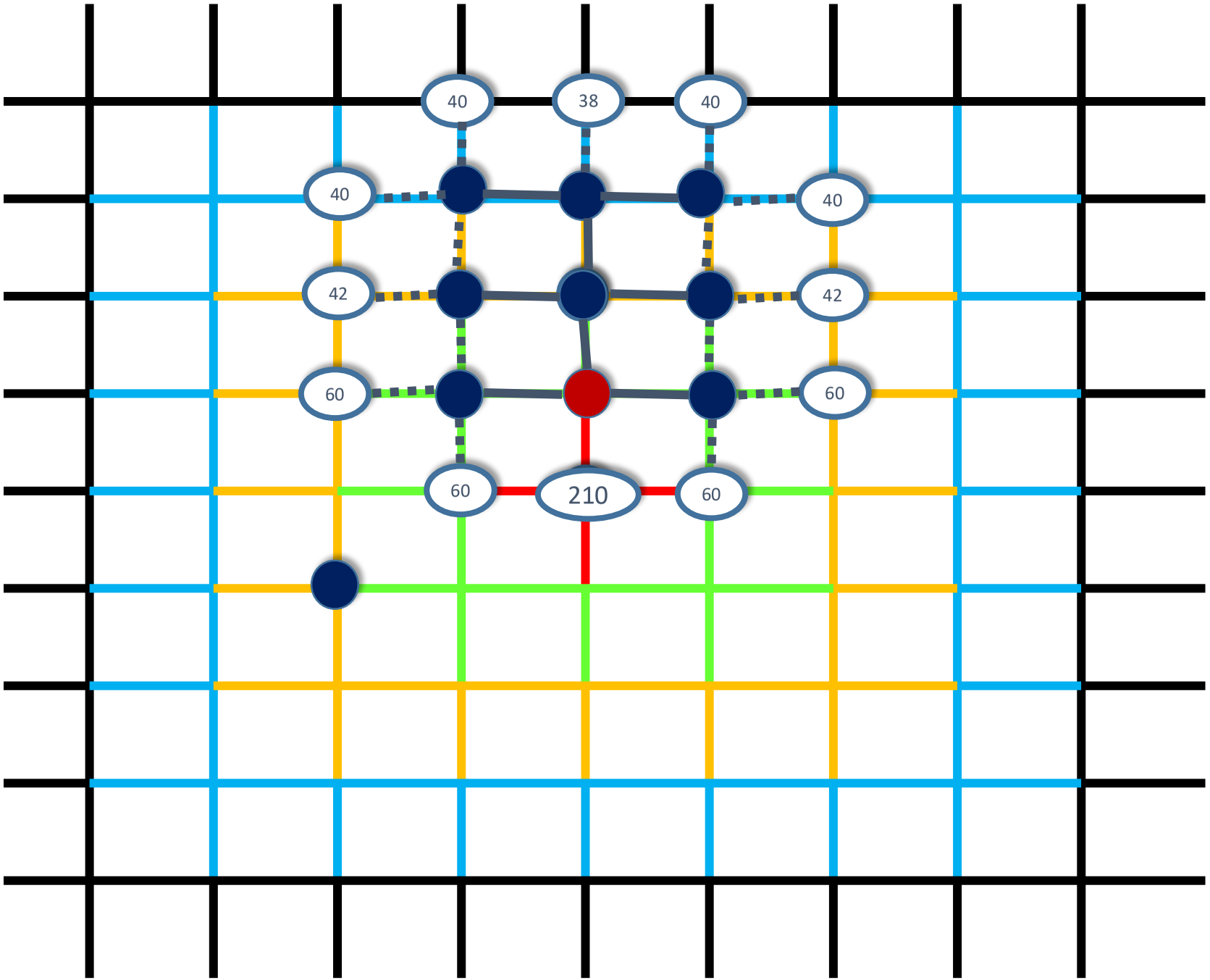
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise



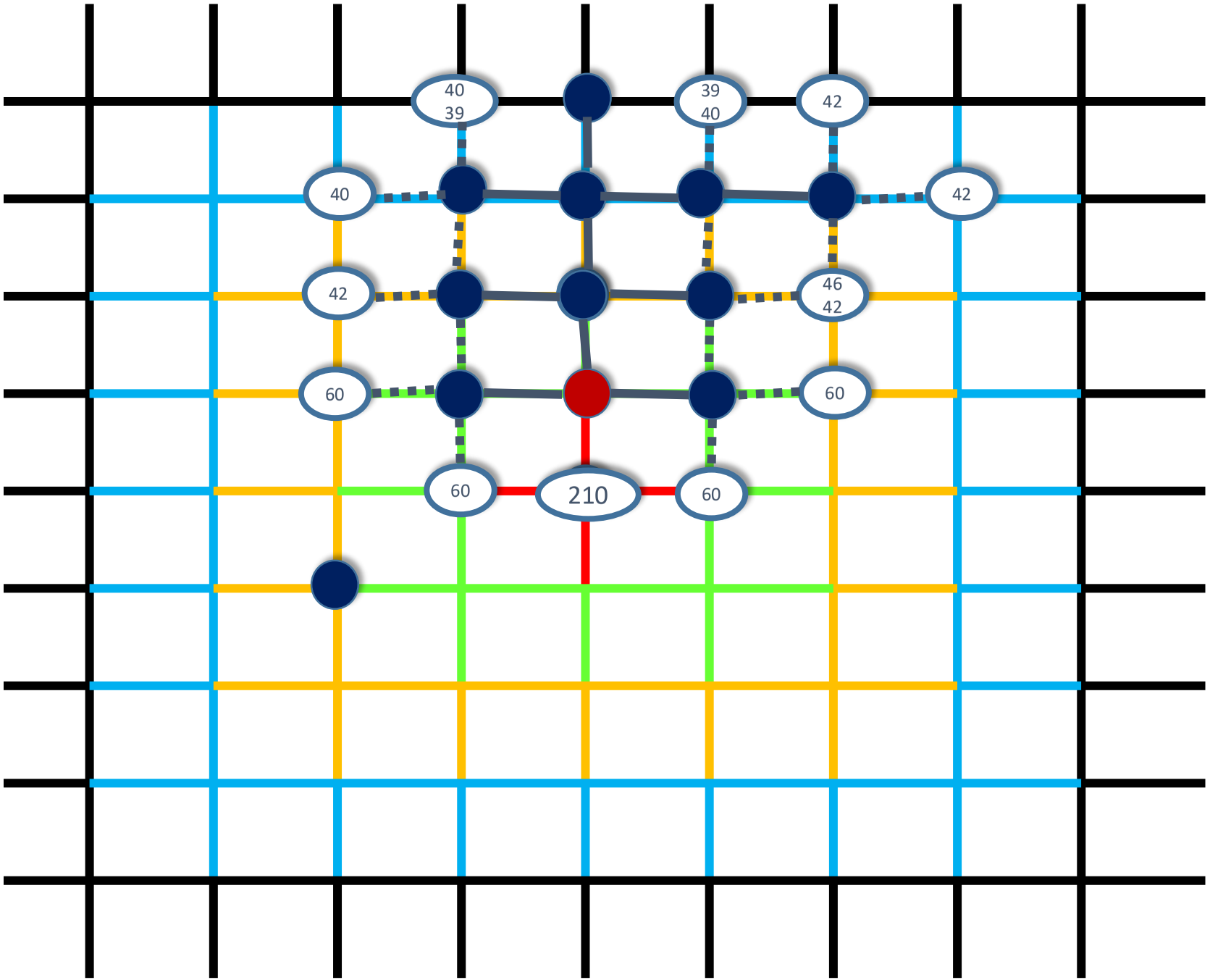
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise



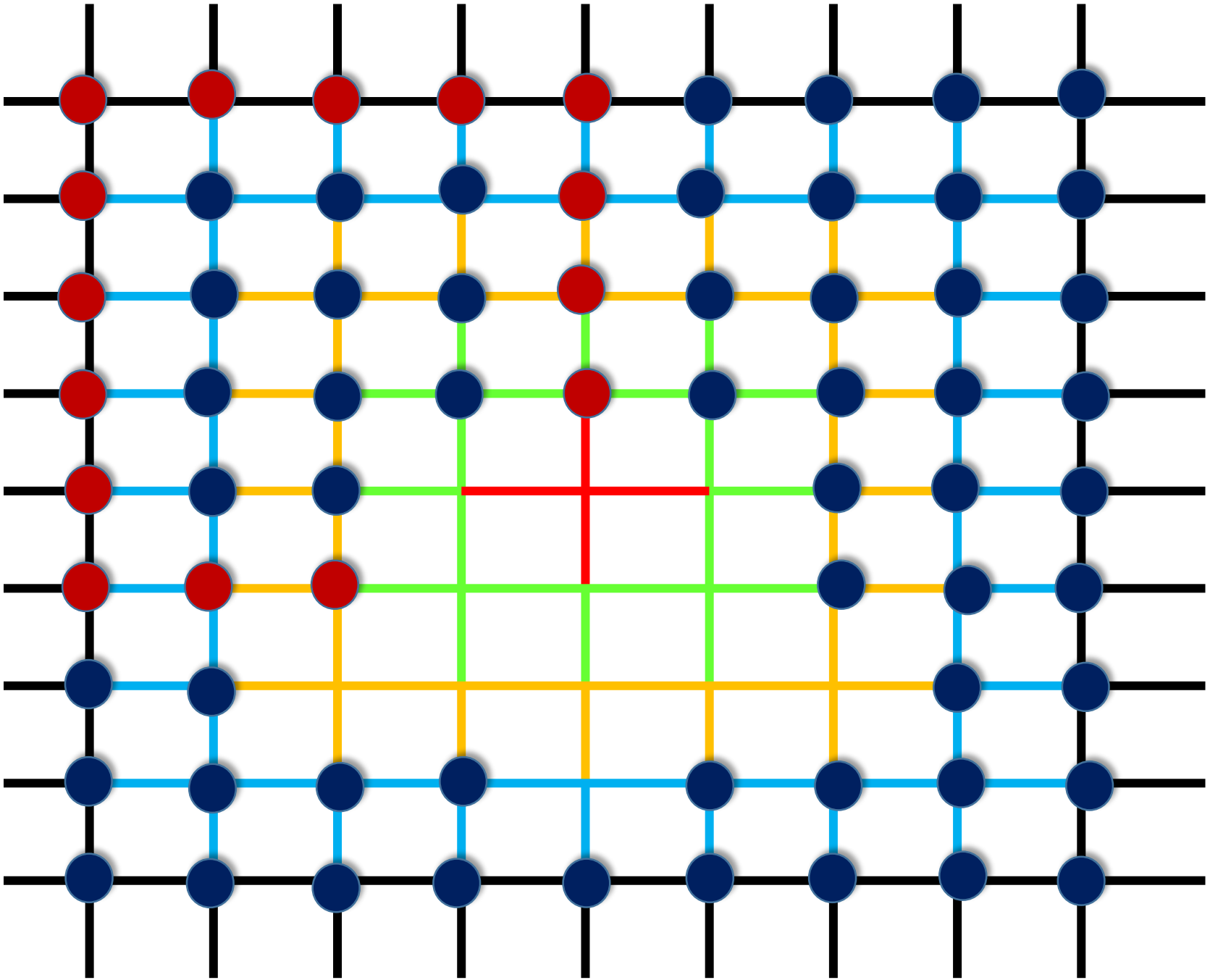
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise



Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

Conventio
Clockwise


Path Cost
Path Cost
Path Cost
Path Cost
Path Cost
Conventio
Clockwise
Number E
Shortest C

directional Uniform Cost Search

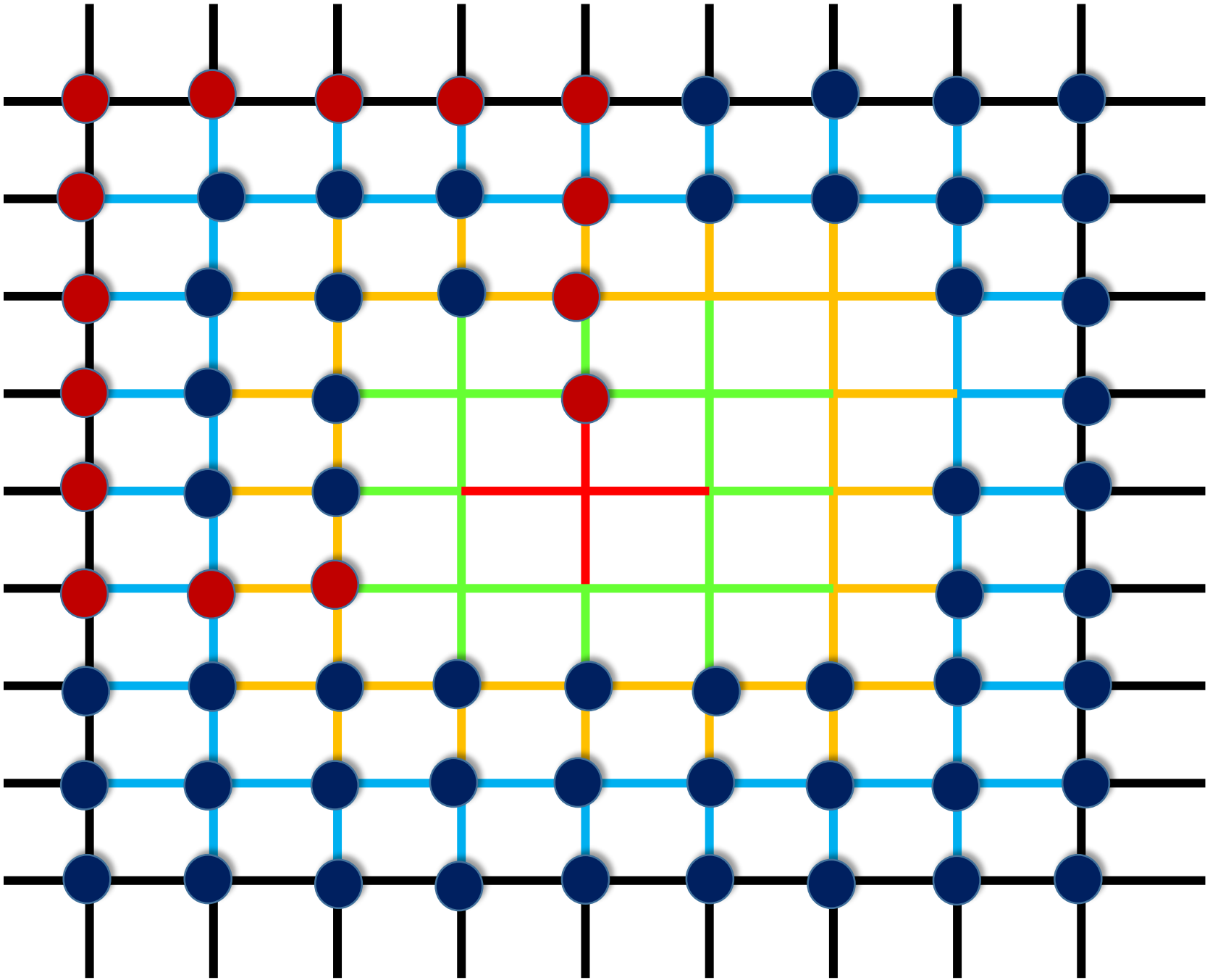
ny manner of expanding frontiers is OK

- Alternating both frontiers – good for parallel computing
- Taking the min – good in weighted graphs where hubs have high cost

opping 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
Path Cost
Path Cost
Path Cost
Path Cost

Number E
Shortest C

Notes:
- Expand
  of front
- Clockw
  breaks

: Search

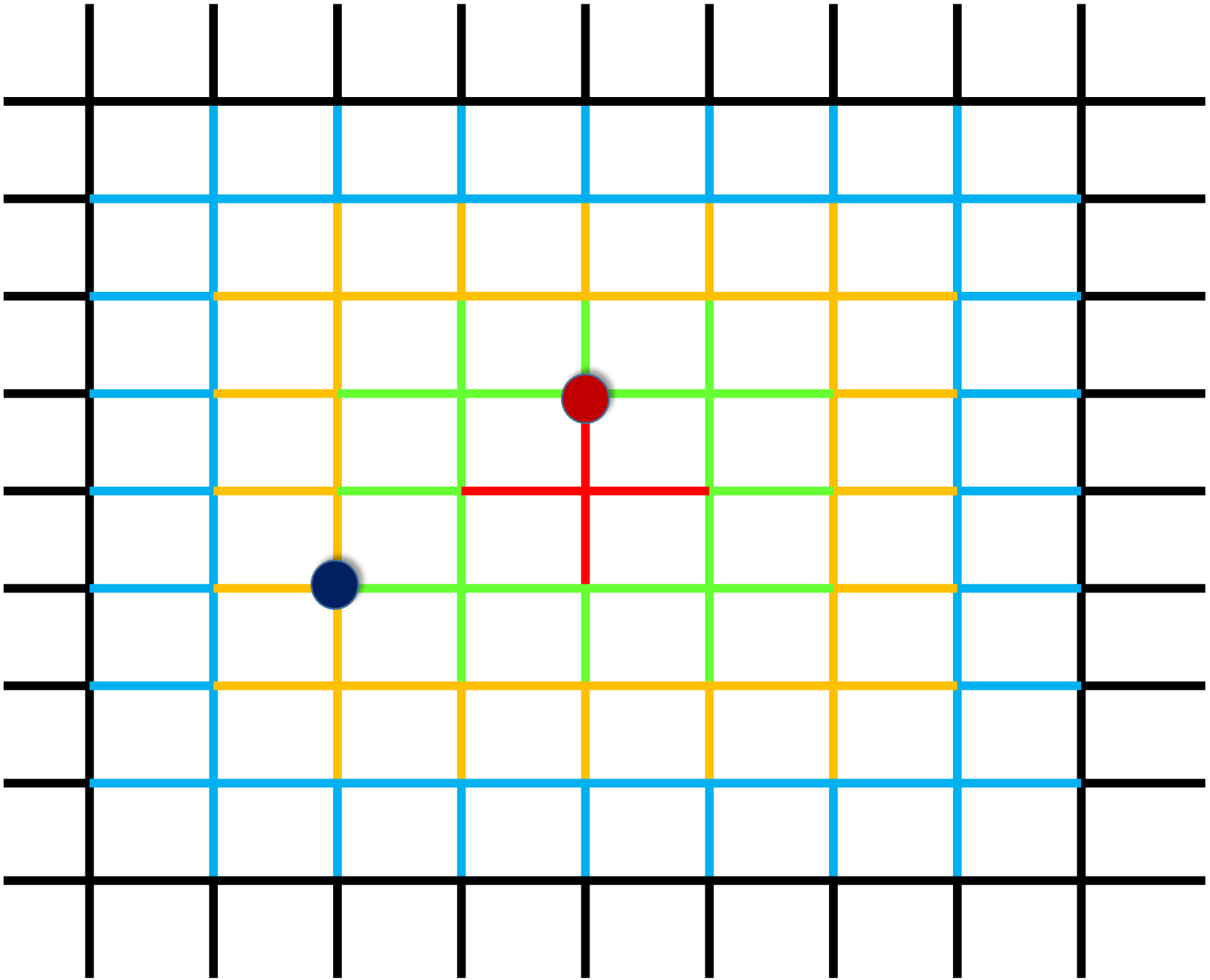
ıange the heap sort to include a heuristic function

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

ıoice of a good heuristic:

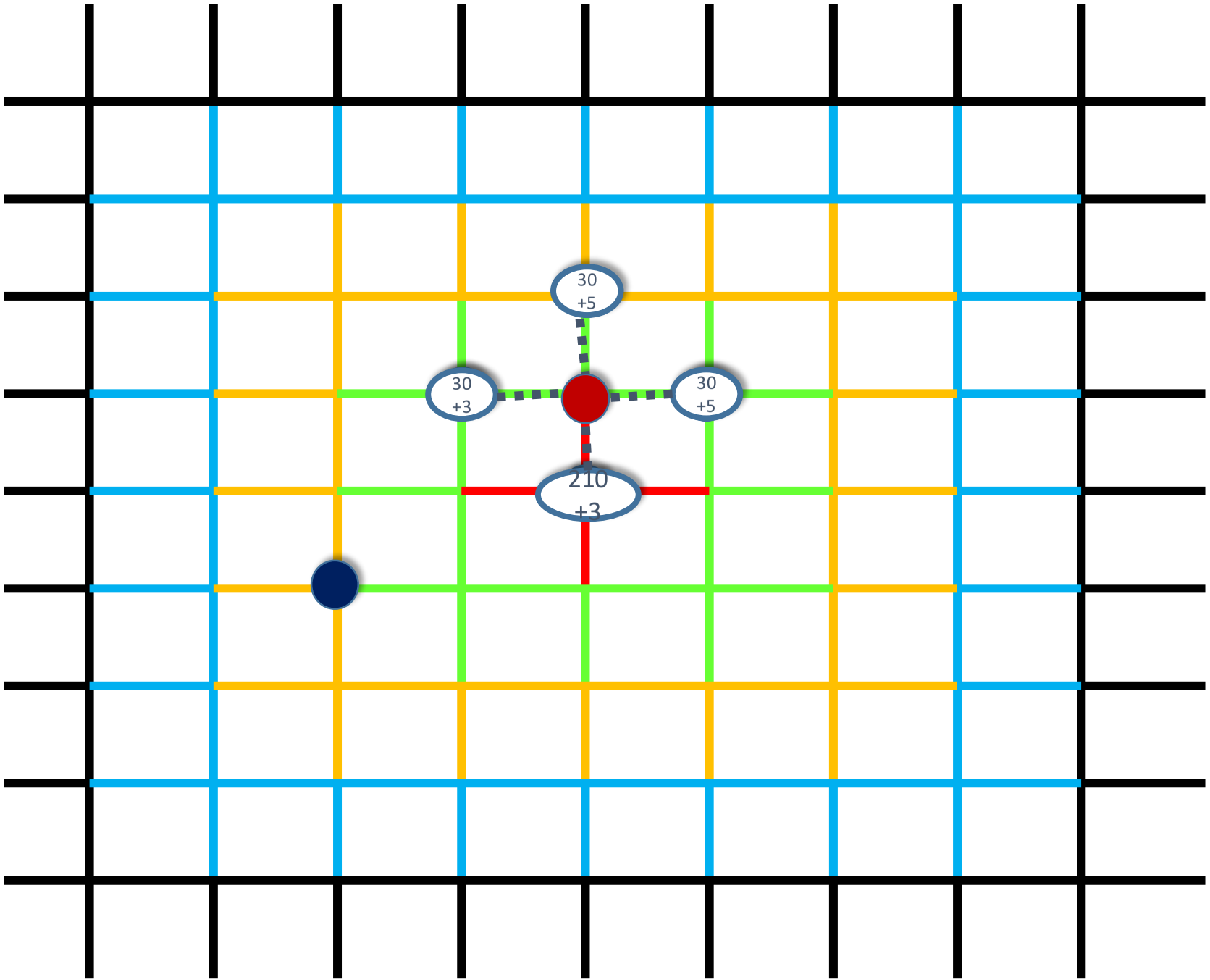
- Admissible: underestimates
- Consistent (strict): monotonic

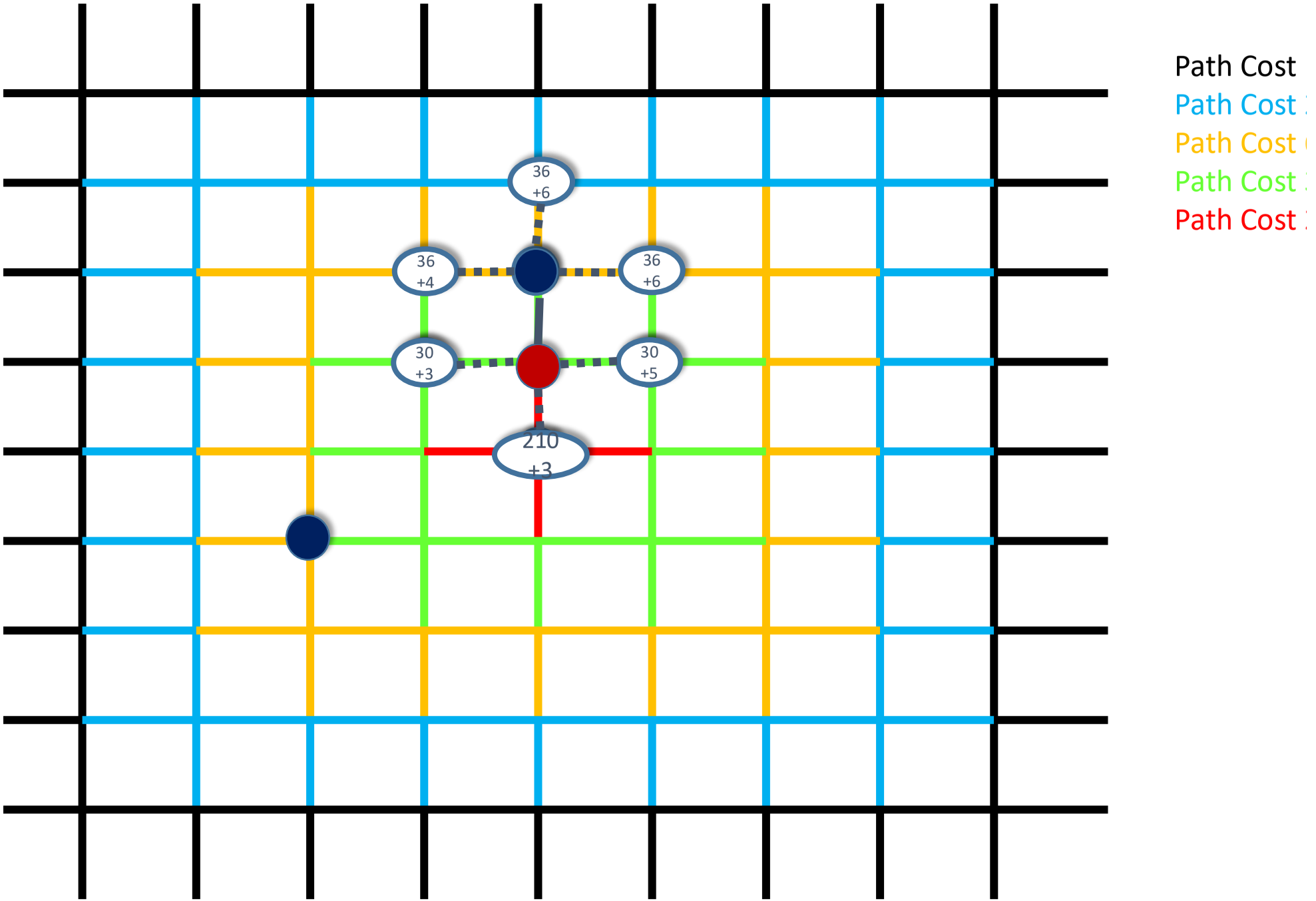
ıe better the heuristic, the quicker the search

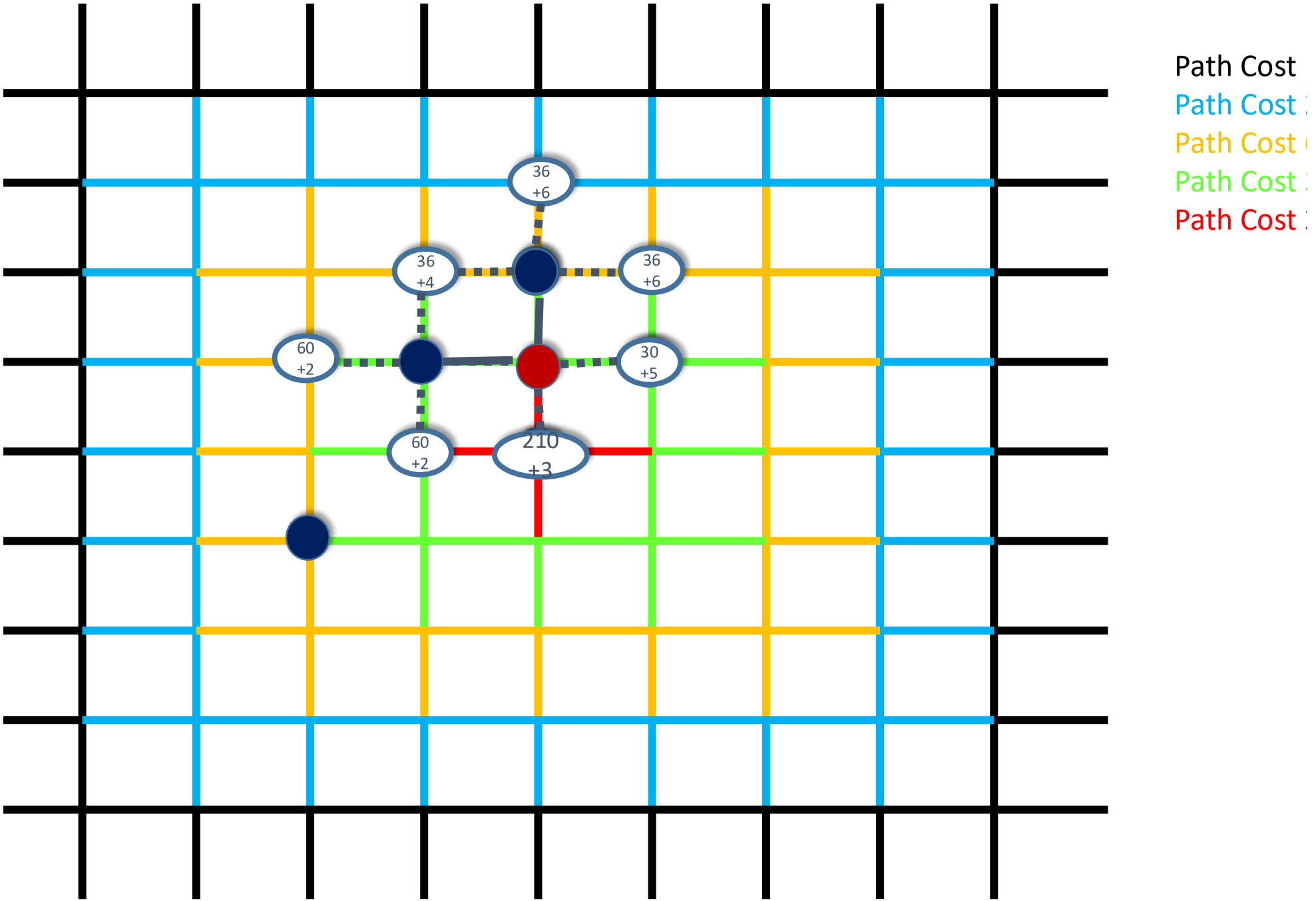


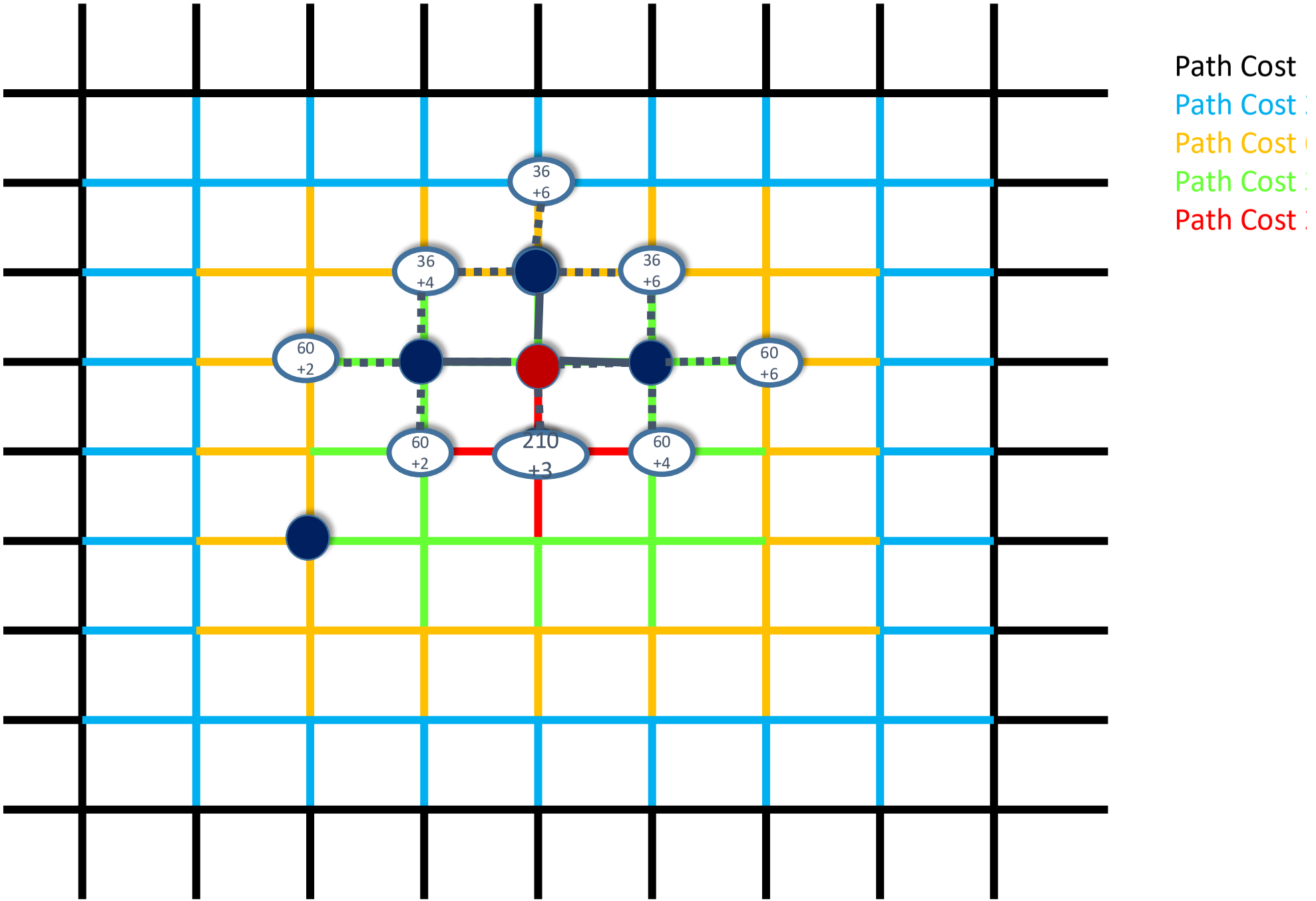
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

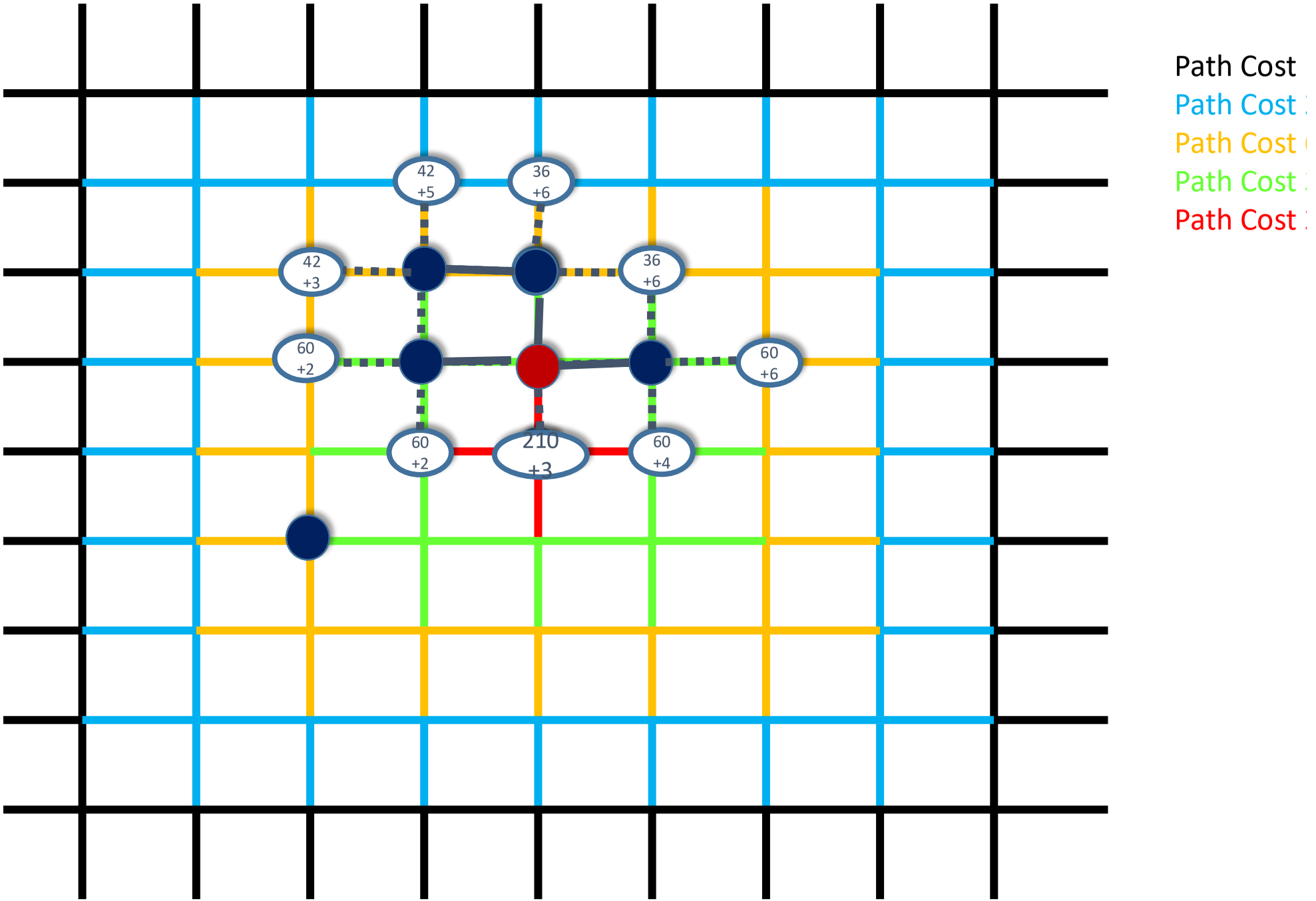
Notes:
- Manha
  Distanc
  Heurist

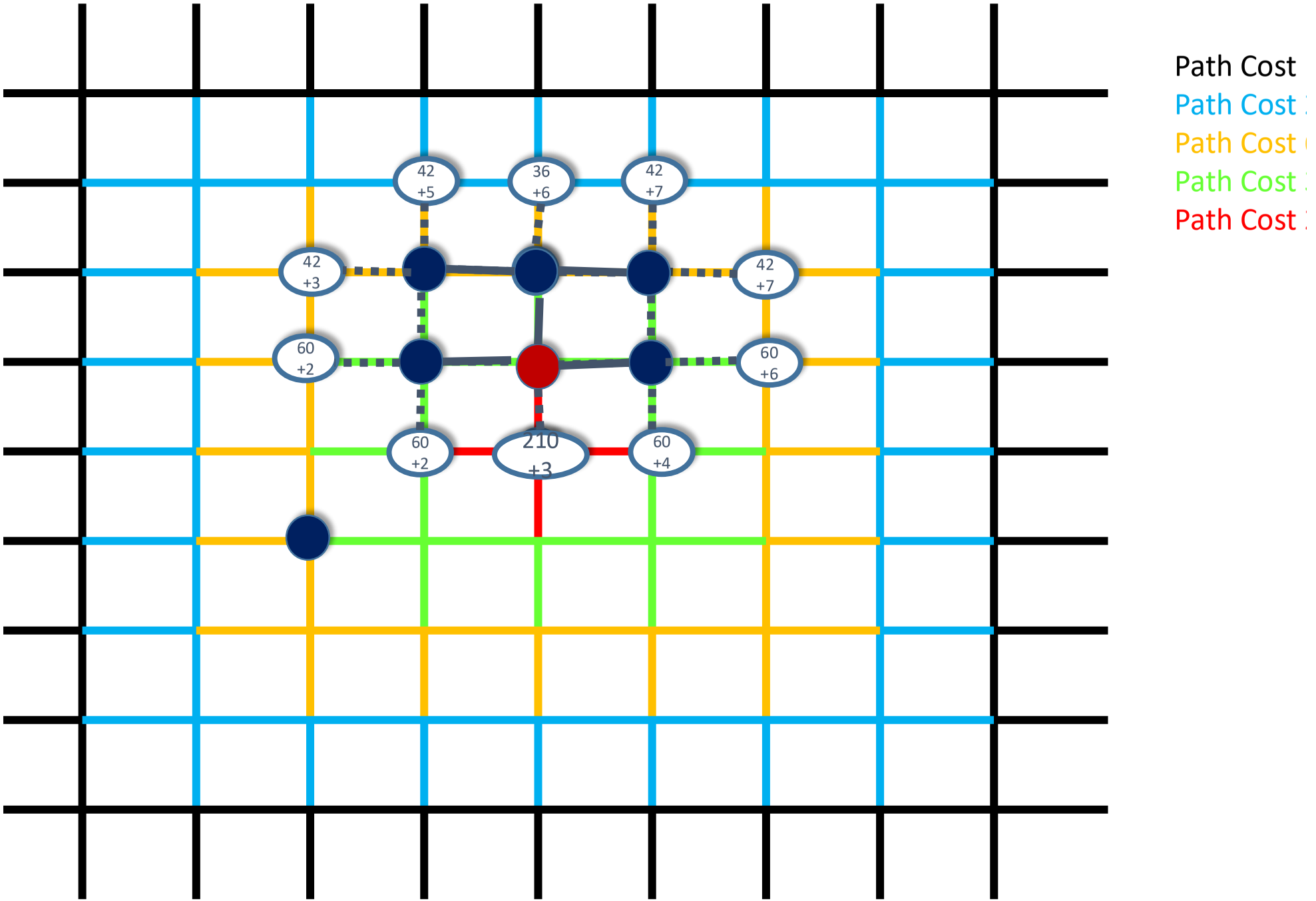

30
+5
30
+3
30
+5
210
+3
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

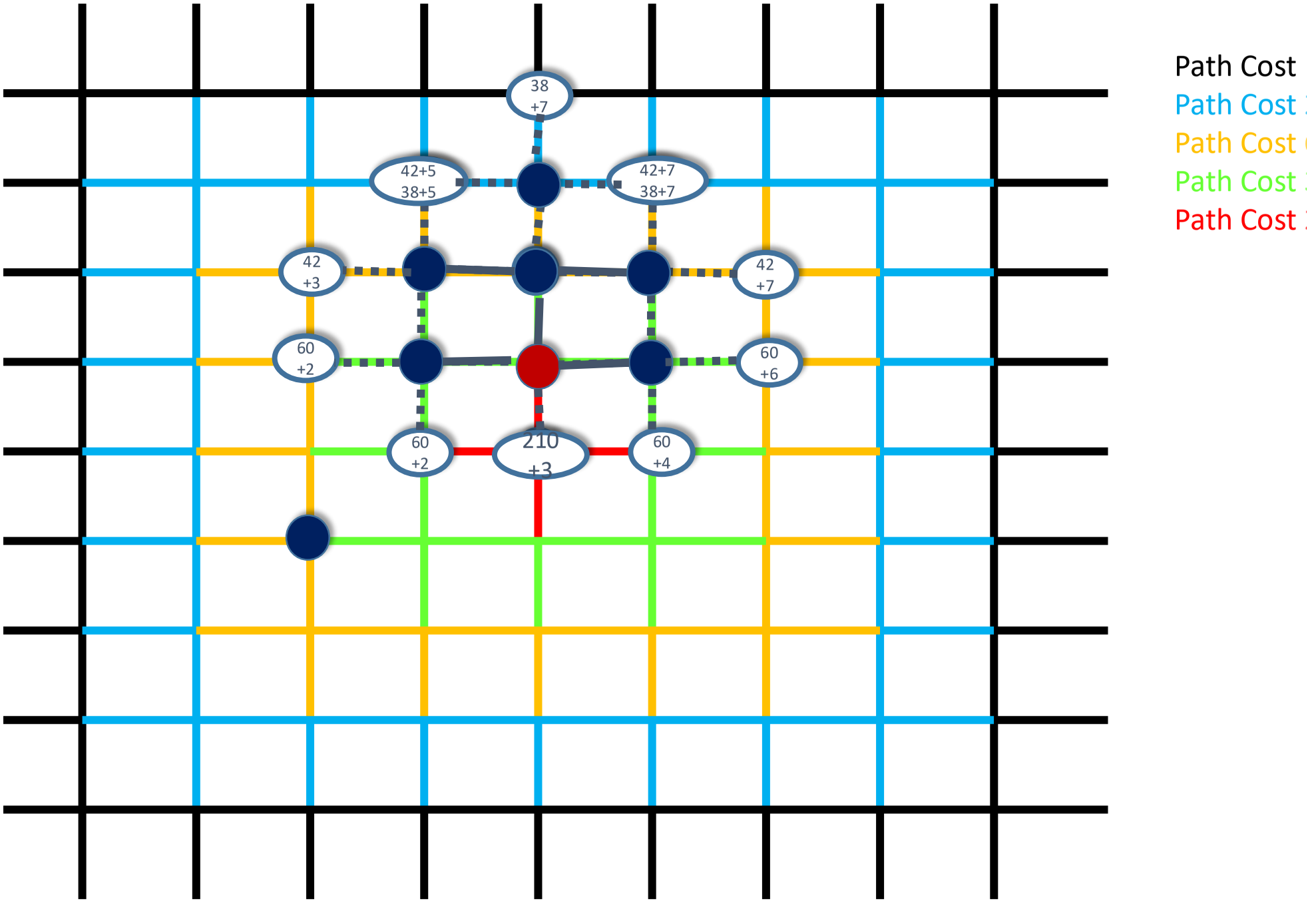

36
+6
36
+4
36
+6
30
+3
30
+5
210
+3
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

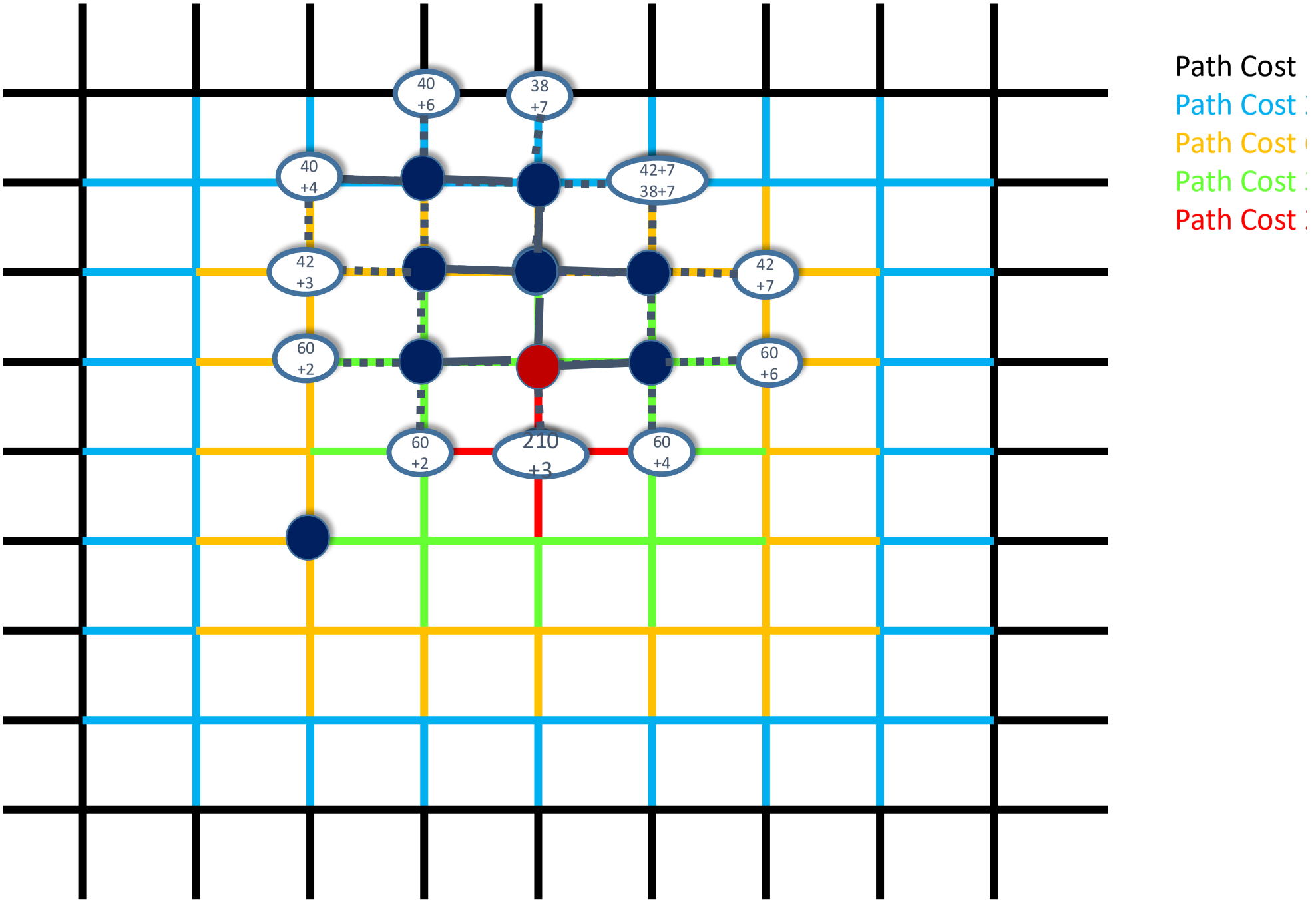

36
+6
36
+4
36
+6
60
+2
30
+5
60
+2
210
+3
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

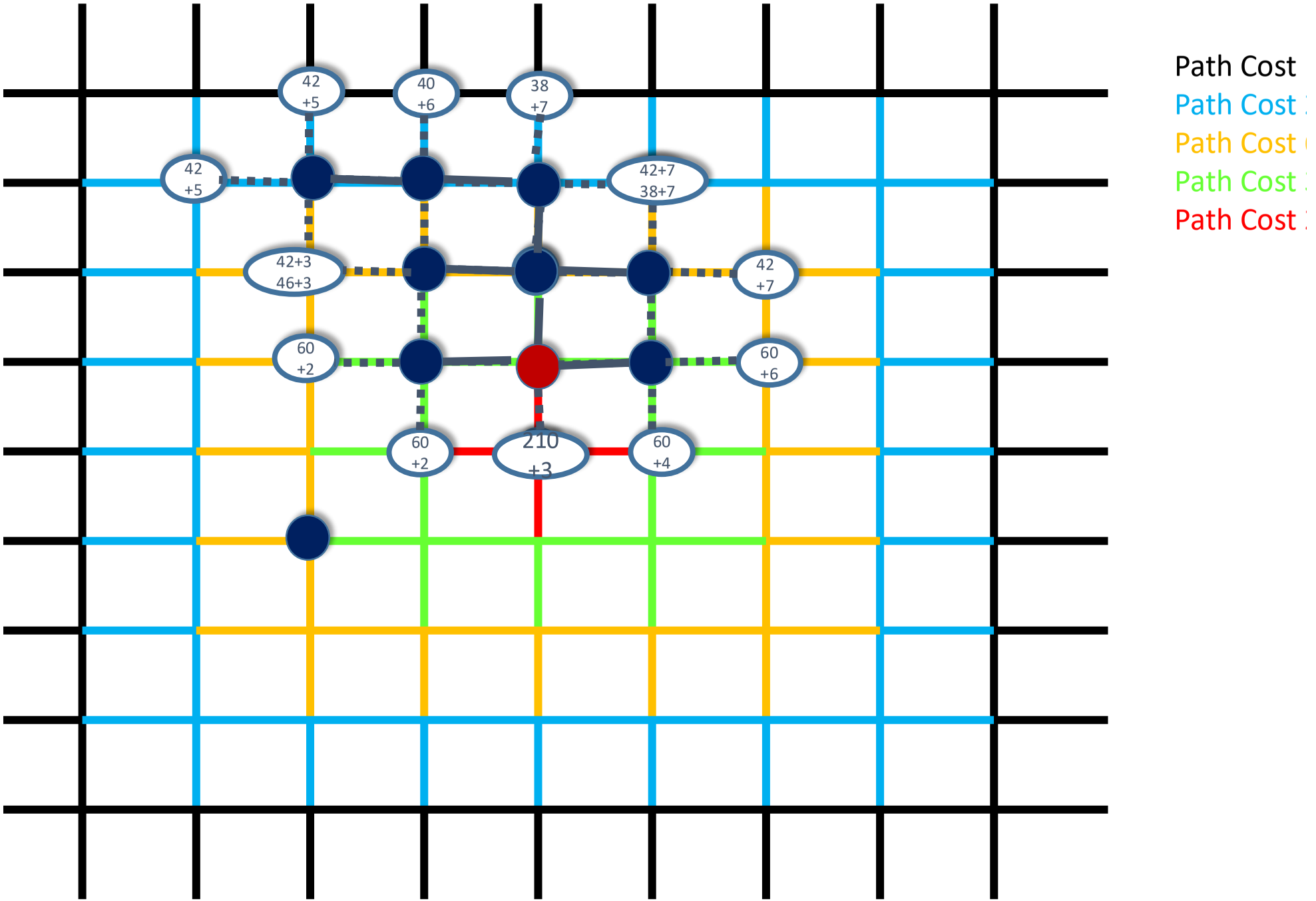

36
+6
36
+4
36
+6
60
+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

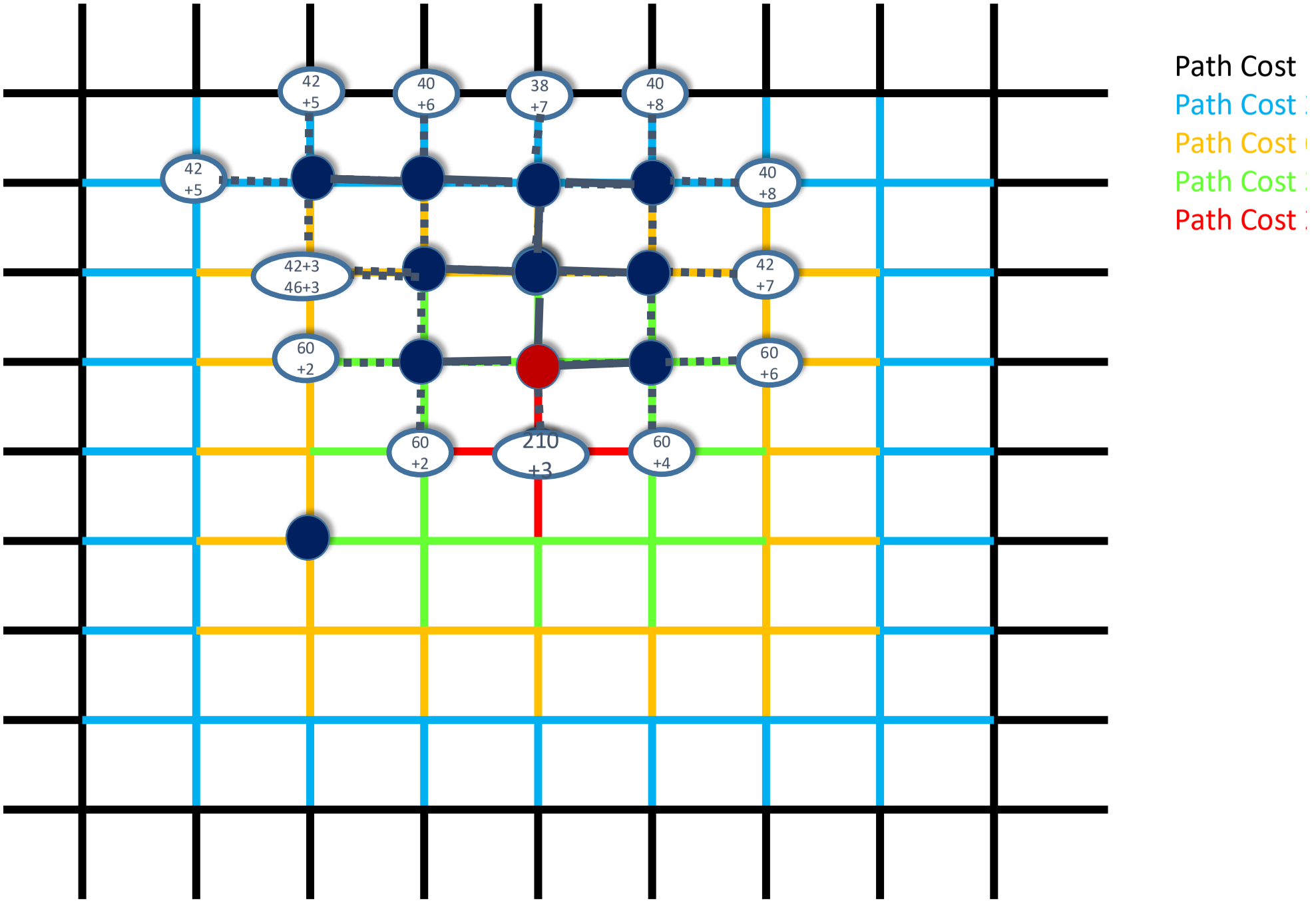

42
+5
36
+6
42
+3
36
+6
60
+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

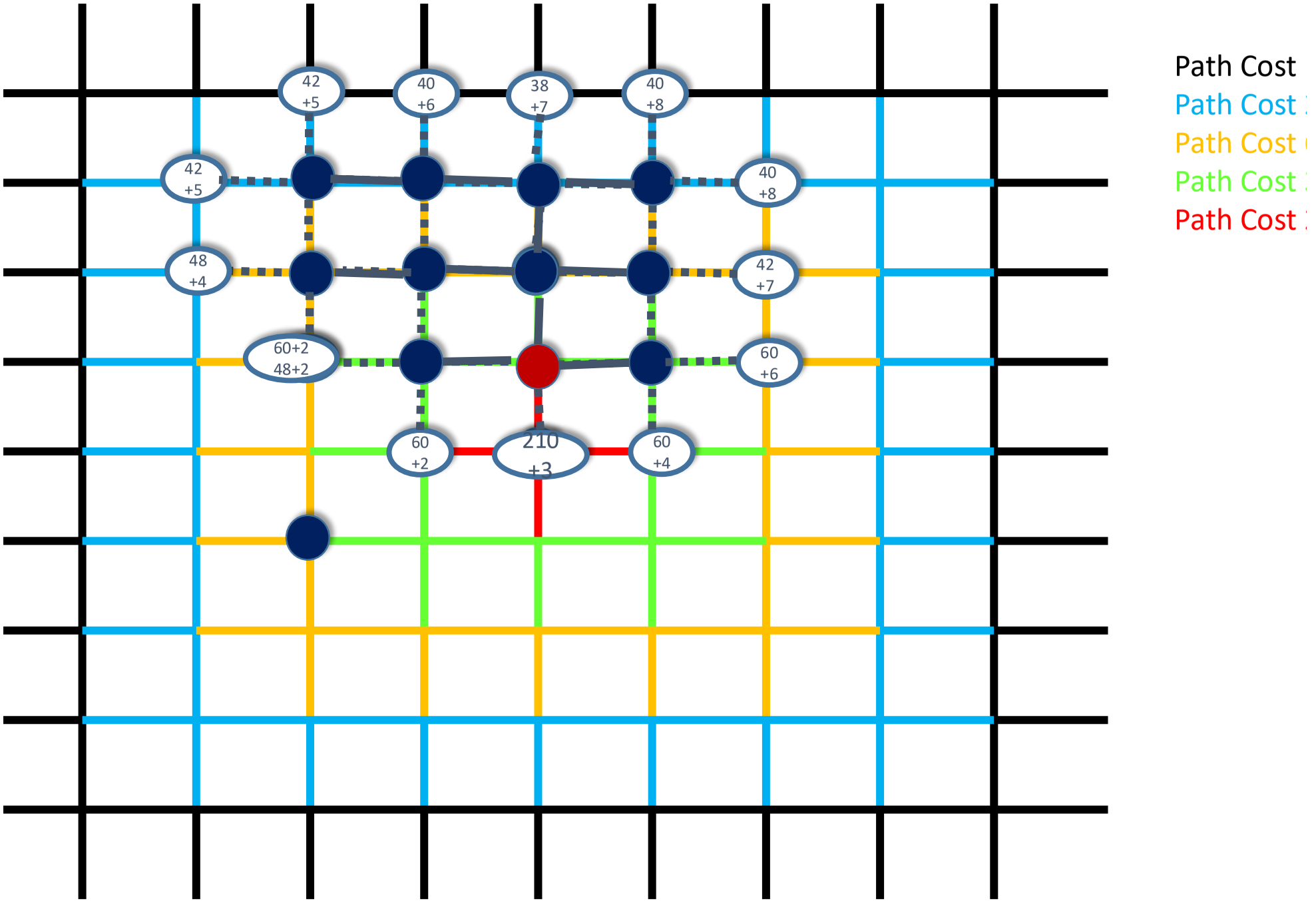

42
+5
36
+6
42
+7
42
+3
42
+7
60
+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

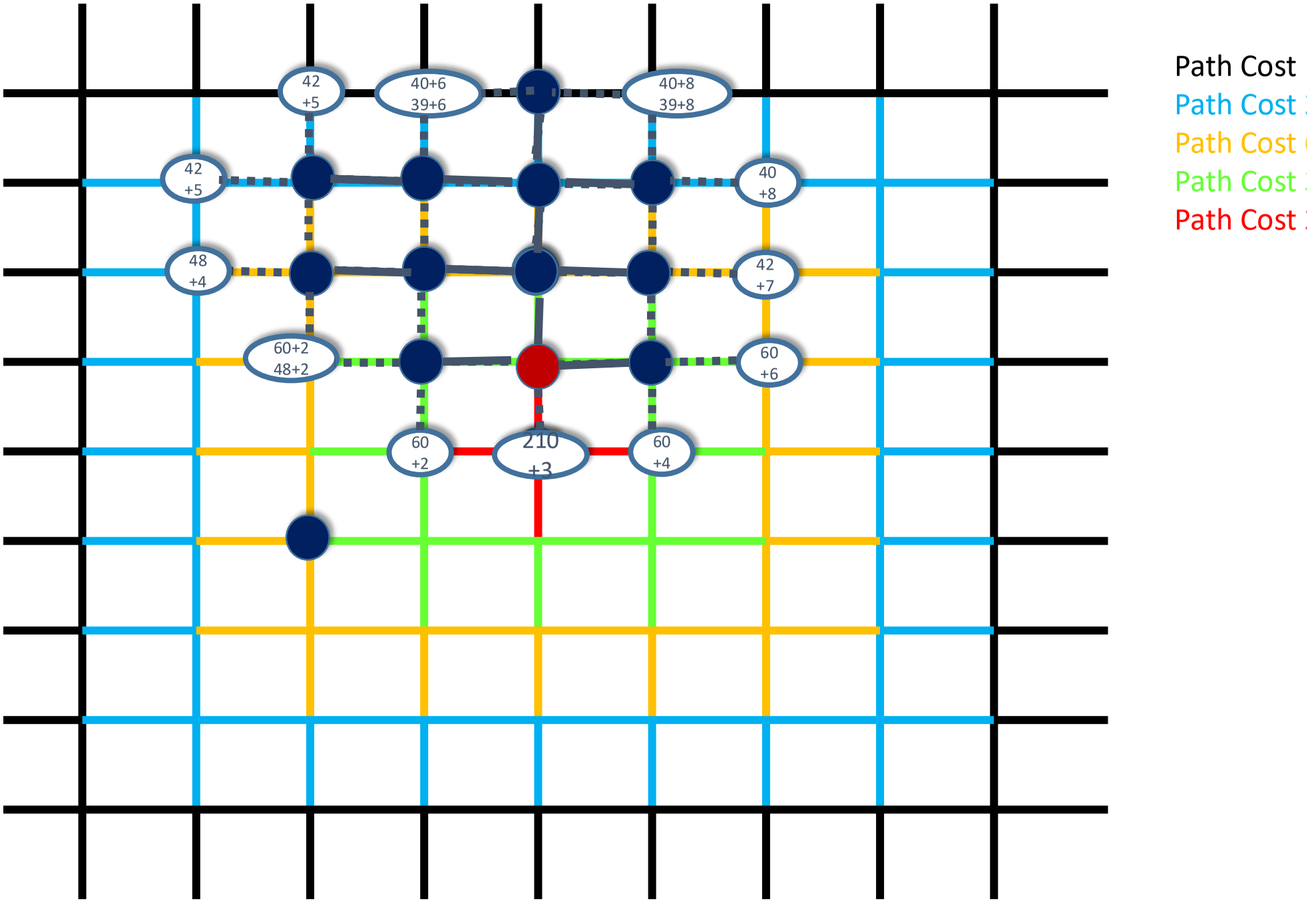

38
+7
42+5
38+5
42+7
38+7
42
+3
42
+7
60
+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

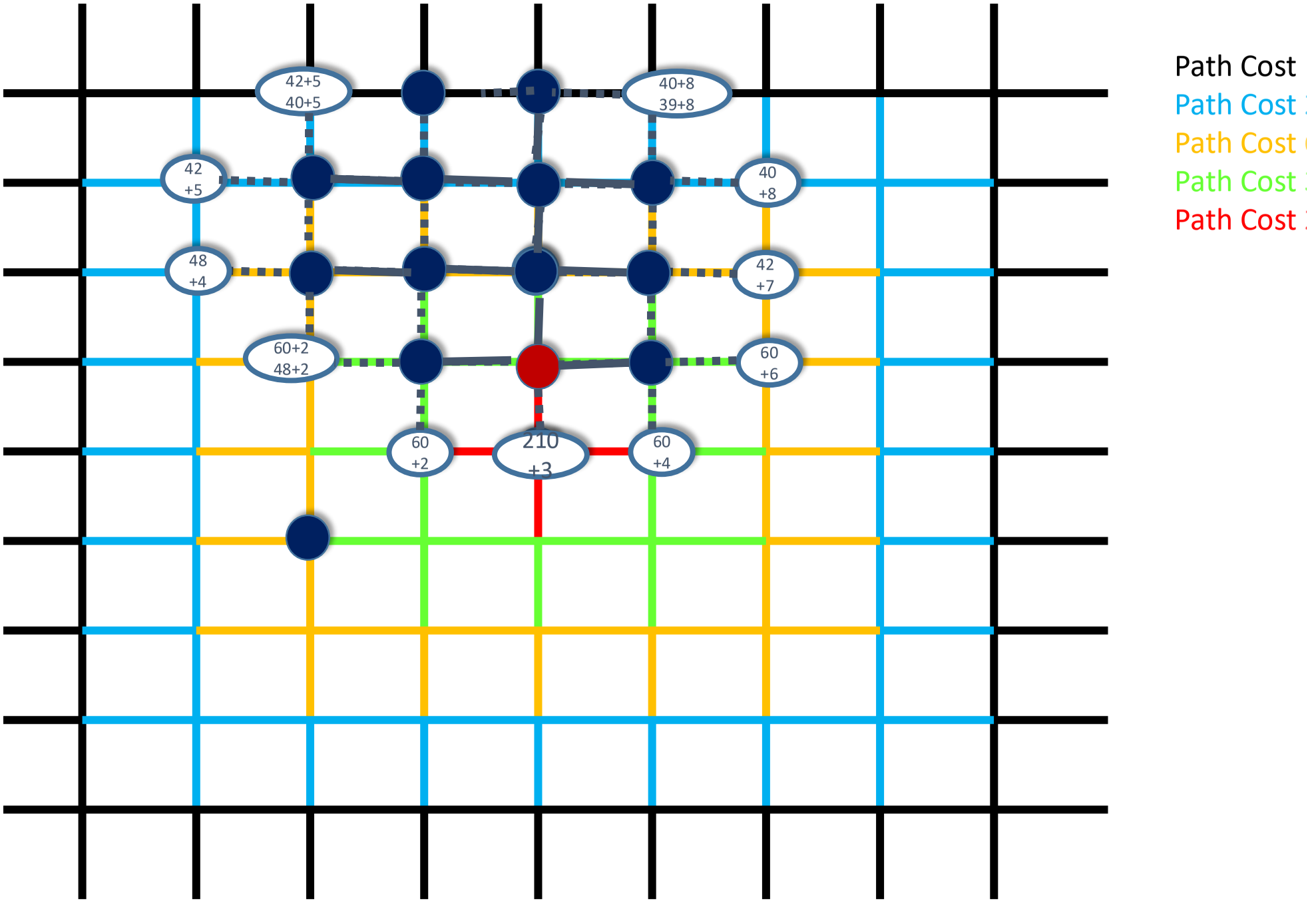

40
+6
38
+7
40
+4
42+7
38+7
42
+3
42
+7
60
+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

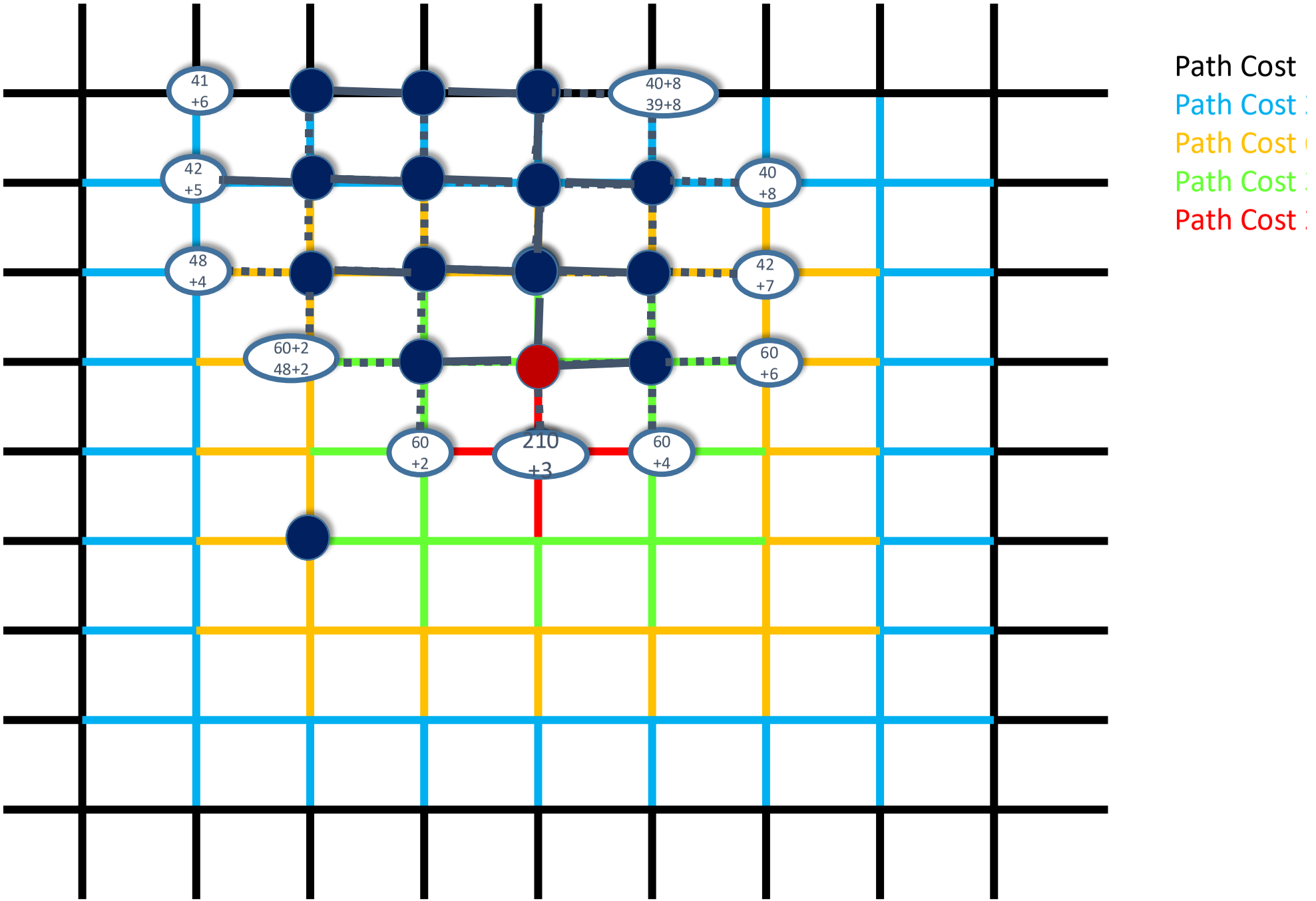

42
+5
40
+6
38
+7
42
+5
42+7
38+7
42+3
46+3
42
+7
60
+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

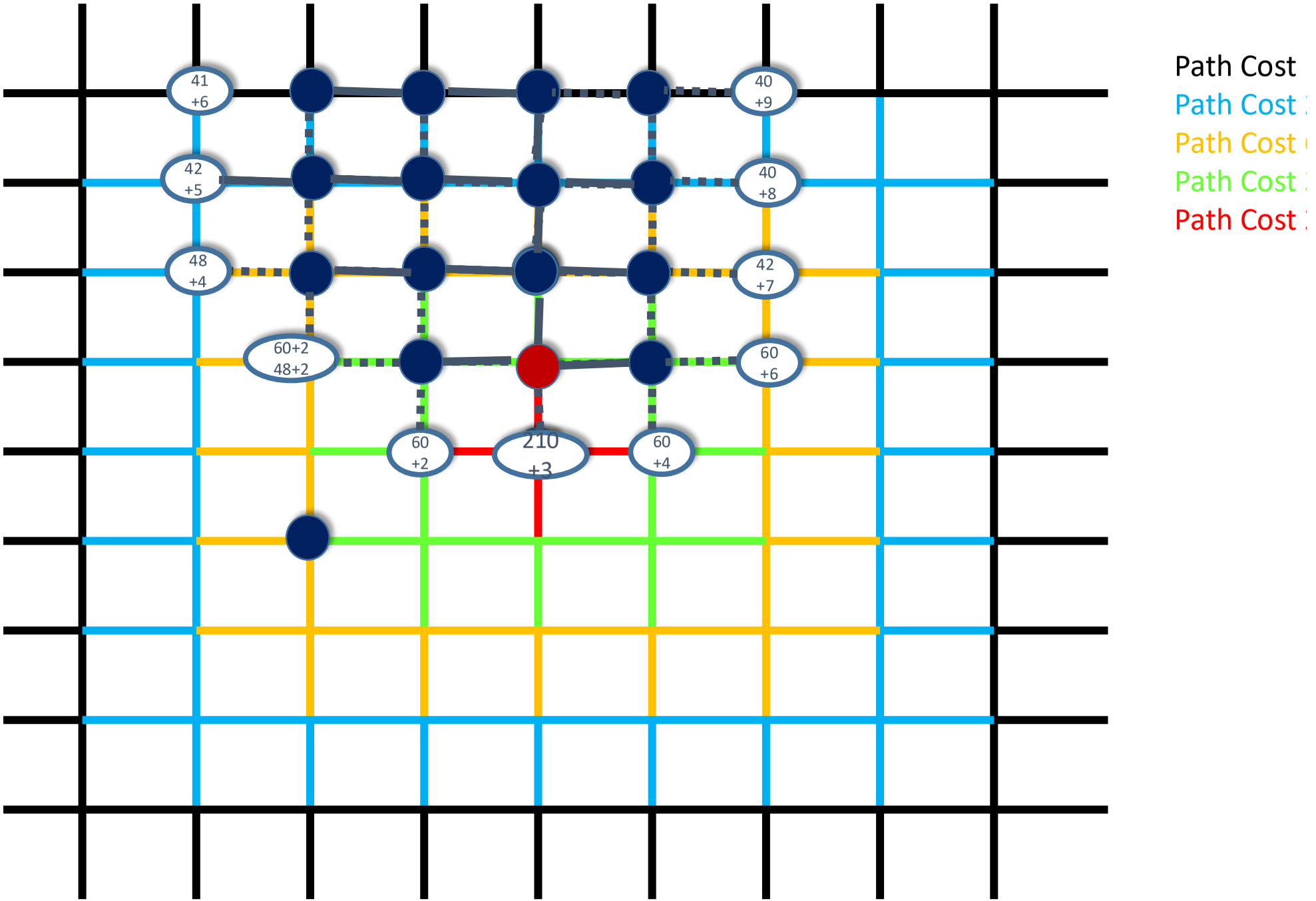

42
+5
40
+6
38
+7
40
+8
42
+5
40
+8
42+3
46+3
42
+7
60
+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

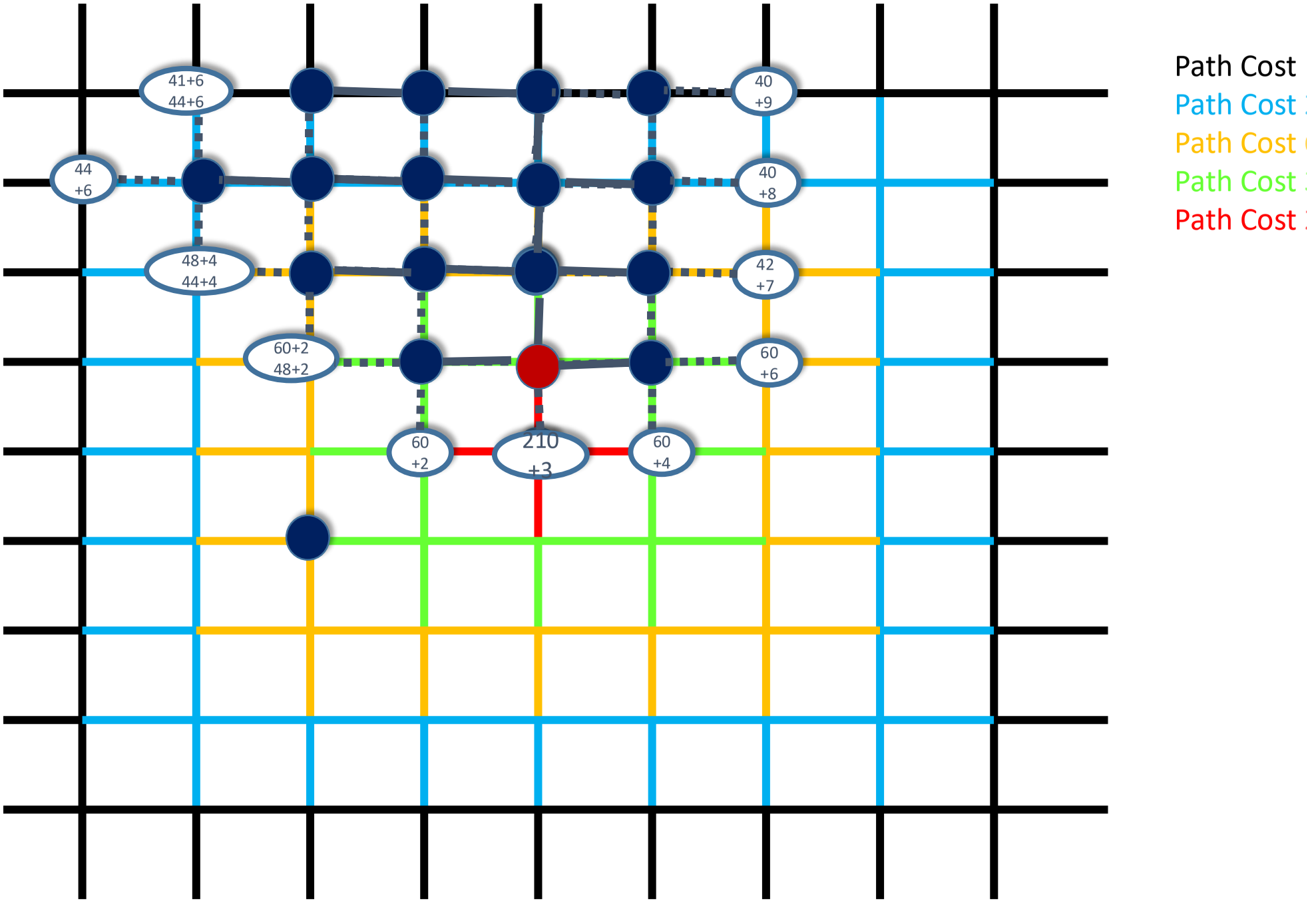

42
+5
40
+6
38
+7
40
+8
42
+5
40
+8
48
+4
42
+7
60+2
48+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

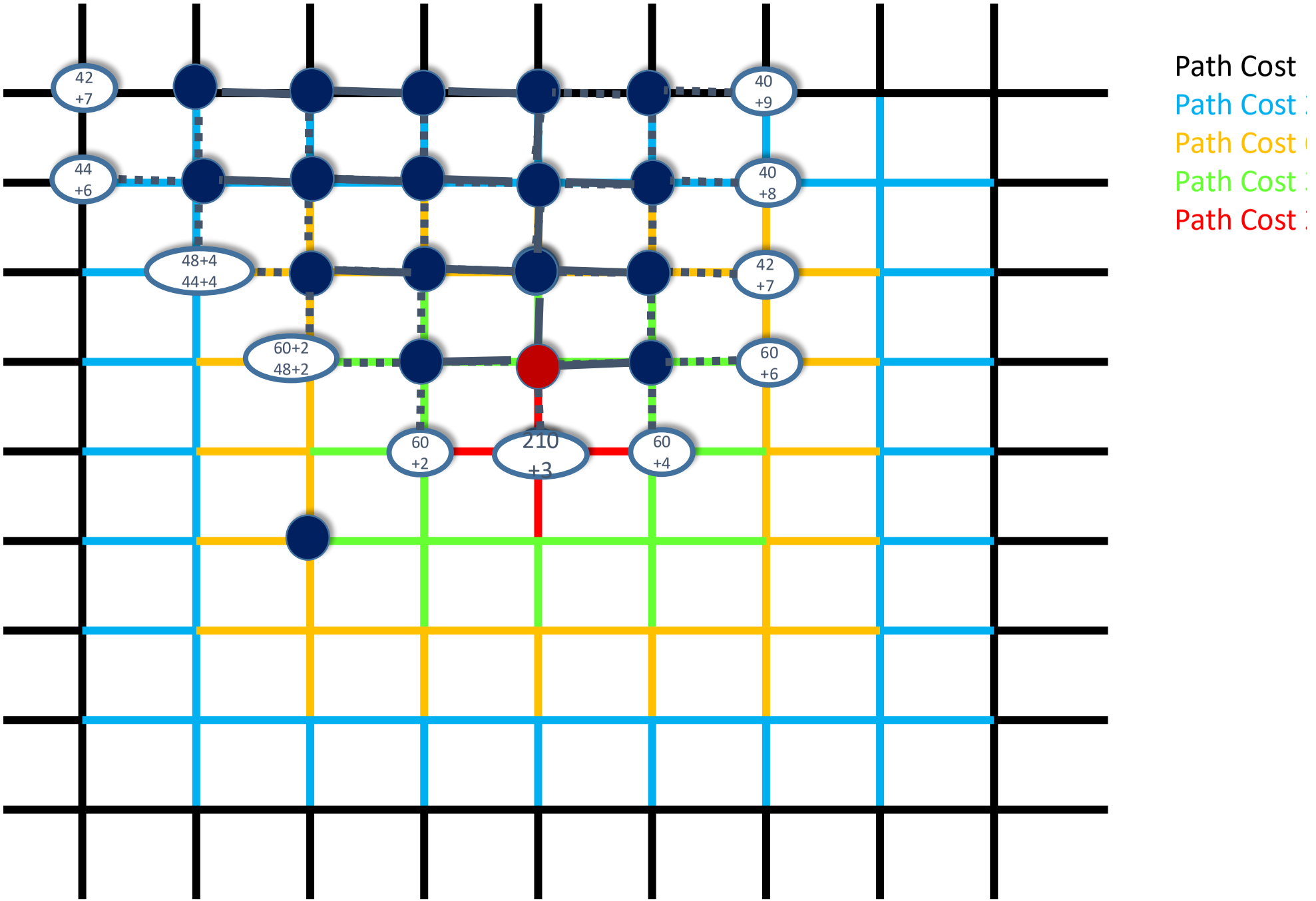

42
+5
40+6
39+6
40+8
39+8
42
+5
40
+8
48
+4
42
+7
60+2
48+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

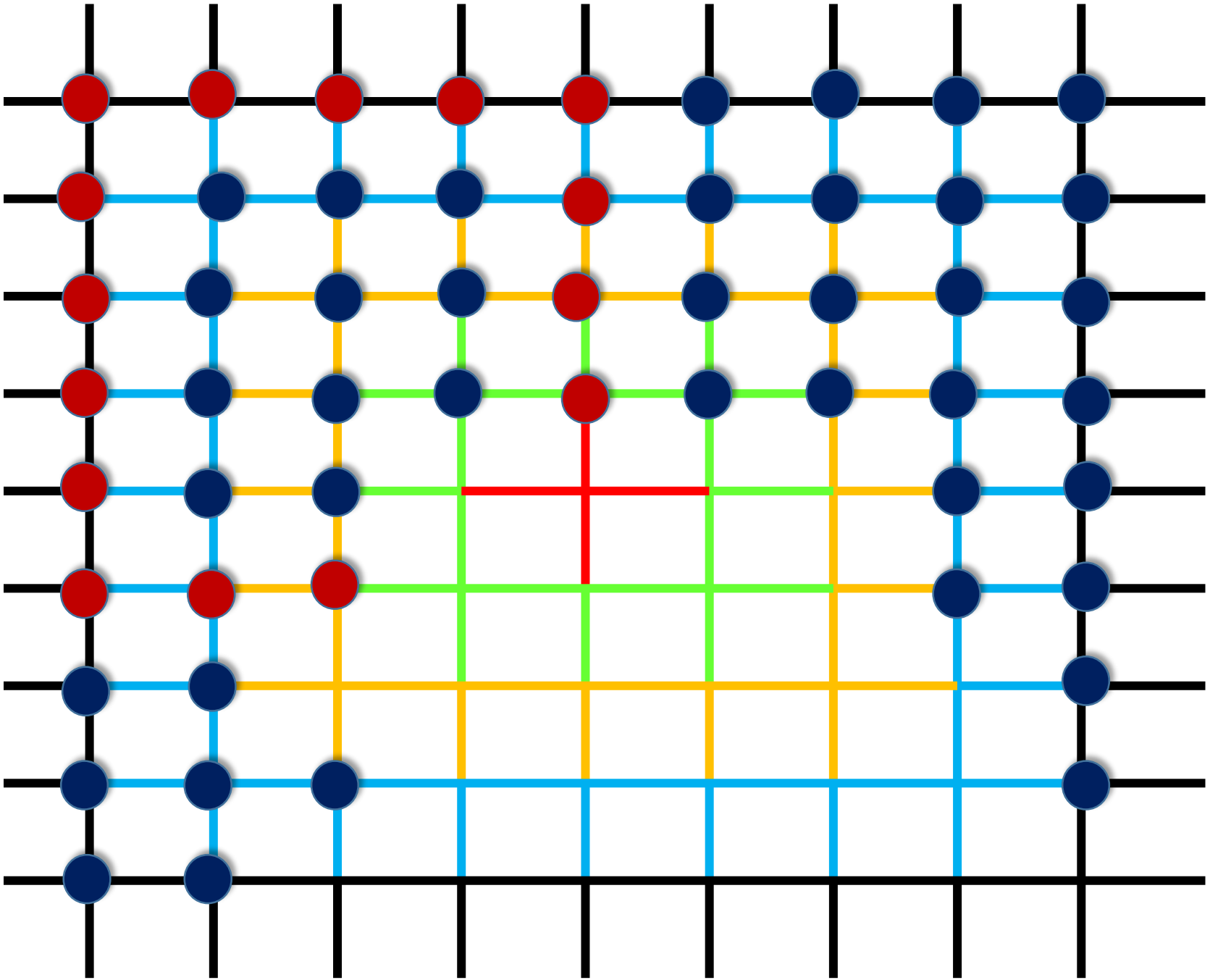

42+5
40+5
40+8
39+8
42
+5
40
+8
48
+4
42
+7
60+2
48+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost


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
Path Cost
Path Cost
Path Cost
Path Cost

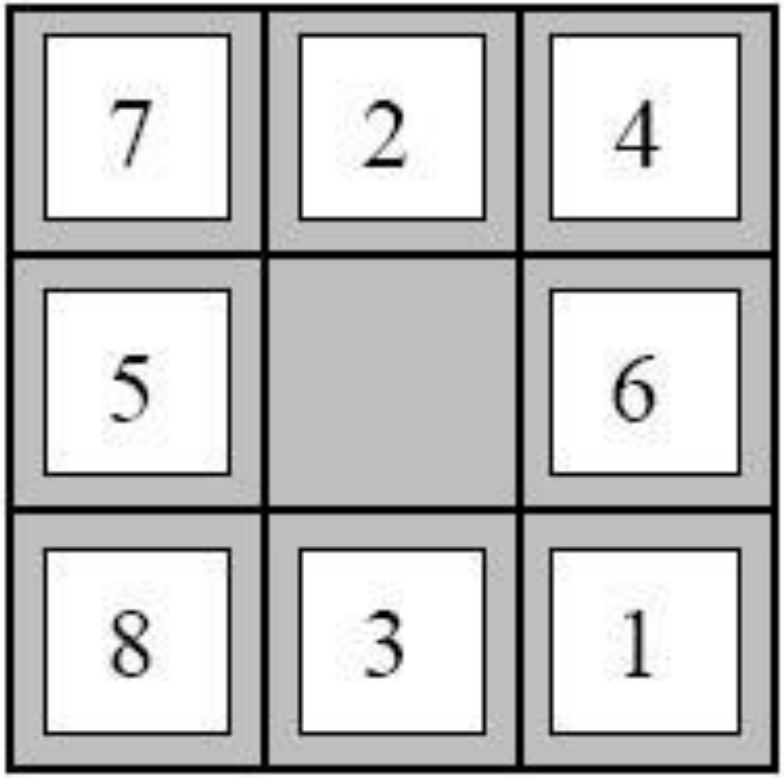

41
+6
40
+9
42
+5
40
+8
48
+4
42
+7
60+2
48+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost


41+6
44+6
40
+9
44
+6
40
+8
48+4
44+4
42
+7
60+2
48+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost

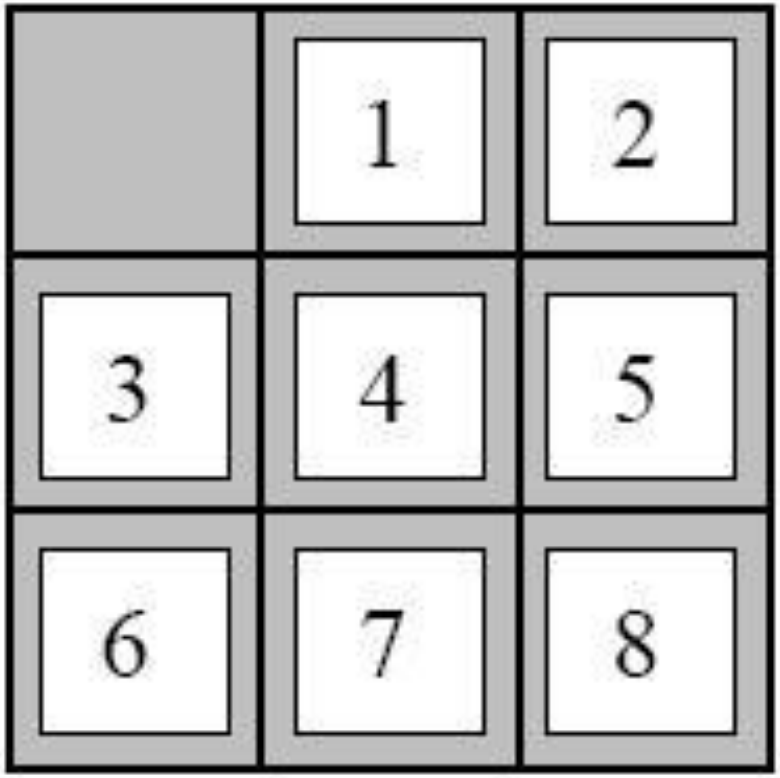

42
+7
40
+9
44
+6
40
+8
48+4
44+4
42
+7
60+2
48+2
60
+6
60
+2
210
+3
60
+4
Path Cost
Path Cost
Path Cost
Path Cost
Path Cost


Path Cost
Path Cost
Path Cost
Path Cost
Path Cost
Number E
Shortest C

hoosing Good Heuristics

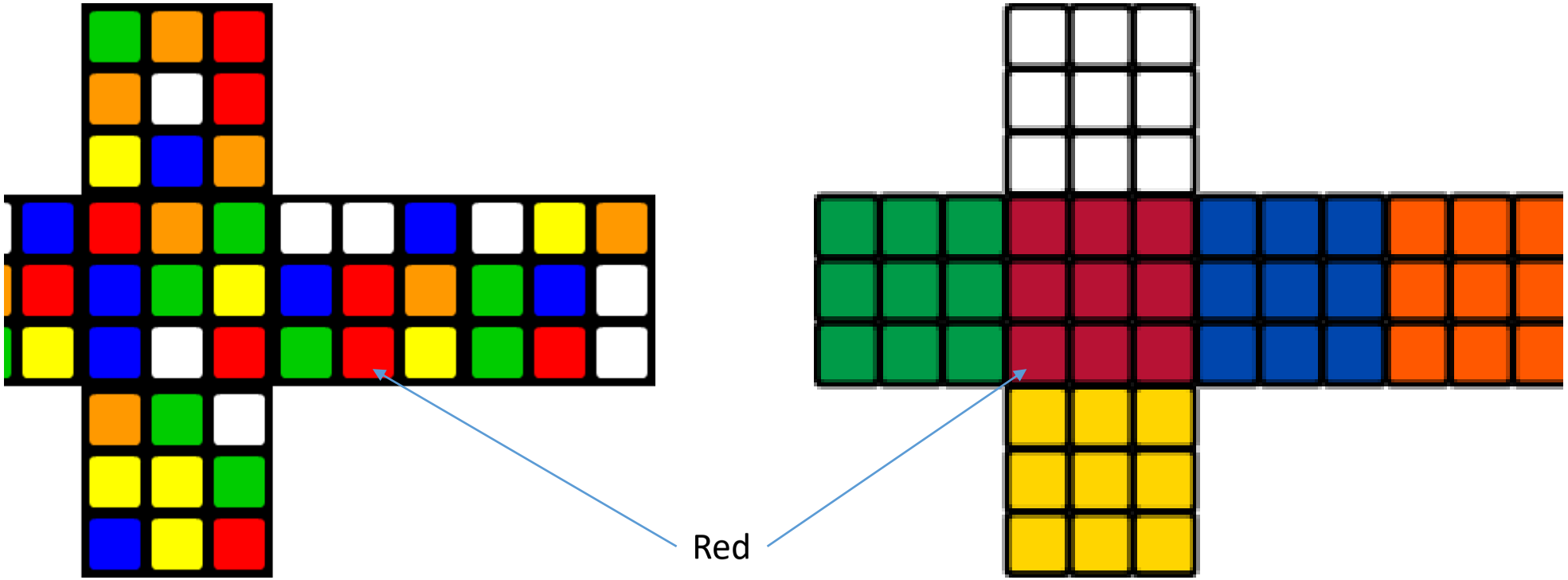


Start State



Goal State

Image from AI



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