Review: Name Of The Paper

First1 Last1
email1@epfl.ch

First2 Last2
email2@epfl.ch

December 26, 2022

1 Introduction

Say a few general words about the general context of the paper you chose. Explain why the topic is of interest, or where it can be applied. If it is about a piece of software or artifact, give a description of it. State the main result of the paper and why it is new or how it improves on previous state of knowledge. You can cite references using, for example [1] and make a succint presentation of the organisation of your report.

2 Preliminaries

State the technical details that are necessary to understand the paper. It is generally a collection of definitions, concepts and notations with potentially a few preliminary results. It can be, for example the mathematical framework in which the topic of your paper is expressed. In particular, fix the notation you will be using for your review.

3 Body

Explain the paper, in your own words. Don't go into as many details as the original text, but the person reading your review should have a general understanding of the paper's results and how those results can be obtained. The structure and content of this section of course heavily depends on the paper itself. Don't hesitate to split it in multiple sections or subsections, for example:

3.1 Functional Graph

We represents graphs using associate lists of each vertex and its edges to its nearby edges with its distance (or weight).

```
case class Graph(graph:
    List[(Int, List[(Int, Distance)])])
```

definition of a valid graph

```
def validGraph(graph:
    List[(Int, List[(Int, Distance)])]): Boolean =
  noDuplicates(graph) &&
  graph.forall(e => noDuplicates(e._2)) &&
  graph.forall(e =>
    e._2.forall((i, _) => graph.get(i) != None())) &&
  graph.forall(n =>
    n._2.forall(z => 0.toDist <= z._2)) &&
  graph.forall { case (n, a) =>
        a.get(n) match {
        case None()  => true
        case Some(d) => d == 0.toDist
        }
}
```

3.2 The Algotirhm

the algorithm

```
def dijkstra(start: Int): List[Node] = {
    require(graph.get(start) != None())
    iterate(Nil[Node](), prepare(start))
}
```

3.2.1 Representing distance

Distance used in Dijistra algorithm, a distance is either infinite or a non-negative number.

```
sealed abstract class Distance
case object Inf extends Distance
case class Real(i: BigInt) extends Distance {
    require(i >= 0)
}
```

We also need to define addition and comparsion bewtween Distance, which is just integer addditon and comparsion with infinity.

3.2.2 A verified `getMin`

One important step in the dijkstra algorithm is extracting the node with minimial distance to the source, so we need to build a pure function that return the node that is closest to the source and the rest of nodes.

```
def getMin(l: List[Node]): (Node, List[Node]) = {
  require(l != Nil())
  l match
    case Cons(h, t) => getMinAux(h, t, Nil[Node]())
} ensuring (res =>
  res._2.size == l.size - 1 &&
    res._2.content ++ Set(res._1) == l.content &&
    res._2.map(_._1).content ++ Set(res._1._1) ==
      l.map(_._1).content &&
    res._2.forall(n => res._1._2 <= n._2)
)
```

Here we would call a helper function `getMinAux` which carries the min value, the traversed list and the rest to get the result. Ensuring that the remaining list is 1 shorter, the set of content of the remaining list with min node equals the original content of the original list. Finally the most important property is the all the nodes in the remaining list should have a longer distance than the we min one we get.

In this way, we ensure that the `getMin` function is correctly implemented.

3.2.3 Phase 1: prepare

prepare will generate a list of nodes, which is a list tuples associating each vertex's distance to the source node.

```
def prepare(start: Int): List[Node] = {
    require(graph.get(start) != None())
    prepareAux(graph, start)
} ensuring (res => prepareProp(res, graph, start))

def prepareAux(
    graph: List[(Int, List[Node])],
    start: Int
): List[Node] =
```

3.2.4 Phase 2: main loop

```
def iterate(seen: List[Node],
    future: List[Node]): List[Node]
```

4 Future Work

5 Conclusion

Recall briefly what the paper achieves, and how it is new. Express your critical skil on the paper and explain what you think are the strong and weak points of the paper. Also tell how you could potentially use the paper's results in your future project. You can also suggest further work or extensions to the paper.

References

[1] Bibliography management in LaTeX. https://overleaf.com/learn/latex/Bibliography_management_in_LaTeX.