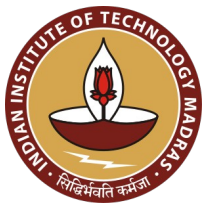
Effective Parallelization of the Vehicle Routing Problem

Rajesh Pandian M[†], Somesh Singh[*], Rupesh Nasre[†] & N.S.Narayanaswamy[†]

[†]Indian Institute of Technology Madras, India.
[*]CNRS and LIP, France.



2023

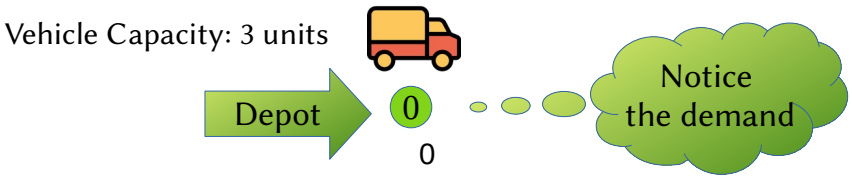
Capacitated Vehicle Routing Problem (CVRP)

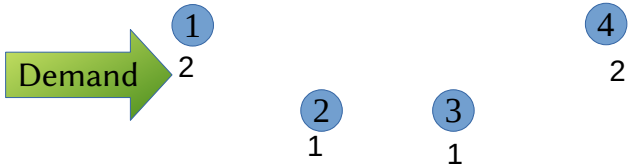
Input : Given *n* nodes (single Depot and customers) with their coordinates (x_i, y_i) and demands $d_i > 0$ for $i \in n$, Vehicle capacity *C*. Node 0 is Depot and has zero demand.

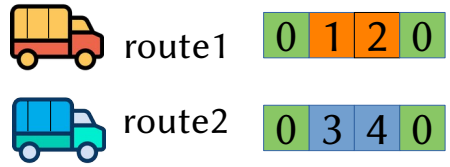
Output: Set of routes serving all the customers respecting the vehicle capacity from/to Depot.

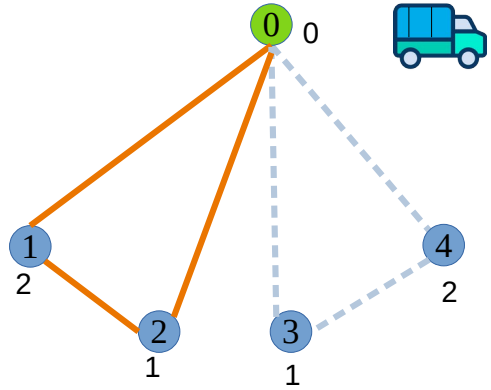
Goal : Minimize total distance travelled.

If Capacity = sum of demands d_i ,
CVRP ➜ Travelling Salesman Problem









NP-Hard

Motivation

Current state-of-the-art
- work only on smaller instances
- has a large solution Gap.
- takes a lot of time.

Instance	Number of customers	Time (s)	
		Base2	Base1
Flanders2	30,000	8,355	2,534
Flanders1	20,000	7,768	2,031
Brussels1	15,000	7,164	871

Table 1: State-of-the-art GPU methods are time-consuming.

RQ1. Can we invent a simpler algorithm?

RQ2. Can we reduce Gap on large instances?

RQ3. Design Parallelization friendly algorithms?

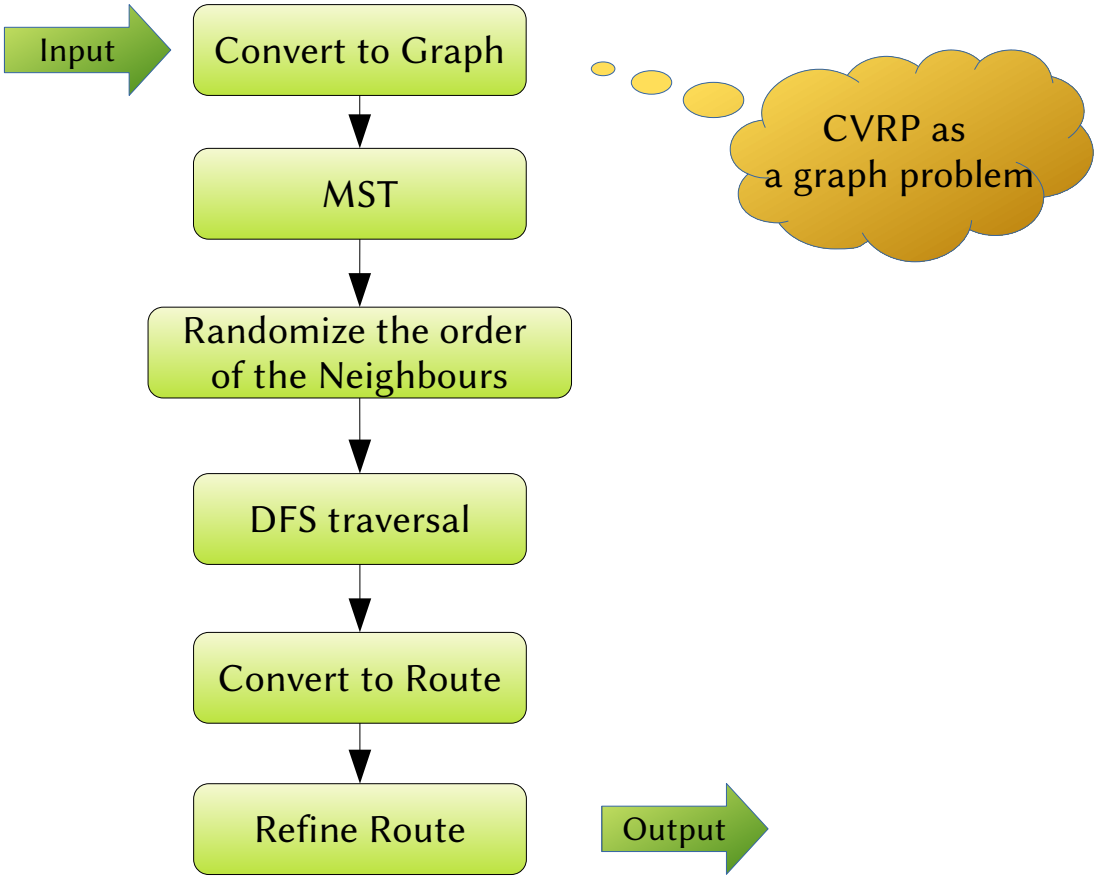
Our **ParMDS**
- Serial and Parallel implementation
- Combining MST and DFS
- Uses Local-search approach
- Uses Randomization approach

$$Gap = \frac{Z_S - Z_{BKS}}{Z_{BKS}} \times 100$$

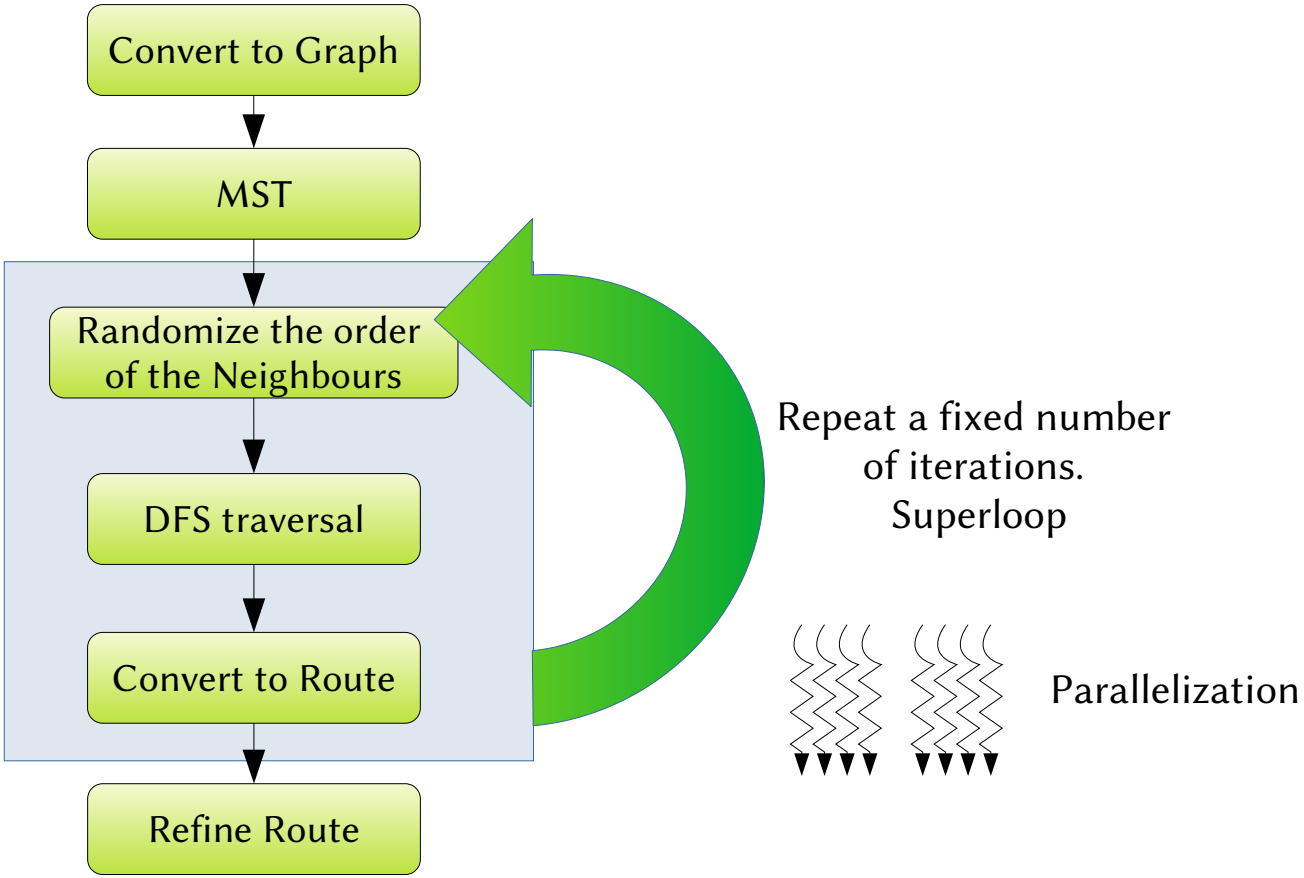
Baseline1: P. Yelmewad and B. Talawar. Parallel Version of Local Search Heuristic Algorithm to Solve Capacitated Vehicle Routing Problem, Cluster Computing, 2021.

Baseline2: M. Abdelatti and M. Sodhi. An improved GPU-accelerated heuristic technique applied to the Capacitated Vehicle Routing Problem, GECCO, 2020.

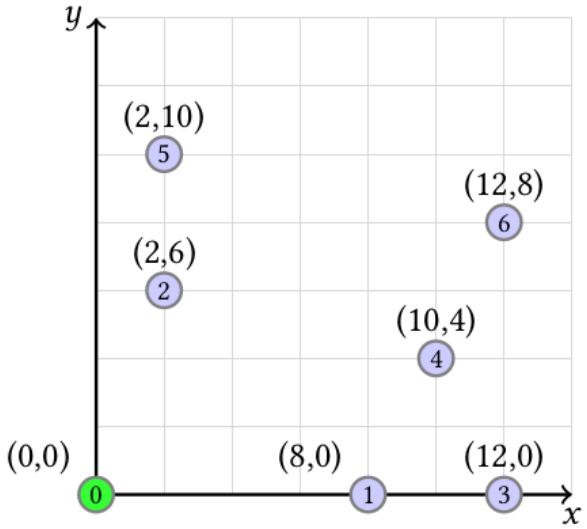
Overview - ParMDS



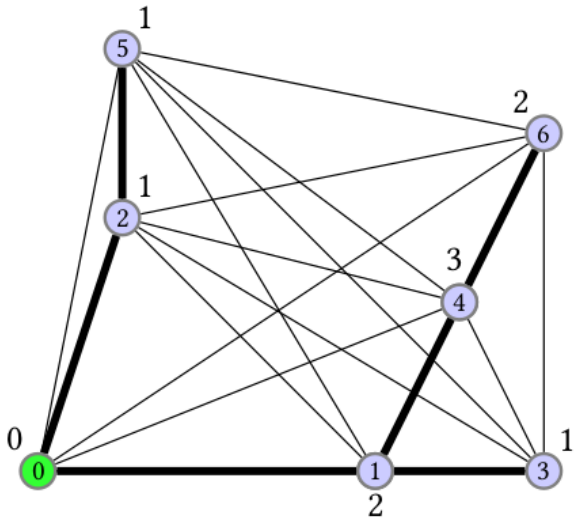
Overview - ParMDS



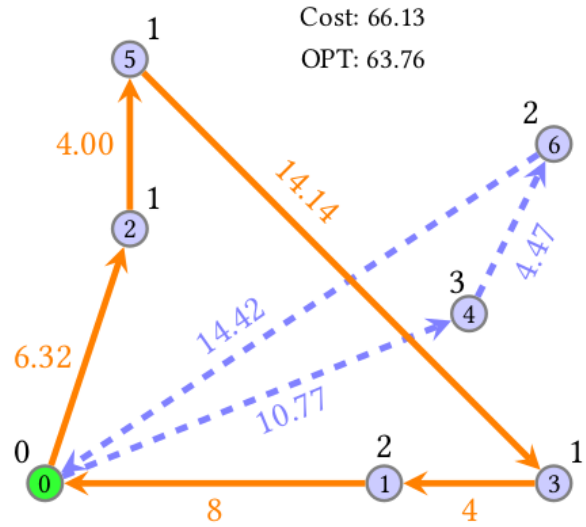
Example - Overview



(a) Input instance I



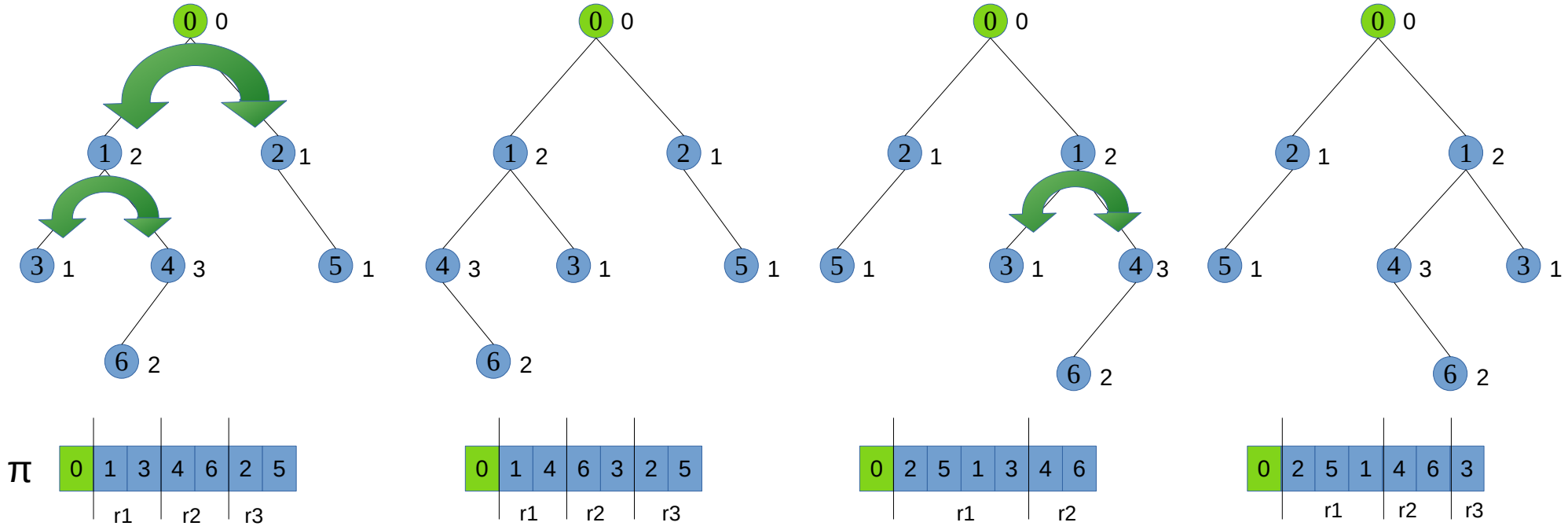
(b) Graph for I, along with node-demands



(c) Final routes generated by ParMDS

ParMDS on an example input instance with n = 7 and Vehicle Capacity = 5.

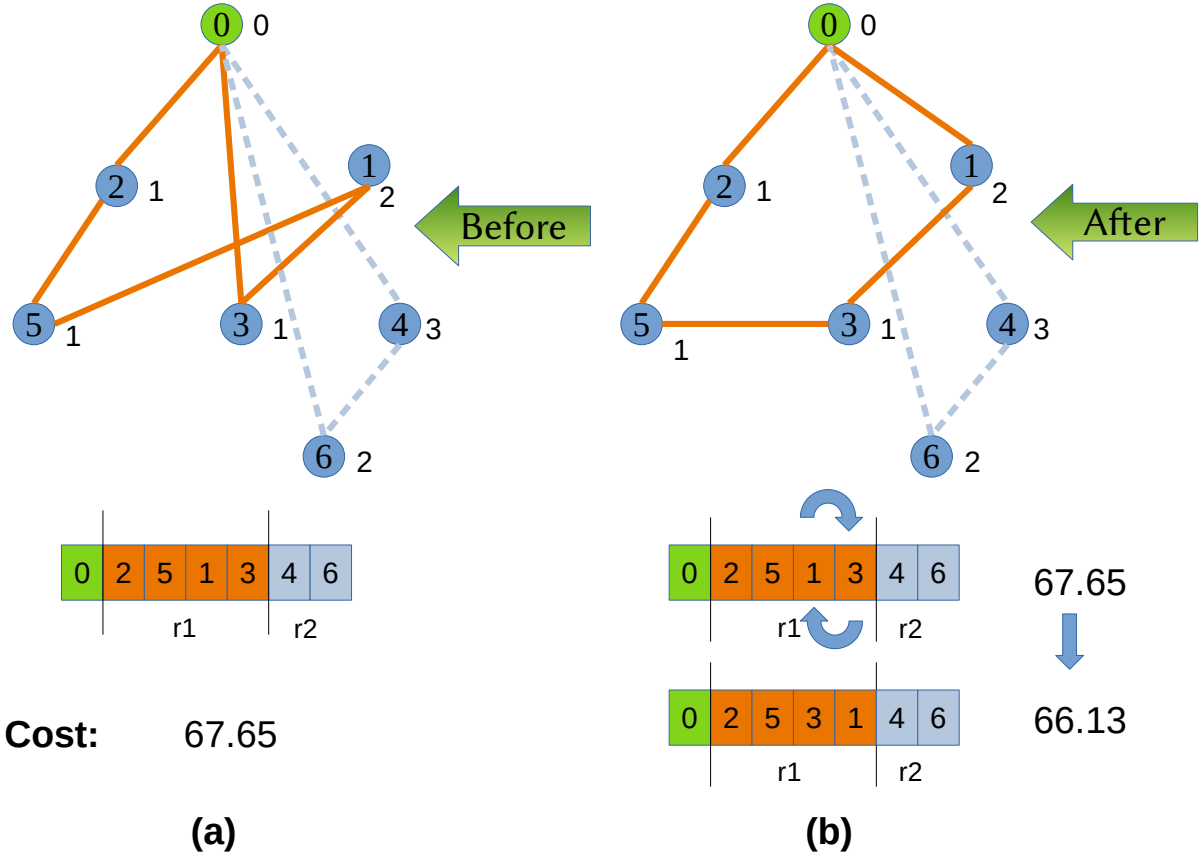
Example – DFS and Randomization



Cost: 74.19 (a) 78.19 (b) 67.65 (c) 83.65 (d)

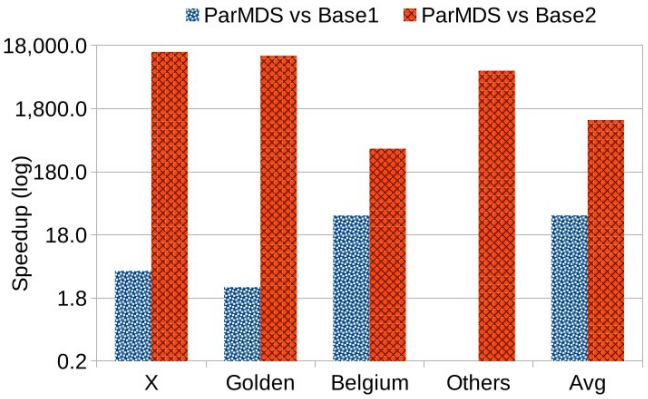
Takeaway: Randomizing neighbours of MST may yield a different DFS ordering. Hence, a different route!

Intra-route optimization - 2Opt



Experiments

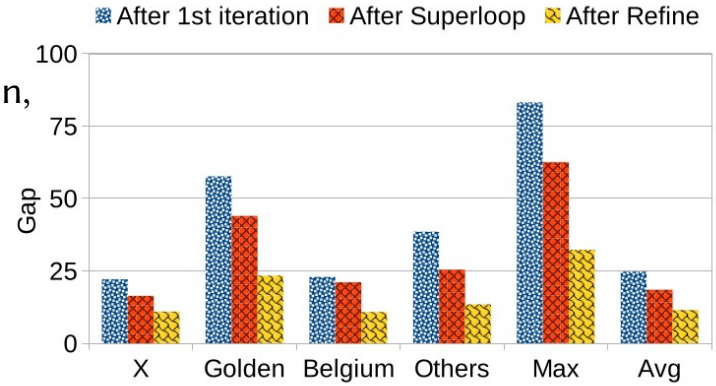
- 130 Instances of CVRPLIB
  - X
  - Golden
  - Belgium
  - Others
- Intel Xeon CPU E5-2640 v4
  - 40 cores
  - Clock 2.4 Ghz
  - RAM 64 GB
- Baselines on GPU
  - NVIDIA's Tesla P100
  - 3584 cores & 12GB global memory
  - CUDA 11.5
- Our Code uses
  - **SeqMDS:** GCC 9.3.1
  - **ParMDS:** nvc++ compiler NVIDIA's HPC SDK 22.11



Method	Execution Time (s) using Random
SeqMDS	1,722.44
ParMDS-Standard	1,522.26
ParMDS-Strided	186.50

Check out our paper for more detailed analysis

Gap at the end of : 1st iteration, Superloop and Refine step



Summary

- GPU parallelization has limitations on larger instances
  - Takes longer time
  - Solution Gap is large
- Our technique combines simpler algorithms/techniques
  - MST and DFS
  - Uses randomization
  - Uses parallelization
  - Open source code
- Our parallelization technique can be extended to other iterative local-search / genetic algorithms

https://github.com/mrprajesh/parMDS

Future Directions

ParMDS can be extended

- to use OpenACC for running on GPUs
- to incorporate direction-aware local-search
- to integrate inter-route optimizations

Thank you

Looking for Postdoc