POLITECNICO
MILANO 1863

Foundation Operation Research
Walking Bus Challenge

Professor. Malucelli Federico, Tresoldi Emanuele

Erba Alessandro
Festa Biagio

A.A 2016/2017

Contents

1 Walking Bus Challenge Report **2**
1.1 Abstract 2
1.1.1 note about the code 2
1.2 General Idea of the Solver 2
1.3 HESolver 3
1.4 A Algorithm Solver 3
1.5 GASolver 6

Chapter 1

Walking Bus Challenge Report

1.1 Abstract

This document aims to explain how we have accomplished to the annual Foundation Operations Research Challenge .[1]

We will introduce to our solution giving the rational of each one of the solver that we have implemented. The document is divided in sections. In section one we will explain the general idea of the whole solution purposed. In the other sections we will explain implemented algorithms. For each one of these we will point out the pros and cons aspect found during our research.

1.1.1 note about the code

The solution is completely written using C++ 11.
We have developed around 3000 lines of code. The solution is tested on Linux x64 but it can be easily ported in other OS.

1.2 General Idea of the Solver

The purposed challenge is a multi objective problem. First the number of leaves, second the dangerousness of the path.

Given the first objective we have thought that the most useful approach for building the solver was to express it as a Search problem. At the same time the second objective suggested the use of a Genetic Algorithm in order to decrease the danger and the number of leafes.

The Solver is composed of three different Algorithms:

[1]Walking Bus Challenge available on Beep channel "Foundations Operation Research [Federico Malucelli]"

- HESolver: Find a feasible solution building paths going from the outskirts nodes to the school node.
- ASolver: Find a feasible solution building paths going from the school node to the outskirts nodes.
- GASolver: Compute refinements given the population provided by HESolver and ASolver.

1.3 HESolver

The gerneral idea of this solver is to find all the paths stating search from the furthest node from the School Node.

In order to do that we compute the distances of all nodes to the school. Starting from the first node (the furthest) we choose the node to attach to it using an Heuristic that orders all the free nodes. The heuristic function is composed of 6 different heuristic functions, each one estimates a different parameter. Specifically these heuristic parameters are:

- grade of isolation of a point
- distance from the evaluator point
- the node is considering the attach to a node alrady in a path
- factor scale if the considered node is near to school
- how much the connection deviates from the linear connection from node to the school

In this way every time we add a node to a path we reorder the frontier of attachable node wrt the last attached node. This technique resembles the Greedy Best First [2] united to a Depth First Search[3].

pros & cons

Pros: Most intuitive solution
Cons: Elevate Branching factor leads efficiency in case of Backtracking.

1.4 A Algorithm Solver

This algorithm starts form the School Node and orders all the other free nodes from the nearest to the furthest. It connects to the school the nearest node. Than connects all the others (according to their distance to the school node).

[2]https://en.wikipedia.org/wiki/Best-first_search
[3]https://en.wikipedia.org/wiki/Depth-first_search

First it tries to connect new nearest to an existent path, if none of the existent path is suitable for the considered node it creates a new one.

An important aspect that we have take in to account is determinism of this algorithm. The order of the attempts of connection to an existent path is defined by the euclidean distance between considered node and the leaf of the considered path. The first path tried is the one which leaf is the nearest to the node that we want to attach. In this way every time that we compute the algorithm we obtain the same solution.

In order to find other feasible solutions and avoid local minimum, we have introduced scaled randomic permutations that perturbs the order of found path list according to random probabilities.

In Figure 1 we show an example of what randomness perturbation technique can allow. The node 5 is the next node that should be connected to a path. According to the deterministic ordering of connection attempts, we will first try to connect 5 to path that has as leaf 4, and that to the path that has as leaf 3. Assume that the first attempt succeeds, every time that we are in this situation we will connect 5 -> 4. If we consider a wider Node space, are we sure that it is the best connection possible? Of course not.

So if we introduce randomness perturbation we can break determinism and sometimes we can try to attach 5 -> 3 before trying the connection 5 -> 4. If we execute the ASolver for a wide number of time we can find solutions that are better than the deterministic one.

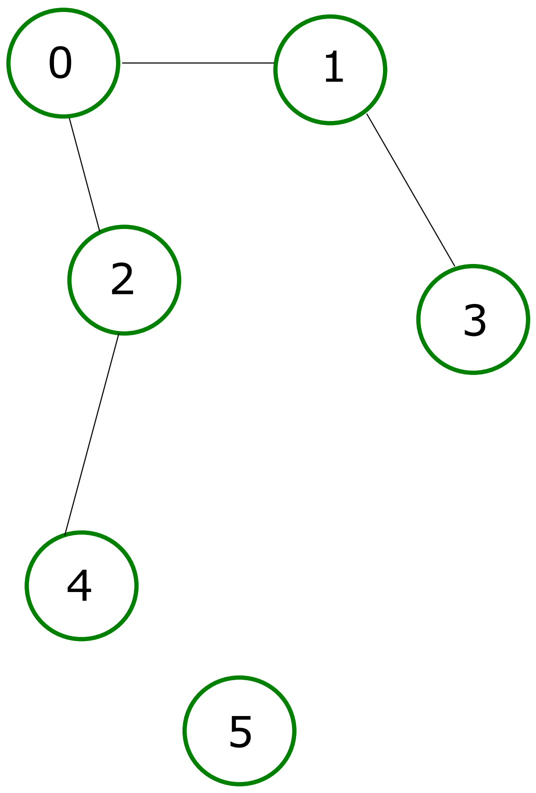

Figure 1.1: ASolver randomness example

pros & cons

Pros: fast search in the node space
Cons: none

1.5 GASolver

This is the genetic algorithm. It uses best solutions from the solutions found by the HESolver and ASolver.

Using Crossover and Mutations it can easily decrease the Danger of the solution and sometimes the number of leaves.