ANALISA PENYELESAIAN TUGAS BESAR
MATAKULIAH ANALISA DAN DESAIN ALGOTIMA

STUDI KASUS TRAVELLING SALESMAN PROBLEM (TSP)

Disusun untuk Memenuhi Matakuliah Analisa dan Desain Algoritma

Yang dibimbing oleh Bapak Agusta Rakhmat Taufani, S.T., M.T.

Oleh:

SABRI SANGJAYA	160535611819
MUHAMMAD RIZAL R	160535611857
NIENDHITTA TAMIA LASSELA	160535611823

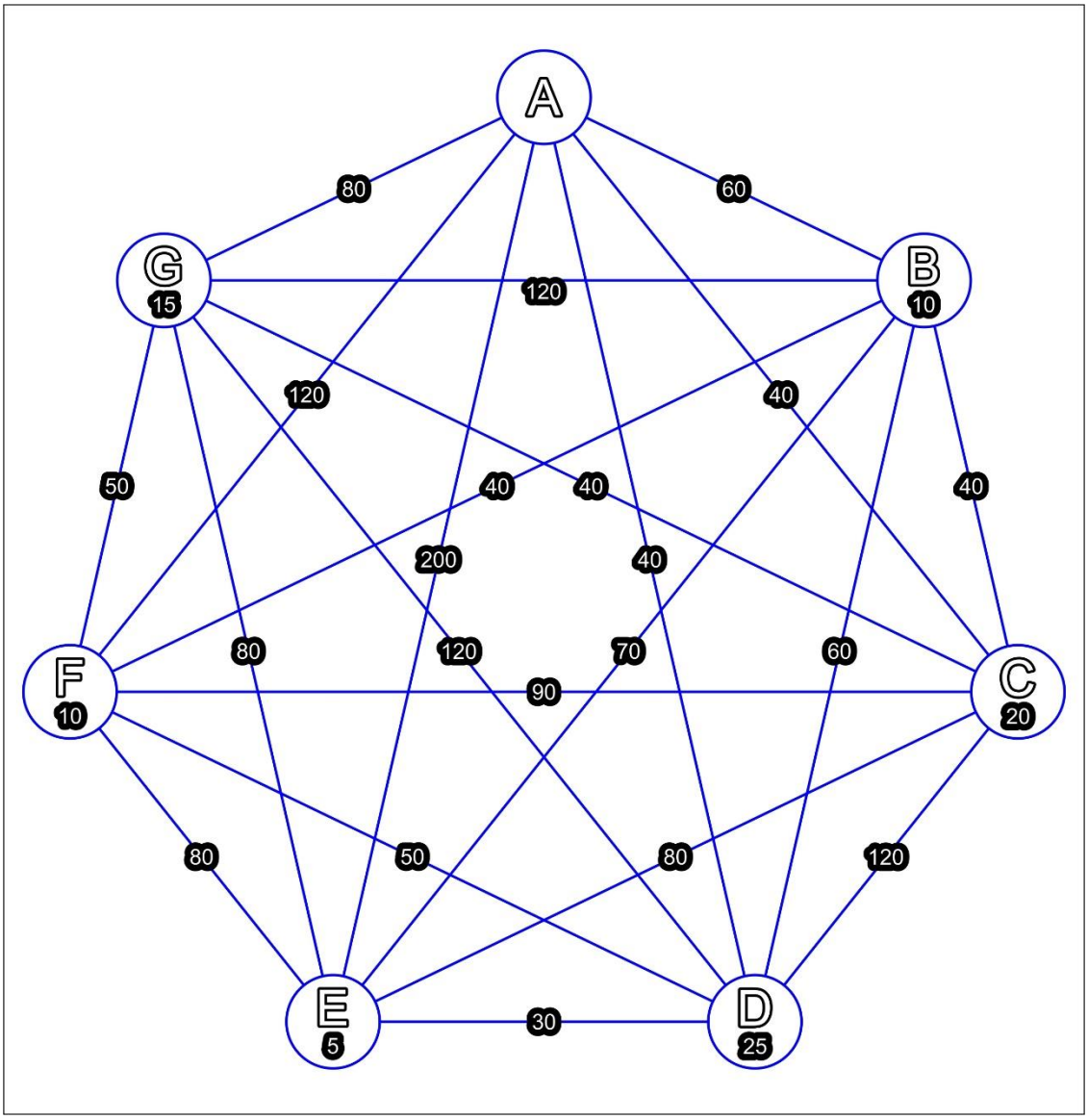
S1 TEKNIK INFORMATIKA 2016

OFFERING B

UNIVERSITAS NEGERI MALANG
FAKULTAS TEKNIK
JURUSAN TEKNIK ELEKTRO
NOVEMBER 2018

Traveling Salesman Problem (TSP) adalah permasalahan yang sudah cukup tua di dunia optimasi. Pada kasus ini seorang dokter ditugaskan untuk memberikan obat kepada masing-masing lokasi grouppies tersebut. Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Persoalan optimasi (optimization problems): persoalan yang menuntut pencarian solusi optimum. Persoalan optimasi ada dua macam: Maksimasi (maximization) dan Minimasi (minimization). Dalam kasus ini adalah minimasi.

Karena ada waktu tempuh dan waktu penyembuhan maka masalah tersebut bisa direpresentasikan sebagai Weighted Graph dan karena bobot perjalanan pulang dan pergi dari suatu kota sama maka menggunakan undirected graph



Gambar 1 Graph Waktu Tempuh dan Waktu Penyembuhan

Matriks Waktu Tempuh Antar Lokasi

	A	B	C	D	E	F	G
A	0	60	40	40	200	120	80
B	60	0	40	60	70	40	120
C	40	40	0	120	80	90	40
D	40	60	120	0	30	50	120
E	200	70	80	30	0	80	80
F	120	40	90	50	80	0	50
G	80	120	40	120	80	50	0

Tabel 1 Matriks Waktu Tempuh

No.	Lokasi	Groupies	Waktu Penyembuhan(menit)
1	B	2	10
2	C	4	20
3	D	5	25
4	E	1	5
5	F	2	10
6	G	3	15

Tabel 2- Tabel Waktu Penyembuhan

Penyelesaian Algoritma Greedy

Algoritma Greedy adalah algoritma yang memecahkan masalah langkah per langkah; pada setiap langkah:

- mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "take what you can get now!") maka pada setiap langkah akan mengambil waktu tempuh dan penyembuhan paling cepat/sedikit.
- berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global

Kota = Waktu Tempuh + Waktu Penyembuhan

1. B = 60 + 10 |C = 40 + 20 |D = 40 + 25 |E = 200 + 5 |F = 120 + 10 |G = 80 + 15 |
   B = 70 |C = 60 |D = 65 |E = 205 |F = 130 |G = 95 |
   Pilih C

2. B = 40 + 10 |D = 120 + 25 |E = 80 + 5 |F = 90 + 10 |G = 40 + 15 |
   B = 50 |D = 145 |E = 85 |F = 100 |G = 55 |
   Pilih B

3. D = 60 + 25 |E = 70 + 5 |F = 40 + 10 |G = 120 + 15 |
   D = 85 |E = 75 |F = 50 |G = 135 |
   Pilih F

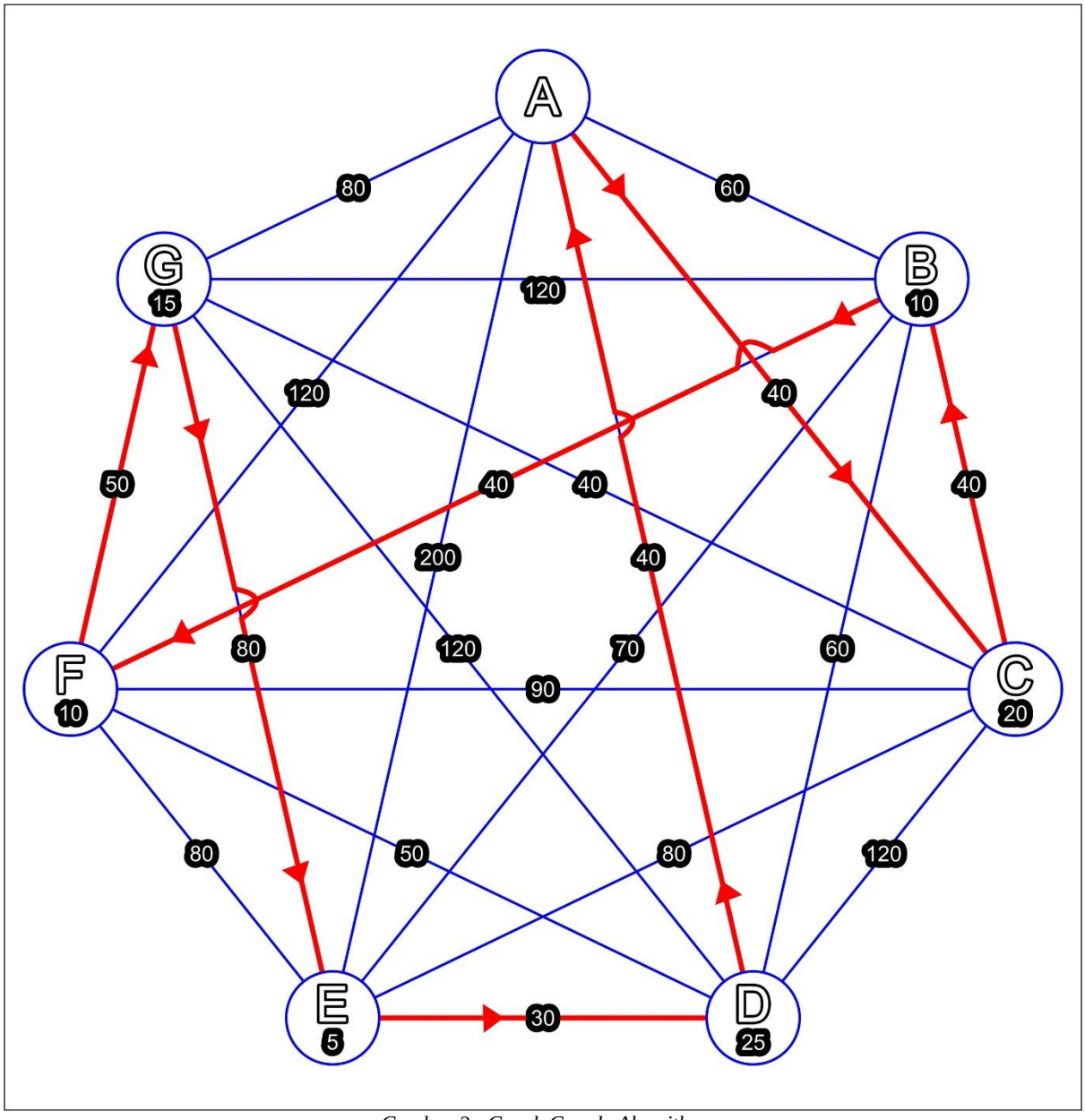
4. D = 50 + 25 |E = 80 + 5 |G = 50 + 15 |
   D = 75 |E = 85 |G = 65 |
   Pilih G

5. D = 120 + 25 |E = 80 + 5 |
   D = 145 |E = 85 |
   Pilih E

6. D = 30 + 25 |
   D = 55 |
   Pilih D

7. Kembali ke A

Maka Rute dari algoritma greedy adalah ['A', 'C', 'B', 'F', 'G', 'E', 'D','A'] Waktu Total : 405



Gambar 2 - Graph Greedy Algorithm

Pseudocode Algoritma Greedy

```
1.  INPUT start, kota, groupy
2.  Rute = []
3.  Waktutempuh = 0
4.  Kotasekarang = start
5.  Tambahkan nilai start pada rute
6.  FOR i=1 to jumlah kota +1
7.          tempKota.nama = empty
8.          tempKota.waktu = empty
9.          FOREACH tetangga in Kotasekarang
10.                 IF tetangga not Kotasekarang and tetangga not in rute
11.                 and jumlahKota not equal jumlahRute
12.                         IF tempKota.waktu not empty
13.                                 IF tempKota.waktu greater than
14.                                 (tetangga.waktupenyembuhan + tetangga.waktuperjalanan)
15.                                         tempKota.waktu = (tetangga.waktupenyembuhan +
16.                                                                 tetangga.waktuperjalanan)
17.                                         tempKota.nama = tetangga.nama
18.                                 ENDIF
19.                         ELSE
20.                                 tempKota.waktu = (tetangga.waktupenyembuhan +
21.                                                         tetangga.waktuperjalanan)
22.                                 tempKota.nama = tetangga.nama
23.                         ENDIF
24.                 ENDIF
25.         END
26.         IF jumlahKota equal jumlahRute
27.                 tempKota.waktu = kotaSekarang[start].waktuperjalanan
28.                 tempKota.nama = start
29.         ENDIF
30.         Tambahkan tempKota pada rute
31.         Kotasekarang = tempKota.nama
32.         Waktutempuh = Waktutempuh + tempKota.waktu
33. END
```

Baris 1 – 5 merupakan inisialisasi

Baris 6 – 33 merupakan proses perulangan untuk menambahkan rute sebanyak jumlah kota

Baris 7 – 25 merupakan proses mencari nilai terkecil dengan membandingkan masing masing nilai dari waktu perjalanan ditambah waktu penyembuhan menggunakan sequential search

Baris 26 – 29 merupakan proses penambahan rute kembali dari kota terakhir ke kota awal

Dapat dilihat pada pseudocode mengandung looping sejumlah n di dalam looping sejumlah n maka notasi Big-O adalah (N^2).

Penyelesaian Bruteforce

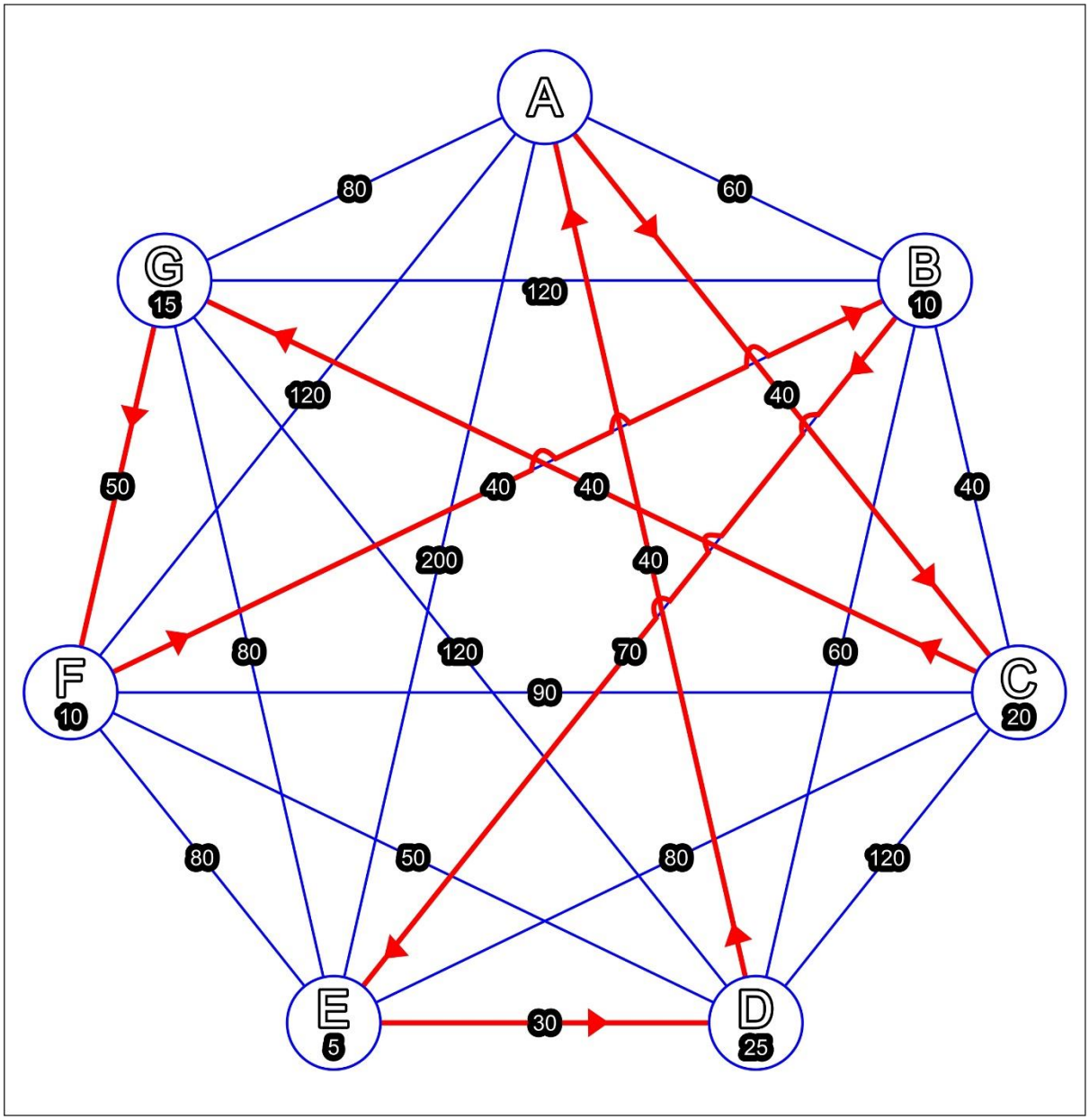
Probabilitas Bruteforce yaitu Permutasi dari N={A,B,C,D,E,F,G}=7 karena A adalah lokasi/awal dan akhir yang pasti maka menjadi 6 faktorial maka terdapat 720 kemungkinan yang dihasilkan dari bruteforce

A	X	X	X	X	X	X	A

$$(7-1)! = 720$$

Dari 720 kemungkinan muncul Solusi Teratas dari Bruteforce yaitu:

['A', 'C', 'G', 'F', 'B', 'E', 'D', 'A'] Waktu Total : 395



Gambar 4 Graph Bruteforce Algorithm

Solusi Teratas Lainnya

 # 1 : ['A', 'C', 'G', 'F', 'B', 'E', 'D', 'A'] Waktu Total : 395
2 : ['A', 'D', 'E', 'B', 'F', 'G', 'C', 'A'] Waktu Total : 395
3 : ['A', 'C', 'B', 'F', 'G', 'E', 'D', 'A'] Waktu Total : 405
4 : ['A', 'D', 'E', 'G', 'F', 'B', 'C', 'A'] Waktu Total : 405
5 : ['A', 'B', 'C', 'G', 'F', 'E', 'D', 'A'] Waktu Total : 425

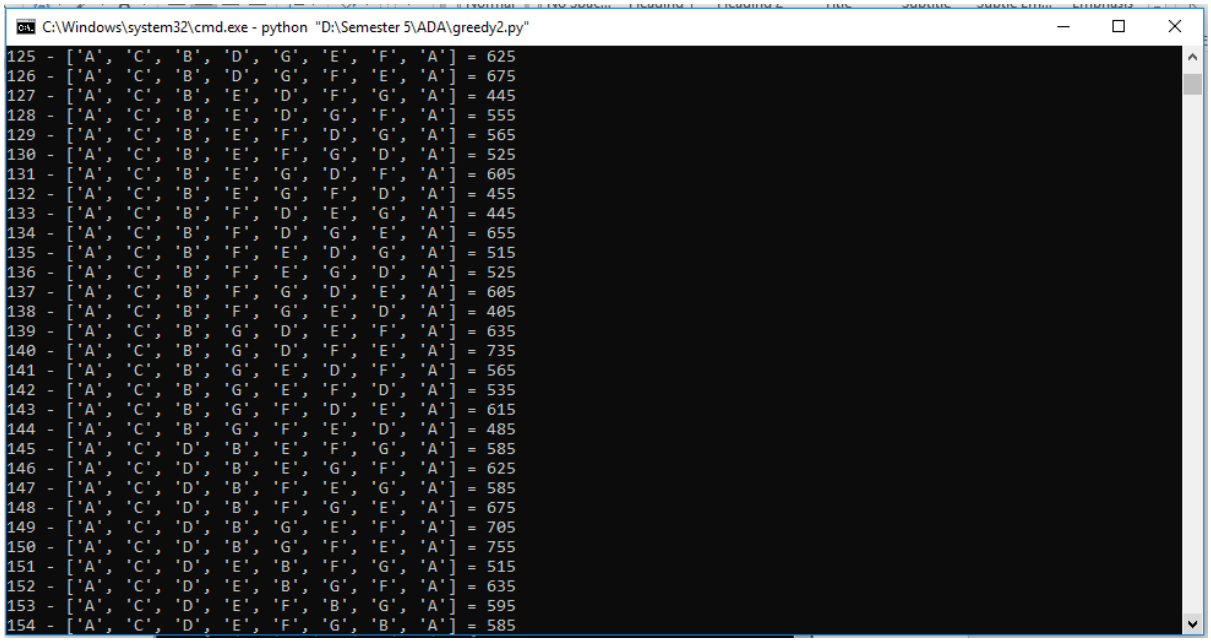
Tabel 3Urutan Solusi Teratas

Kesimpulan

Penggunaan algoritma greedy memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup cepat. Optimum global yang dihasilkan greedy belum tentu merupakan solusi optimum (terbaik), tetapi sub-optimum atau pseudo-optimum.

 # 1 : ['A', 'C', 'G', 'F', 'B', 'E', 'D', 'A'] Waktu Total : 395
2 : ['A', 'D', 'E', 'B', 'F', 'G', 'C', 'A'] Waktu Total : 395
 # 3 : ['A', 'C', 'B', 'F', 'G', 'E', 'D', 'A'] Waktu Total : 405
4 : ['A', 'D', 'E', 'G', 'F', 'B', 'C', 'A'] Waktu Total : 405
5 : ['A', 'B', 'C', 'G', 'F', 'E', 'D', 'A'] Waktu Total : 425

Dapat dilihat bahwa algoritma greedy tidak menghasilkan solusi terbaik , masih ada 2 solusi yang lebih efektif. Pada bruteforce solusi greedy muncul pada kombinasi ke 138 dari 720



Gambar 5 Urutan Solusi Greedy pada Bruteforce

Daftar Rujukan

Github - Simon Westphahl. “TSP brute-force solution”. 2 November 2018. https://gist.github.com/westphahl/432876
Stackoverflow – Albert Rothman. “Basic Greedy Search in Python”. 4 November 2018. https://stackoverflow.com/questions/41749254/basic-greedy-search-in-python