META HEURISTIC OPTIMIZATION
ASSIGNMENT 2

BY Srinivasaraghavan Seshadhri – R00195470
srinivasa.raghavan@mycit.ie

PART – 1:

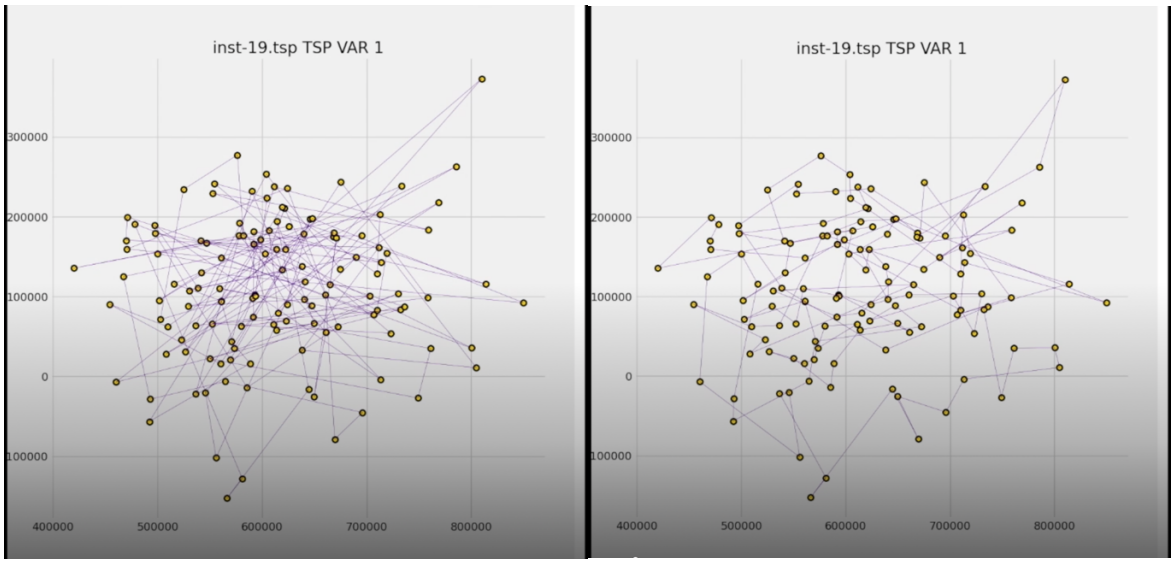
The corresponding file is TSP_A2_R00195470.py to be viewed.

The explanation for the code is given in the py file and henceforth it is not re-explained here to avoid repetition.

There are 3 variants from the question as follows:

VAR 0: Basic 2 opt algorithm
VAR 1: It is similar to N-Queens, in that rather than search all edges we choose an edge at random and we search all possible edges for swapping that edge with.
VAR 2: It is to change the best improvement for the first improvement, as soon as an improving move is found it is made.

The following image is a sample of TSP visualization from the attached TSP_with_graphics.ipynb which can run live visualizations.



The first and the second picture shows before and after solving the TSP. (Please note that the screen capture software used has lagged and therefore the first picture is not exactly the initial tour nor the second picture has the final solution – both have a few frames of difference)

Please watch the attached video - tsp_inst_19_first_variant.mov as a sample for the live graph.

RESULTS:
TIME TAKEN:

	VAR 0	VAR 1	VAR 2
INST 7		448.1093359	496.0838385
INST 19	176.983325	8.892998457	17.41684961
INST 20		62.40729356	87.5508182

TOUR DISTANCE AFTER SOLVING:

	VAR 0	VAR 1	VAR 2
INST 7		88622324.59	153571641.4
INST 19	3793748.43	8604149.199	9384838.207
INST 20		45201064.17	60522338.33

Please note that the VAR 0 has been running for too long and hence the results are not generated yet. Kindly check the comment section of the PDF submission for the missing results.

CONCLUSIONS:

Variant 0 - The basic 2 opt Algorithm:
It is very slow compared to the other 2 variants.
It gives the BEST solution with minimum tour distance.

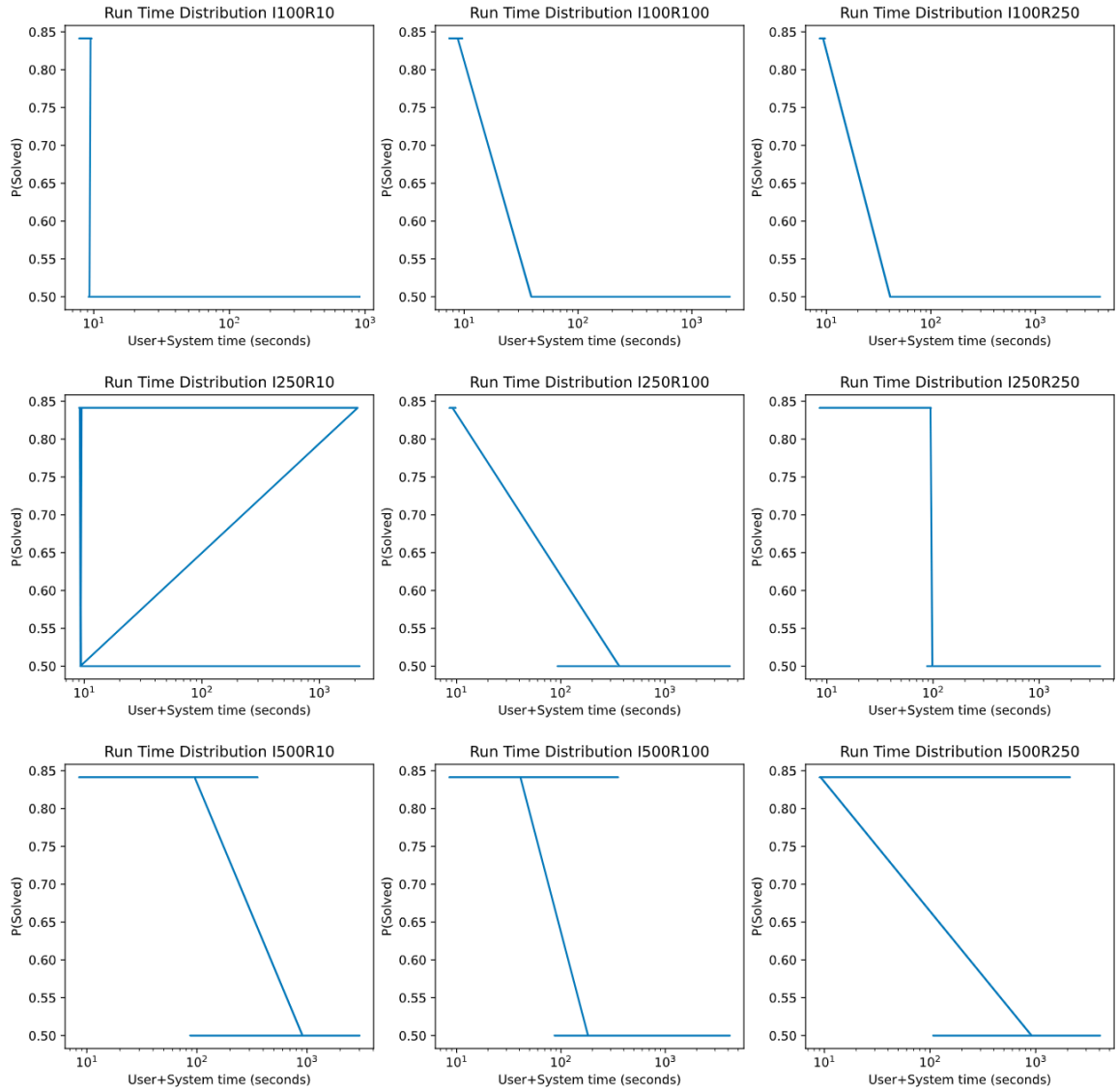
Variant 1:
It is fastest of the 3 variants.
It gives better results in terms of tour distance when compared to VAR 2, but much worse than VAR 0. It is good for optimizing to a level but with very low temporal cost.

Variant 2:
It is quite faster when compared to the basic 2 opt algorithm, but slightly slower than VAR 1.
It is slower and has the worse results in terms of tour distance.

PART-2:

RUN TIME DISTRIBUTIONS:

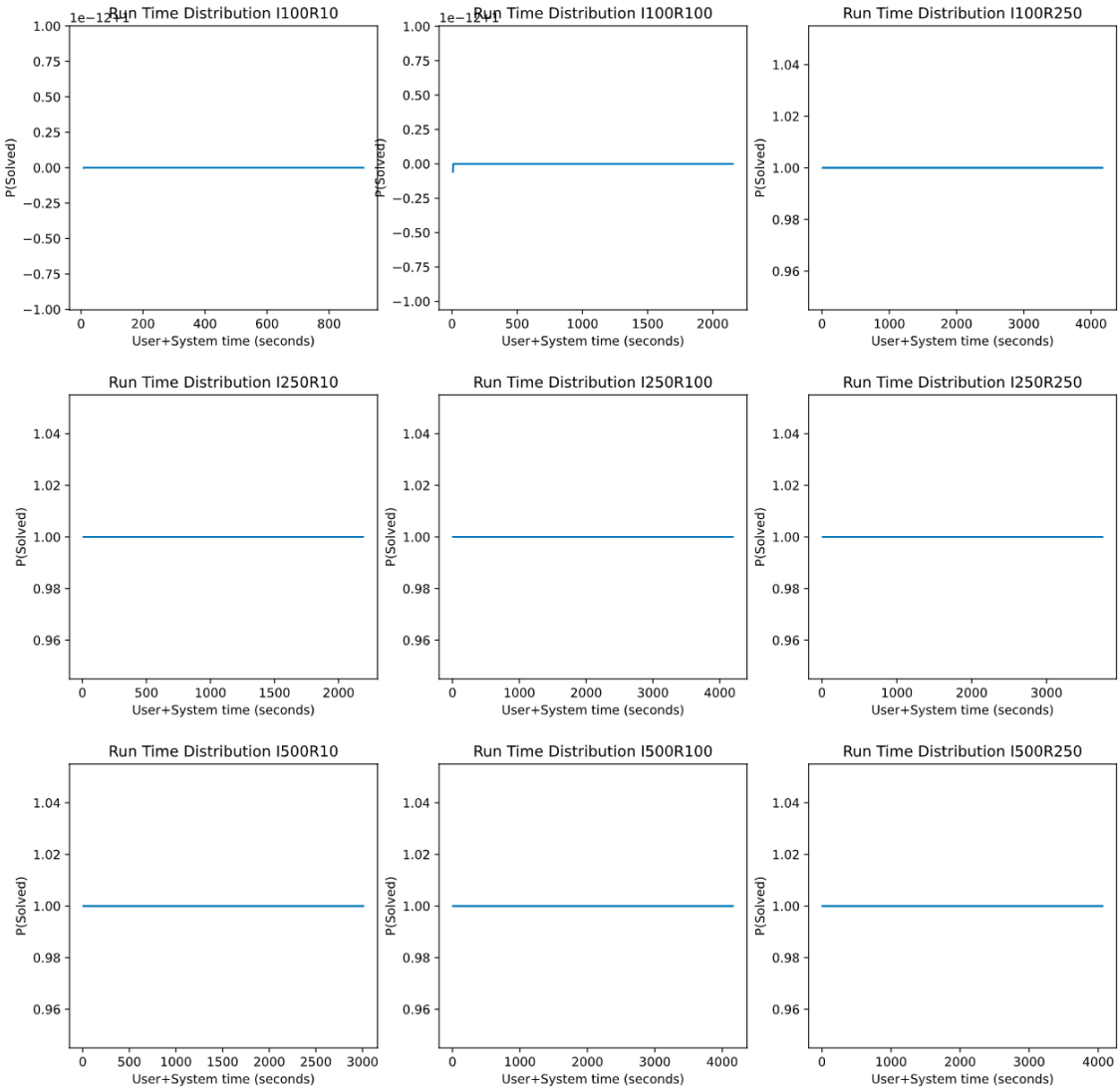
The Hillclimbing.py files were made into 2, where one allowed the side-to-side movements where the other one wouldn't. The RTD_Q_launcher.py launches and runs both variants with different iterations and restart values every 100 times and saves the results into output_pooled_1.txt. There was manual cleaning of the data format involved to gather results. The result_gen.ipynb was used for the data extraction, analysis and visualization purposes.



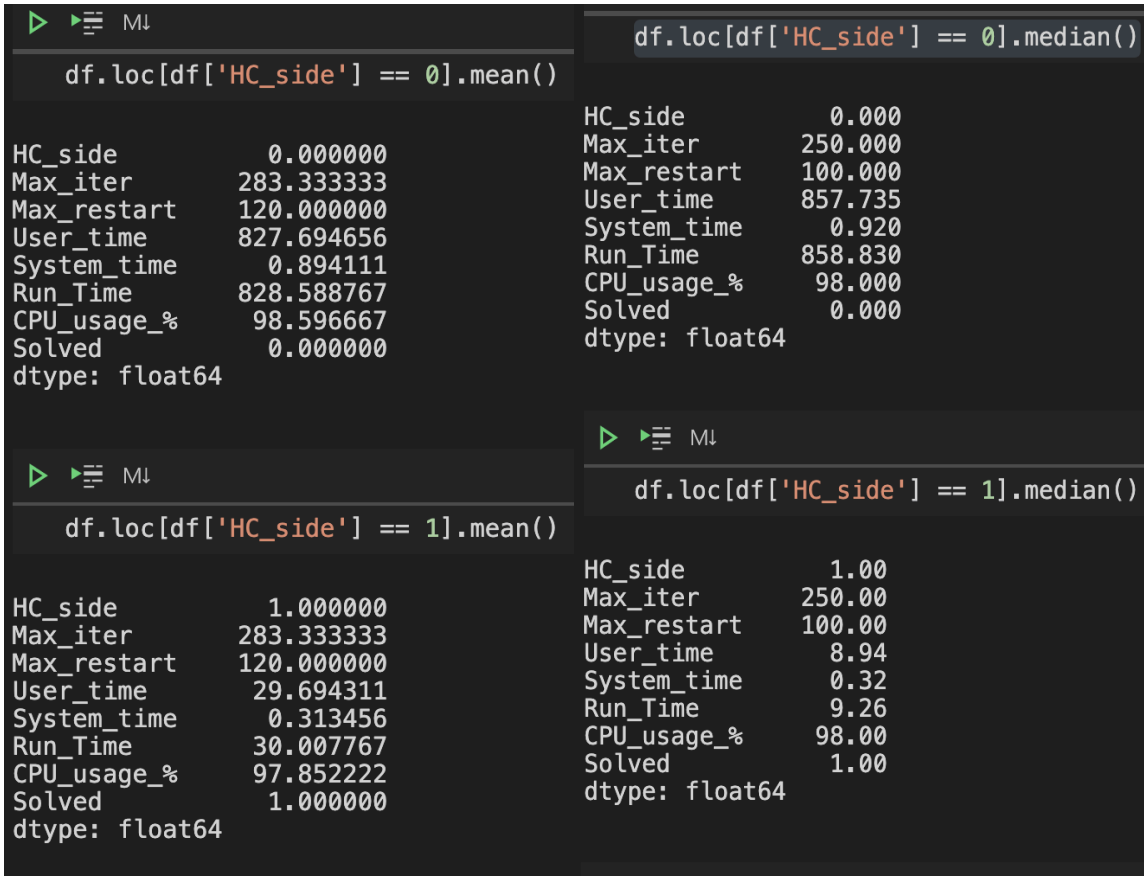
The above graphs contain the CDF of whether the problem was solved or not vs the runtime in seconds.

From the graphs and the processed results data stored in RT_results_data.xlsx, It was observed that the variant in which side to side movement was not allowed took significantly more time but it didn't successfully solve the problem. On the other hand, the variant with the side movements allowed solved it very quickly.

The below graph shows CDF of run times vs run times which show that the runtimes were constant.



The following piece of analysis shows the statistics of when side movement wasn't allowed vs when the side movement was allowed, which supports the above statements.



```
df.loc[df['HC_side'] == 0].mean()

HC_side          0.000000
Max_iter       283.333333
Max_restart    120.000000
User_time      827.694656
System_time      0.894111
Run_Time       828.588767
CPU_usage_%     98.596667
Solved           0.000000
dtype: float64
```

```
df.loc[df['HC_side'] == 0].median()

HC_side          0.000
Max_iter       250.000
Max_restart    100.000
User_time      857.735
System_time      0.920
Run_Time       858.830
CPU_usage_%     98.000
Solved           0.000
dtype: float64
```

```
df.loc[df['HC_side'] == 1].mean()

HC_side          1.000000
Max_iter       283.333333
Max_restart    120.000000
User_time       29.694311
System_time      0.313456
Run_Time        30.007767
CPU_usage_%     97.852222
Solved           1.000000
dtype: float64
```

```
df.loc[df['HC_side'] == 1].median()

HC_side          1.00
Max_iter       250.00
Max_restart    100.00
User_time        8.94
System_time      0.32
Run_Time         9.26
CPU_usage_%     98.00
Solved           1.00
dtype: float64
```

System Configuration:

CODE used for the same:

```
import platform, subprocess

def get_processor_info():
    if platform.system() == "Windows":
        return platform.processor()
    elif platform.system() == "Darwin":
        return subprocess.check_output(['/usr/sbin/sysctl', "-n",
"machdep.cpu.brand_string"]).strip()
    elif platform.system() == "Linux":
        command = "cat /proc/cpuinfo"
        return subprocess.check_output(command, shell=True).strip()
    return ""
print(get_processor_info().strip().decode())
```

The below is a snippet of one of the hyperthreaded cores.

```
processor       : 11
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
stepping        : 10
microcode       : 0xde
cpu MHz         : 3440.362
cache size      : 9216 KB
physical id     : 0
siblings        : 12
core id         : 5
cpu cores       : 6
apicid          : 11
initial apicid  : 11
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse ss
e2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cp
uid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic mov
be popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd i
brs ibpb stibp tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdsee
d adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_ep
p md_clear flush_l1d
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs itlb_multihit srbds
bogomips        : 4399.99
clflush size    : 64
cache_alignment : 64
address sizes   : 39 bits physical, 48 bits virtual
power management:
```