

Borah Routing

Implementing Borah, Owens and Irwin routing algorithm

Guillaume Guérin

Anthony Prudhomme

```
build.js
package.json
README.md
runner.js
server.js
stats.js
```

```
function getMergeBaseFromLocalGitRepo(localRepo) {
  let repo = await Git.Repository.open(localRepo);
  return await Git.Merge.base(
    repo,
    await repo.getHeadCommit(),
    await repo.getBranchCommit('master')
  );
};

async function buildBenchmarkBundlesFromGitRepo(
  commitId,
  skipBuild,
  url = reactUrl,
  clean
) {
  let repo;
  let repoDir = getDefaultReactPath();
  if (!skipBuild) {
    if (clean) {
      //clear remote-repo folder
      await cleanDir(remoteRepoDir);
    }
    if (remote-repo directory already exists) {
      await open(remoteRepoDir);
    }
  }
}
```

Results

- **Red**nodes are the original nodes from the file.
- **Green**nodes are the added Steiner points.
- **Blue**nodes are nodes that were added for rectilinearization.

5

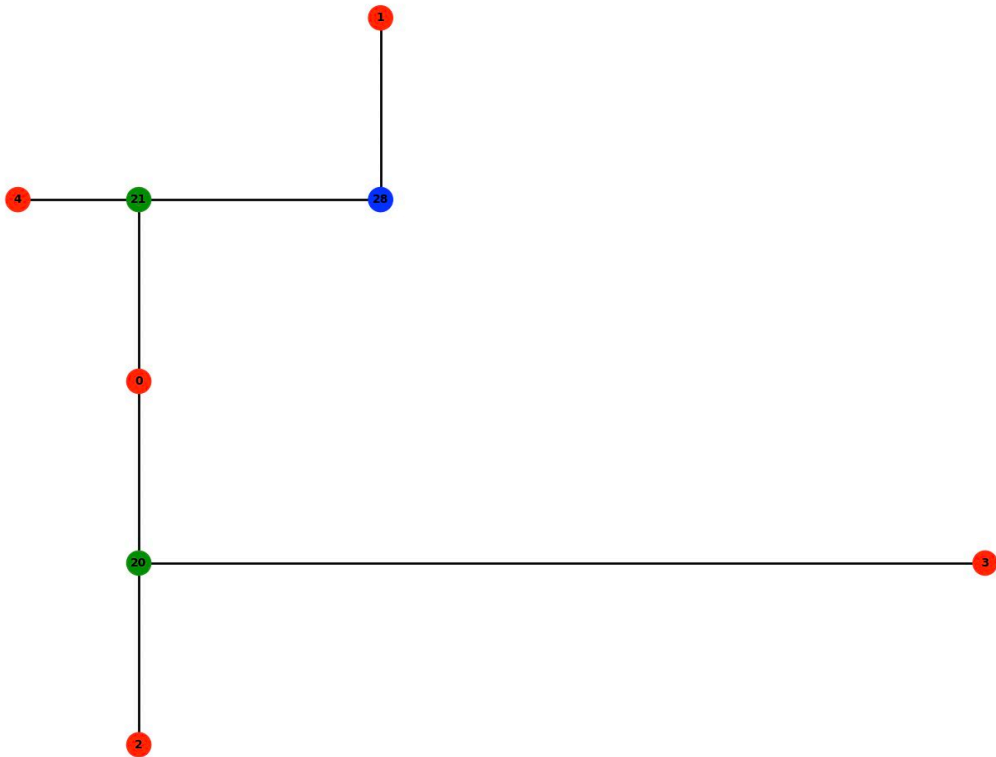
10

20

50

100

5 points



- 5 points
- 2 Steiner points
- MST WL – 21
- BorahWL – 18
- 1 pass
- Prim Runtime – 1ms
- Kruskal Runtime – 1ms
- Borah Runtime – 1ms

5

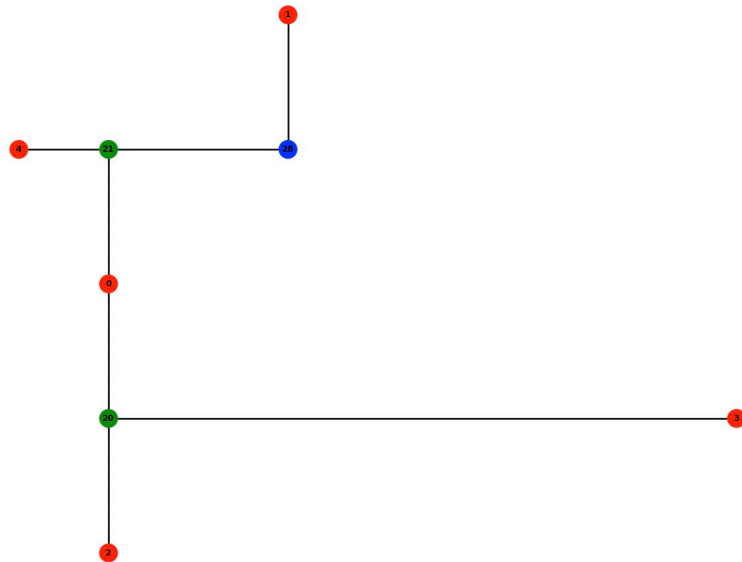
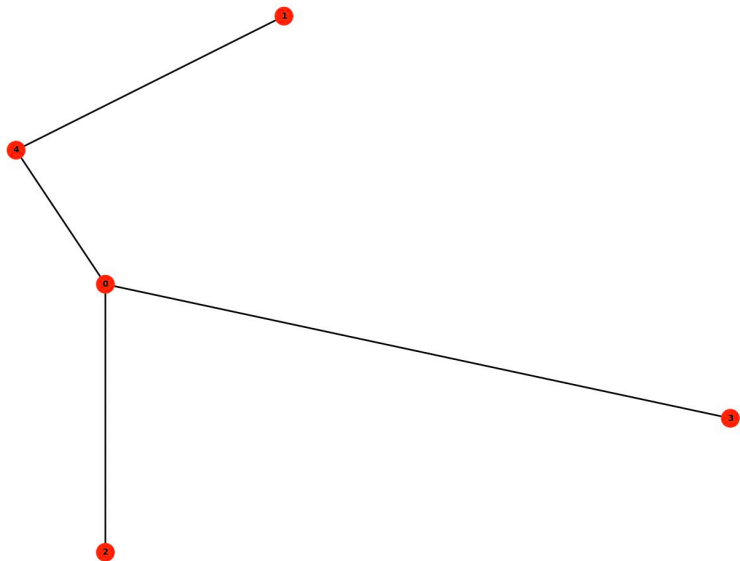
10

20

50

100

5 points



5

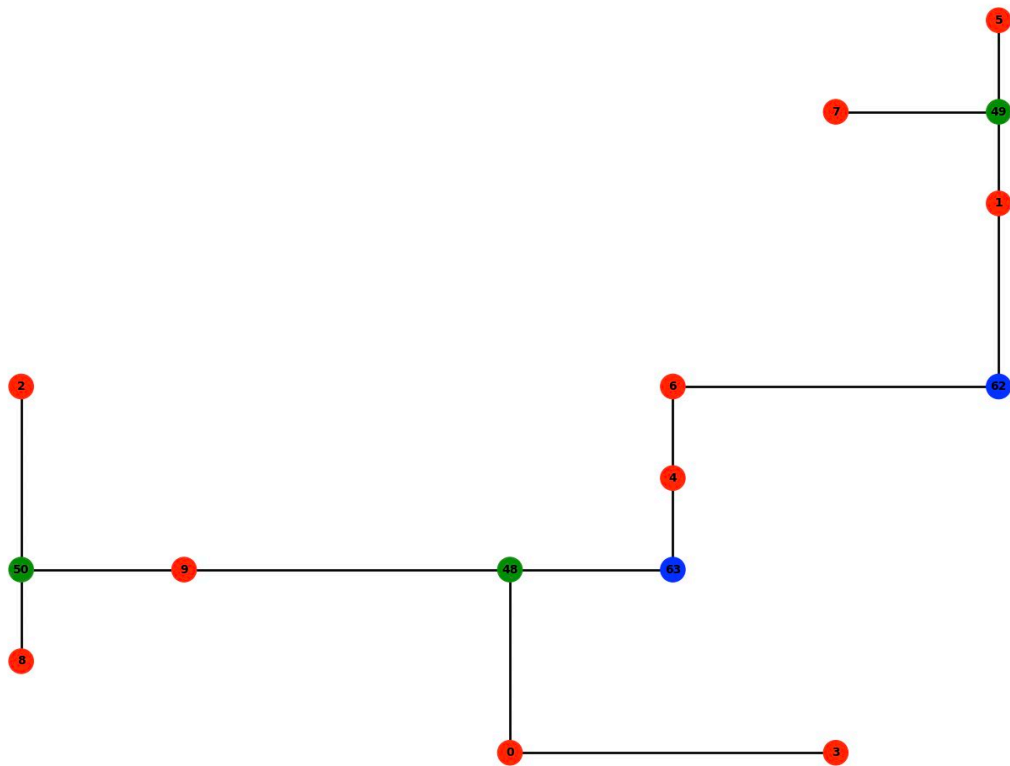
10

20

50

100

10 points



- 10 points
- 3 Steiner points
- MST WL= 24
- BorahWL= 20
- 1 pass
- Prim Runtime= 1ms
- KruskalRuntime= 1ms
- BorahRuntime= 1ms

5

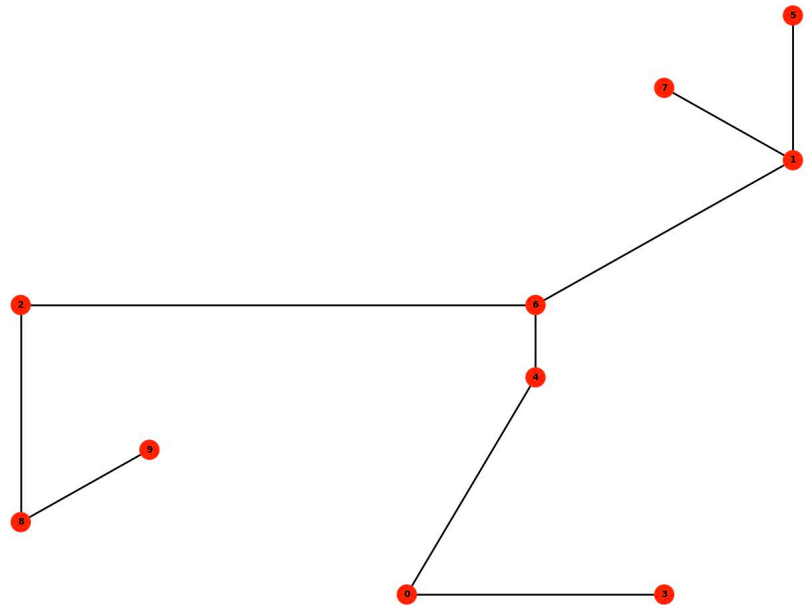
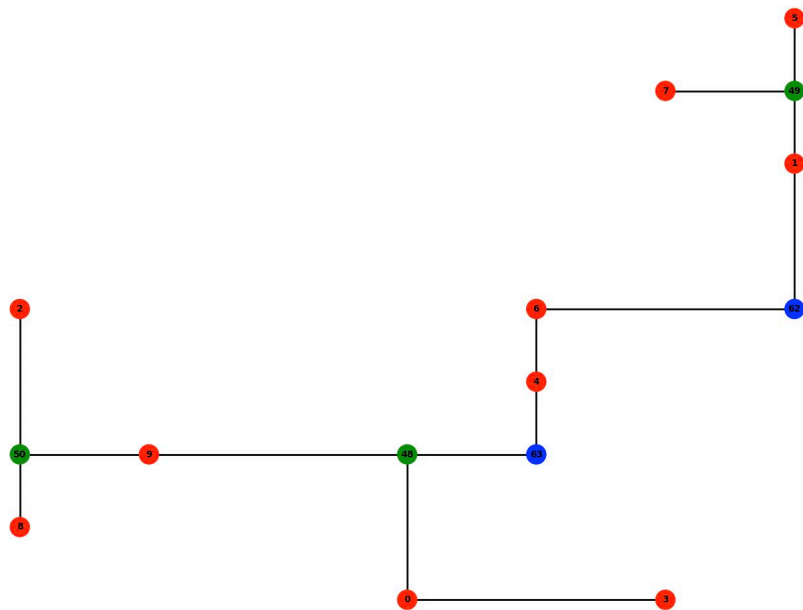
10

20

50

100

10 points



5

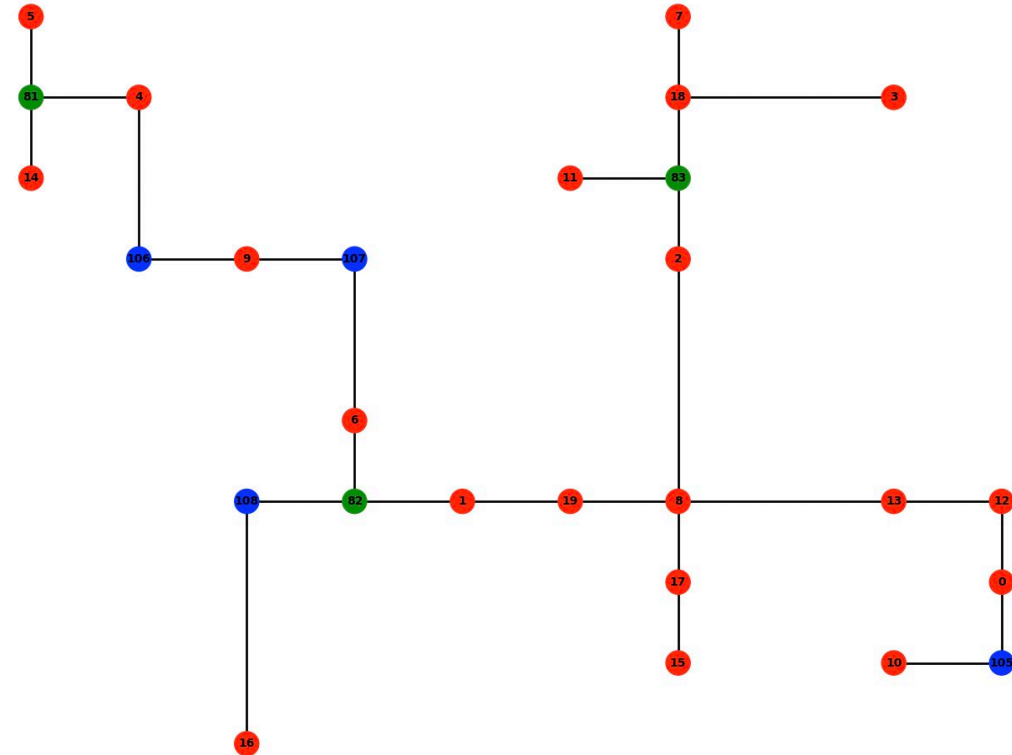
10

20

50

100

20 points



- 20 points
- 3 Steiner points
- MST WL – 37
- BorahWL – 34
- 1 pass
- PrimRuntime – 2ms
- KruskalRuntime – 2ms
- BorahRuntime – 6ms

5

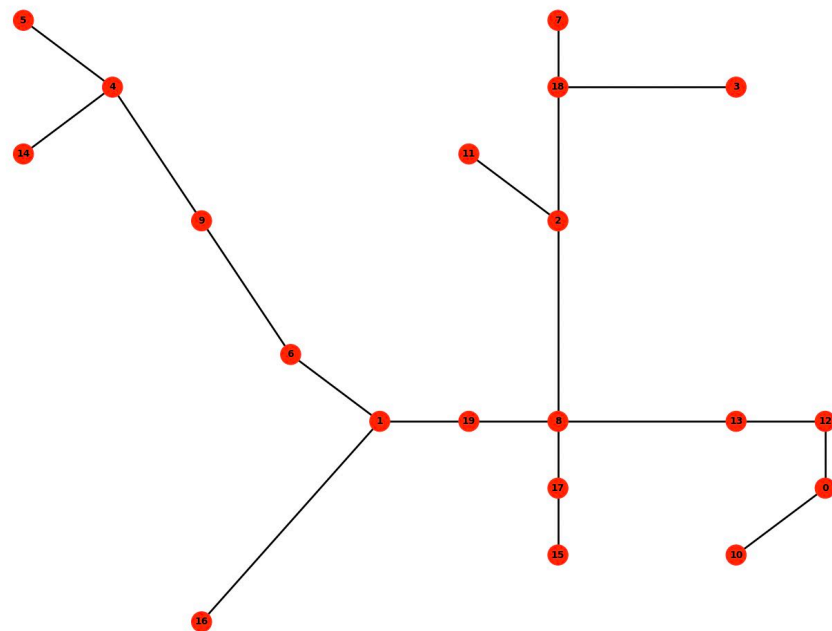
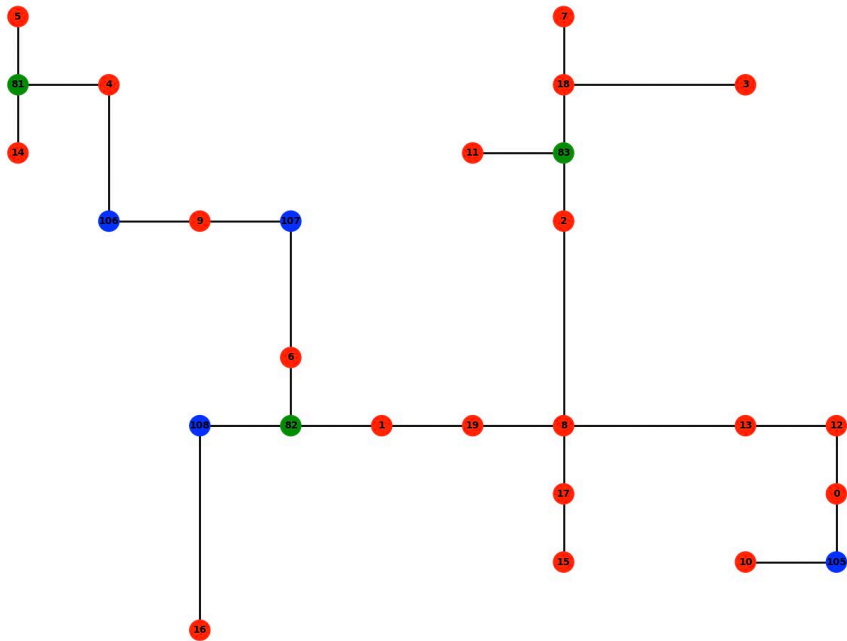
10

20

50

100

20 points



5

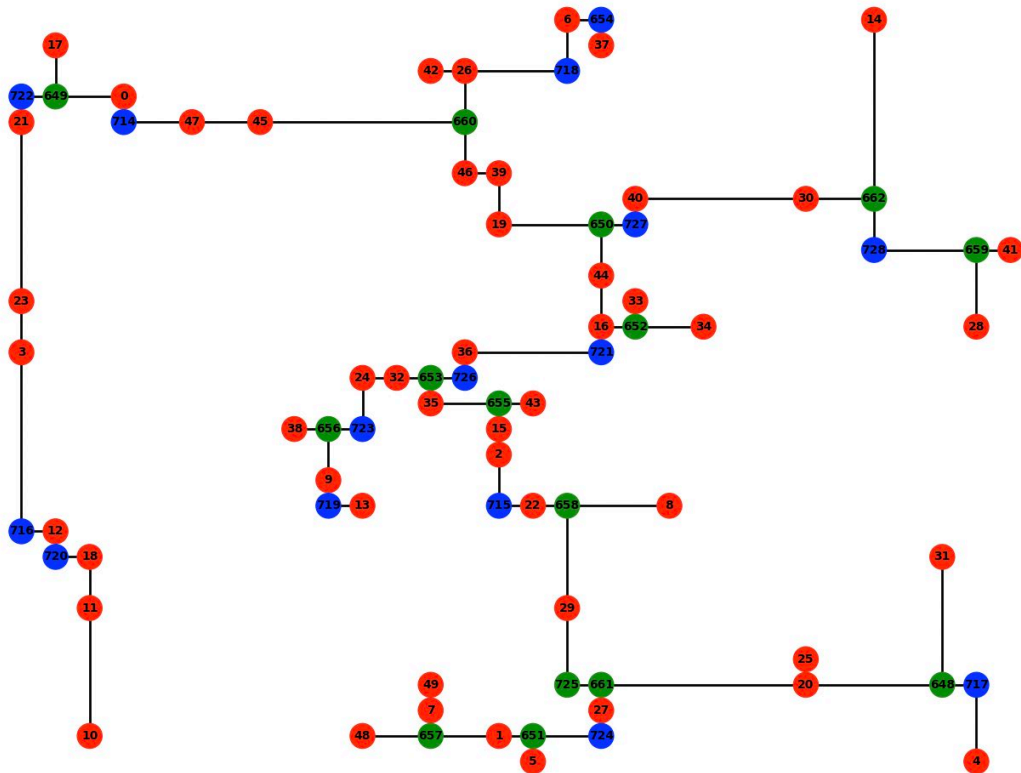
10

20

50

100

50 points



- 50 points
- 16 Steiner points
- MST WL – 183
- BorahWL – 163
- 2 pass
- PrimRuntime – 15ms
- KruskalRuntime – 11ms
- BorahRuntime – 236ms

5

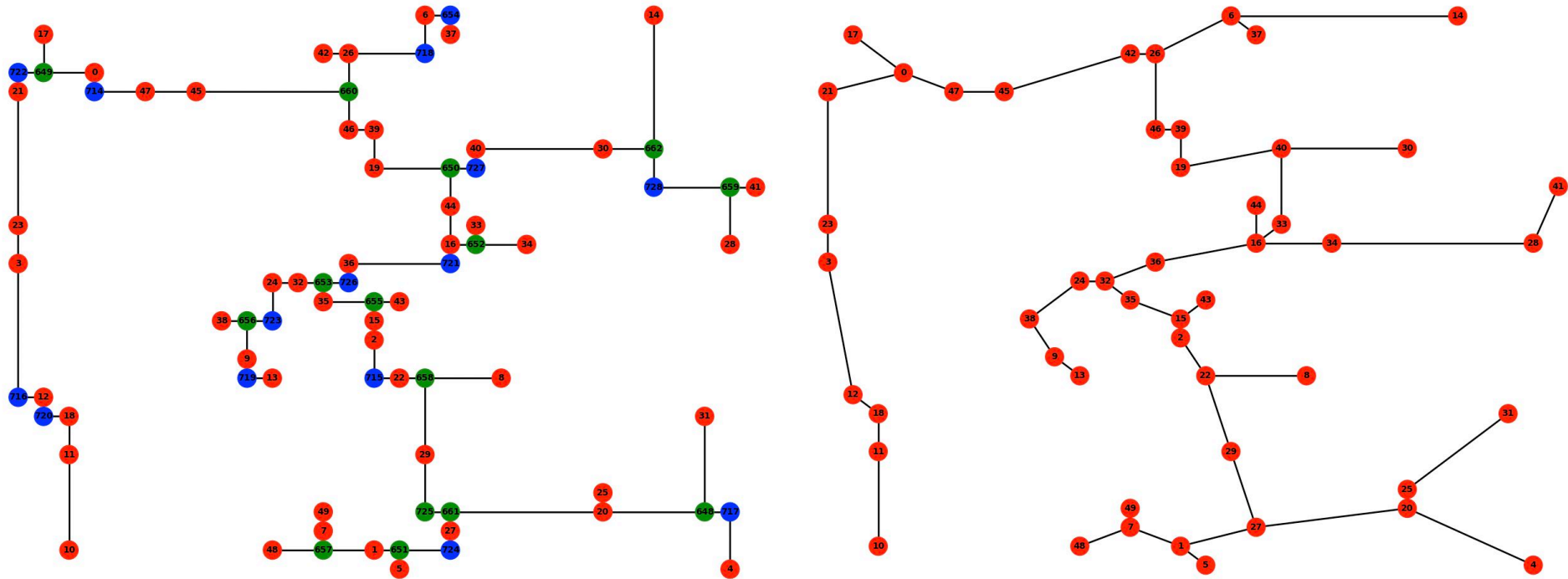
10

20

50

100

50 points



5

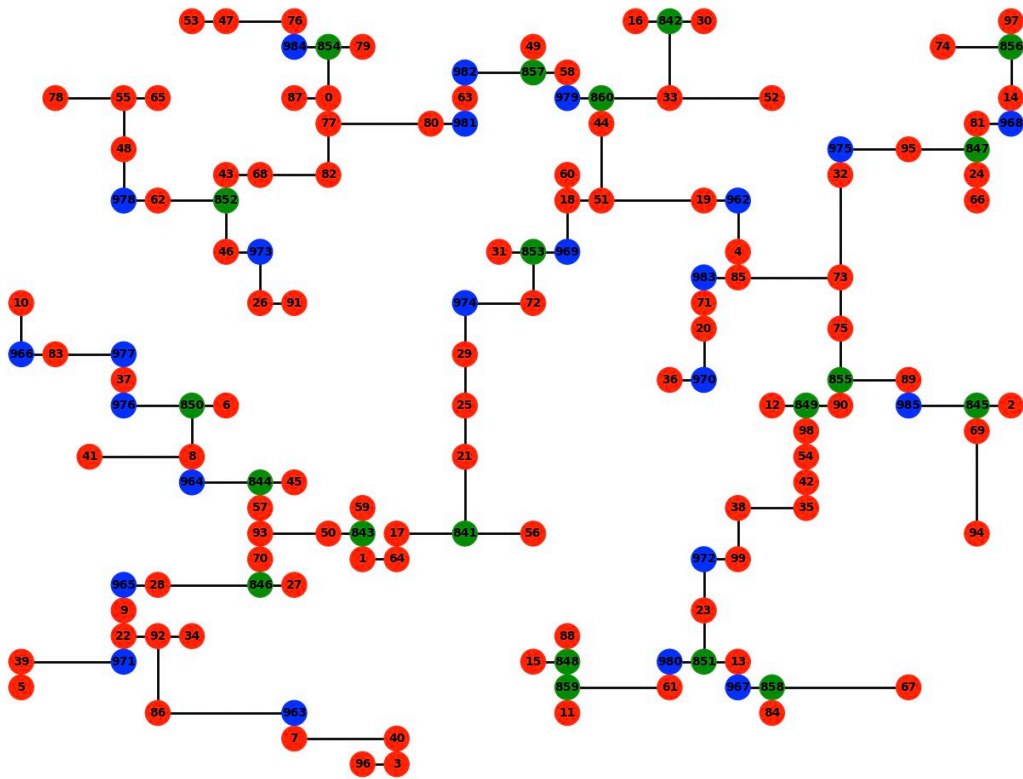
10

20

50

100

100 points



- 100 points
- 20 Steiner points
- MST WL – 242
- BorahWL – 221
- 2 pass
- PrimRuntime – 106ms
- KruskalRuntime – 49ms
- BorahRuntime – 1264ms

5

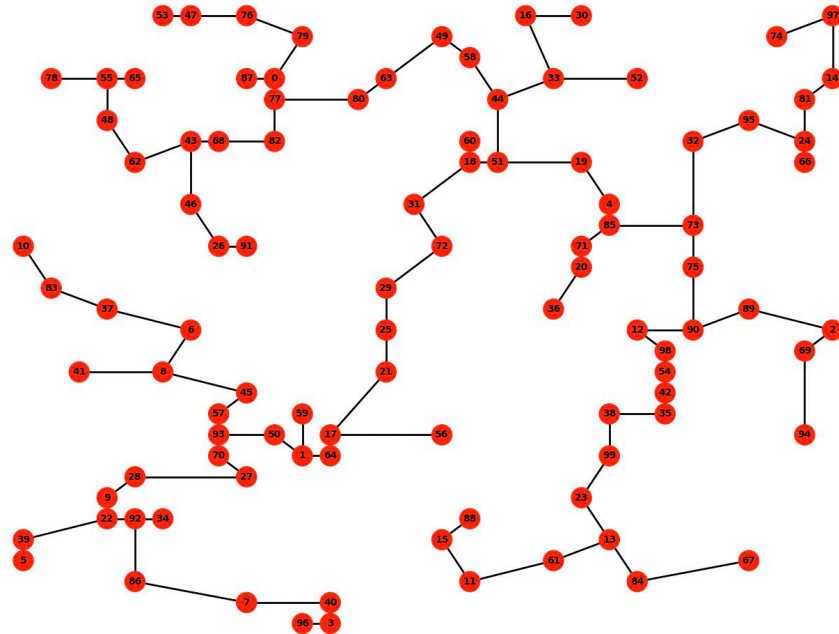
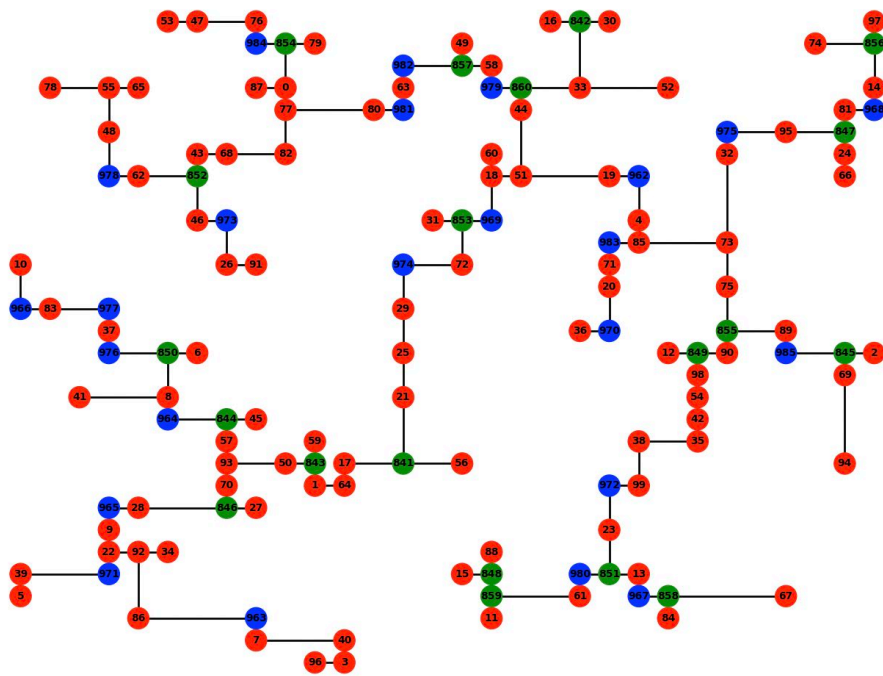
10

20

50

100

100 points



50 pointsprogressvideo



Comparisortable

Benchmark	Wirelength MST	Wirelength final tree	# Steiner points	# pass	Runtime Prim	Runtime Kruskal	Runtime Borah
5	21	18	2	1	1ms	1ms	1ms
10	24	20	3	1	1ms	1ms	2ms
20	37	34	3	1	2ms	2ms	6ms
50	183	163	16	2	15ms	11ms	236ms
100	242	221	20	2	106ms	49ms	1264ms

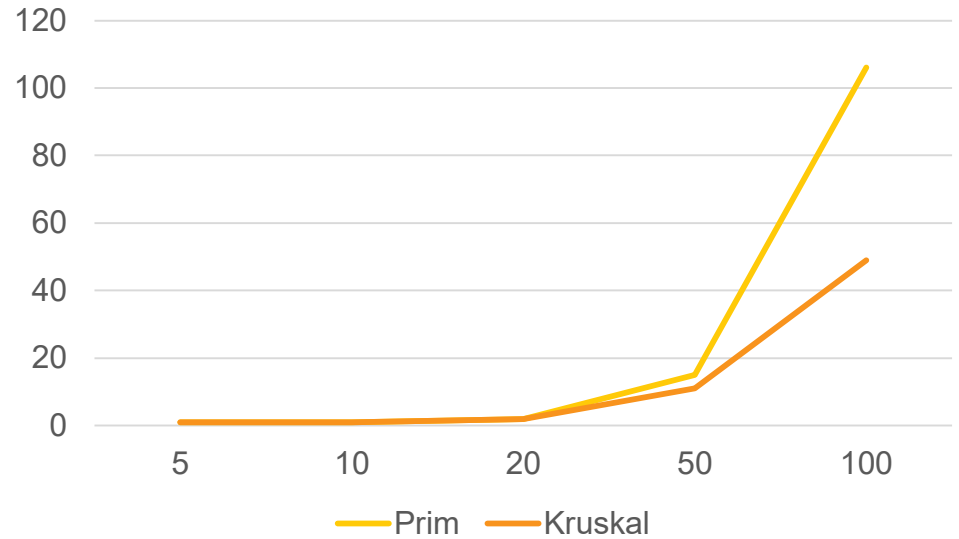
More informations

Benchmark	Wirelength reduction	Average Wirelength reduction/pass	Average wirelength reduction/steiner point	Time spent per point	Percentage of wirelength reduction over total wirelength	# of Steiner per #of points
5	3	3	1,5	0,2	14%	0.4
10	4	4	1,334	0,2	17%	0.3
20	3	3	1	0,3	8%	0.15
50	20	10	1,25	4,72	11%	0.32
100	21	10,5	1,05	12,64	9%	0.2

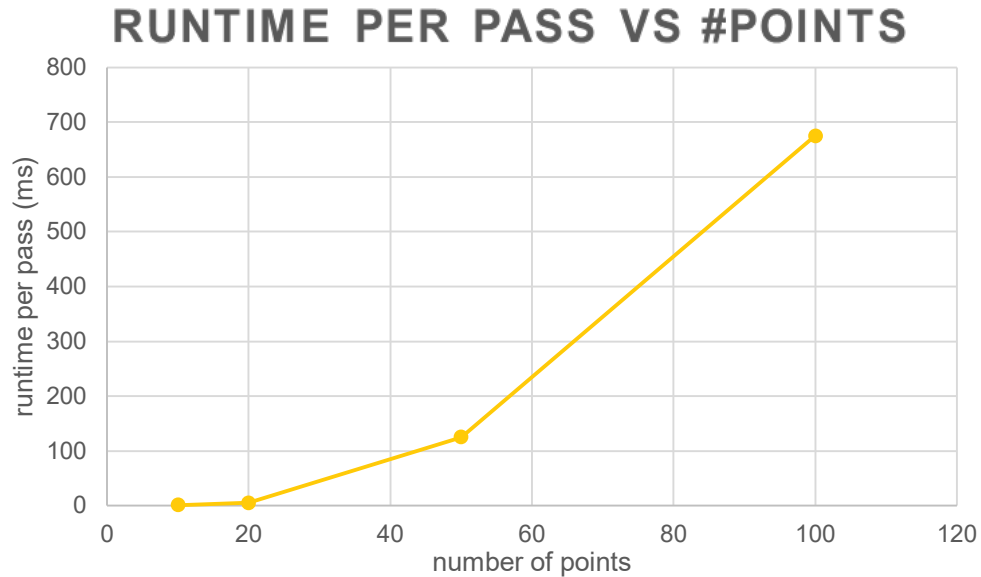
Prim vs Kruskal

# points	Runtime Prim	Runtime Kruskal
5	1ms	1ms
10	1ms	1ms
20	2ms	2ms
50	15ms	11ms
100	106ms	49ms

Prim vs Kruskal



Runtime per Pass VS. # of points



Shape of this graph should be close to $\log(n)$

Contact us

■ If you need to contact us, here are our informations:

- ▶ Anthony Prudhomme anthony.prudhomme84@gmail.com
- ▶ Guillaume Guérin guillaume.guerin123@gmail.com