COMP3888 - phase 1.2

October 2020

1 Introduction

This document describes an approximated algorithm.
Since the optimal algorithm has a high complexity due to predicting an best charging station, this one can possibly reduce the running time by only flying to charging stations when necessary.
Unchanged items including graph construction, simple battery - path strip in 1.1.

2 Detailed explanation

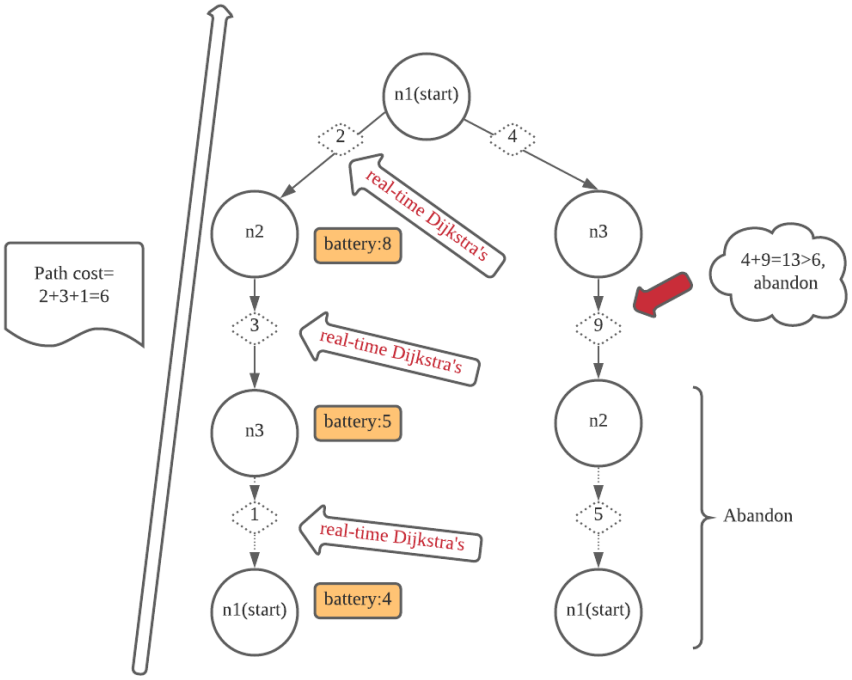
2.1 Change to Dijkstra's

Dijkstra's is **no longer performed prior to the main** branch-and-bound tree. It shall run together with it. When performing a (way point) to (way point) Dijkstra's

- There should be a starting battery provided to Dijkstra's
- Dijkstra's should ensure that the drone can fly to at least one charging station after reaching the way point it's heading
- Record this remaining battery life to feed to next Dijkstra's.

```
1
2 def branch_and_bound (current, current_battery, rest):
3
4     for node in rest:
5
6         # the detail of this step is in phase 2 document
7         next_battery = perform_Dijkstra(
8             start=current,
9             end=node,
10            start_battery=current_battery)
11
```

```
12          # recursive step (do loop when implementing)
13          if path_cost < current_best:
14              branch_and_bound (
15                  node,
16                  next_battery,
17                  rest.pop(node))
```



2.2 Complexity

The complexity of the approximated algorithm is reduced comparing to a full optimal workout. The core problem it solves is, when a drone runs out of battery and cannot use the next optimal routine, we avoid exploring all possibilities to charge the drone. Instead we just instruct the drone to charge when it requires to. This is an approximated algorithm which "thinks" in a more human-like fashion.
This recommended since it is easier to implement.