Experimental Analyses of Heuristics for Horsefly-type Problems

Gaurish Telang

Contents

Part I

Overview

Chapter 1

Description of Problems

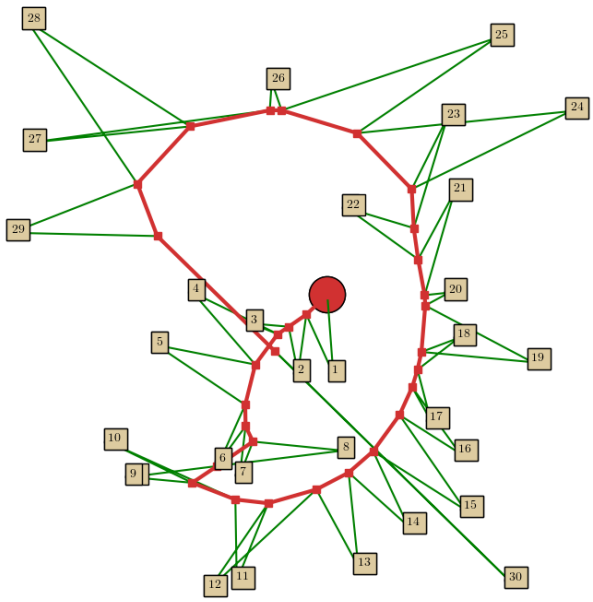

Figure 1.1: An Example of a classic Horsefly tour with $\varphi = 5$. The red dot indicates the initial position of the horse and fly, given as part of the input. The ordering of sites shown has been computed with a greedy algorithm which will be described later

The Horsefly problem is a generalization of the well-known Euclidean Traveling Salesman Problem. In the most basic version of the Horsefly problem (which we call **"Classic Horsefly"**), we are given a set of sites, the initial position of a truck(horse) with a drone(fly) mounted on top, and the speed of the drone-speed φ. [1] [2].

The goal is to compute a tour for both the truck and the drone to deliver package to sites as quickly as possible. For delivery, a drone must pick up a package from the truck, fly to the site and come back to the truck to pick up the next package for delivery to another site. [3]

[1] The speed of the truck is always assumed to be 1 in any of the problem variations we will be considering in this report.

[2] φ is also called the "speed ratio".

[3] The drone is assumed to be able to carry at most one package at a time

Both the truck and drone must coordinate their motions to minimize the time it takes for all the sites to get their packages. Figure 1.1 gives an example of such a tour computed using a greedy heuristic for $\varphi = 5$.

This suite of programs implement several experimental heuristics, to solve the above NP-hard problem and some of its variations approximately. In this short chapter, we give a description of the problem variations that we will be tackling. Each of the problems, has a corresponding chapter in Part 2, where these heuristics are described and implemented. We also give comparative analyses of their experimental performance on various problem instances.

Classic Horsefly This problem has already described in the introduction.

Segment Horsefly In this variation, the path of the truck is restricted to that of a segment, which we can consider without loss of generality to be $[0, 1]$. All sites, without loss of generality lie in the upper-half plane $\mathbb{R}^2_+$.

Fixed Route Horsefly This is the obvious generalization of Segment Horsefly, where the path which the truck is restricted to travel is a piece-wise linear polygonal path. [4] Both the initial position of the truck and the drone are given. The sites to be serviced are allowed to lie anywhere in $\mathbb{R}^2$. Two further variations are possible in this setting, one in which the truck is allowed reversals and the other in which it is not.

One Horse, Two Flies The truck is now equipped with two drones. Otherwise the setting, is exactly the same as in classic horsefly. Each drone can carry only one package at a time. The drones must fly back and forth between the truck and the sites to deliver the packages. We allow the possibility that both the drones can land at the same time and place on the truck to pick up their next package. [5]

Reverse Horsefly In this model, each site (not the truck!) is equipped with a drone, which fly *towards* the truck to pick up their packages. We need to coordinate the motion of the truck and drone so that the time it takes for the last drone to pick up its package (the "makespan") is minimized.

Bounded Distance Horsefly In most real-world scenarios, the drone will not be able to (or allowed to) go more than a certain distance R from the truck. Thus with the same settings as the classic horsefly, but with the added constraint of the drone and the truck never being more than a distance R from the truck, how would one compute the truck and drone paths to minimize the makespan of the deliveries?

[4] More generally, the truck will be restricted to travelling on a road network, which would typically be modelled as a graph embedded in the plane.

[5] In reality, one of the drones will have to wait for a small amount of time while the other is retrieving its package. In a more realisting model, we would need to take into account this "waiting time" too.

Chapter 2

Installation and Use

To run these programs:

A. [*Get Docker*] Docker is open-source containerization software that is easily installable on Windows[1], Mac, and almost any GNU/Linux distribution. For a quick minute introduction to Docker containerization, please watch the first couple of minutes of https://youtu.be/_dfLOzuIg2o

For installation instrutions watch

On GNU/Linux https://youtu.be/KCckWweNSrM

On Windows https://youtu.be/ymlWt1MqURY

On MacOS https://youtu.be/MU8HUVlJTEY

B. [*Download customized Ubuntu image*] Download a customized Ubuntu Docker image that I've created with `docker pull gtelang/Ubuntu_customized`. The customized Ubuntu image is approximately 7 GB which contains all the libraries (e.g. CGAL, VTK, numpy, and matplotlib) that I typically use to run my research codes portably. [2]

C. [*Clone repository*] `git clone gtelang/horseflies.git`

D. [*Build the downloaded Ubuntu image*] Open up your favorite terminal emulator, like say xterm on GNU/Linux, iTerm on macOS or Powershell on Windows 10 and type `cd horseflies ; docker build .` [3]

E. [*Run all experiments again*] If you want to run all the experiments as described in the paper again to reproduce the reported results on your machine [4], type in `docker run`

[1] You might need to turn on virtualization explicitly in your BIOS, as I needed to when installing Docker on Windows. Here is a snapshot of an image when turning on Intel's virtualization technology through the BIOS: https://images.techhive.com/images/article/2015/09/virtualbox_vt-x_amd-v_error04_phoenix-100612961-large.idge.jpg

[2] On my home internet connection downloading this Ubuntu-image typically takes about 5 minutes.

[3] Note the period at the end! This has to be typed in too!

[4] Allowing, of course, for differences between your machine's CPU and mine when it comes to reporting absolute running time

F. [*Interactive mode*] Else if you want to test the algorithms in interactive mode (where you get to select the problem-type, mouse-in the sites on a canvas, set the initial position of the truck and drone and set φ) you will need to set up an X-connection between the Docker container and your host OS.

- On the Ubuntu 16.04 partition of my Thinkpad laptop, I had to follow the instructions given in this short video: https://youtu.be/RDg6TRwiPtg
- On the Windows 10 partition of the same laptop, I had to follow the instructions in https://dev.to/darksmile92/run-gui-app-in-linux-Docker-container-on-windows-host-4kde

If you want to delete the Ubuntu image, (it is 7GB afterall!) please watch this short video https://youtu.be/ySMAXT384bs.

The nice thing about Docker is that it makes it easy to run softwares on different OS'es portably, while also allowing to get rid of the problem of dependency hell https://en.wikipedia.org/wiki/Dependency_hell.

The headache of installing different software dependencies correctly on different machines running different OS'es, is replaced **only** by having to learn how to install Docker on your particular machine, and to know how set up an X-connection between the host OS and an instantiated container.

Saturday 15th December, 2018 15:51

Chapter 3

Overview of the Code Base

Part II

Programs

Chapter 4

Classic Horsefly

4.0.1 Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

"../src/lib/test.py" 11≡

```
⟨ Imports 12a, ... ⟩
# Fixing random state for reproducibility
np.random.seed(19680801)

dt = 0.01
t = np.arange(0, 10, dt)
nse = np.random.randn(len(t))
r = np.exp(-t / 0.05)

cnse = np.convolve(nse, r) * dt
cnse = cnse[:len(t)]
s = 0.1 * np.sin(2 * np.pi * t) + cnse

plt.subplot(211)
plt.plot(t, s)
plt.subplot(212)
plt.psd(s, 512, 1 / dt)

plt.show()
◇
```

File defined by 11, 12b.

4.0.2

⟨ *Imports* 12a ⟩ ≡

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.gridspec as gridspec
◇
```

Fragment defined by 12a, 13.
Fragment referenced in 11.

4.0.3 Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

`"../src/lib/test.py"` 12b≡

```
import numpy as np
import matplotlib.pyplot as plt


y = np.sin(np.arange(10.0) / 20.0 * np.pi) + 1
plt.errorbar(x, y, yerr=0.1, uplims=True)

y = np.sin(np.arange(10.0) / 20.0 * np.pi) + 2
plt.errorbar(x, y, yerr=0.1, uplims=upperlimits, lolims=lowerlimits)

plt.xlim(-1, 10)
◇
```

File defined by 11, 12b.

4.0.4 Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected

font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

⟨ *Imports* 13 ⟩ ≡

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.gridspec as gridspec
◇
```

Fragment defined by 12a, 13.
Fragment referenced in 11.

4.0.5

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.0.6

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 5

Segment Horsefly

Chapter 6

Fixed Route Horsefly

Chapter 7

One Horse, Two Flies

Chapter 8

Reverse Horsefly