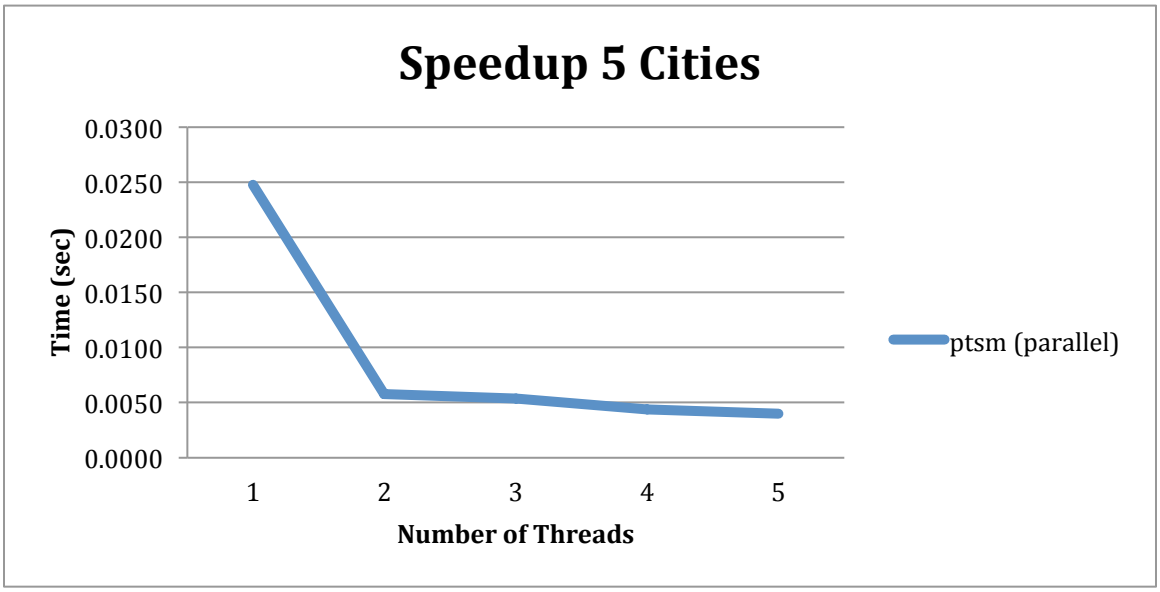
Speedup - 5 Cities

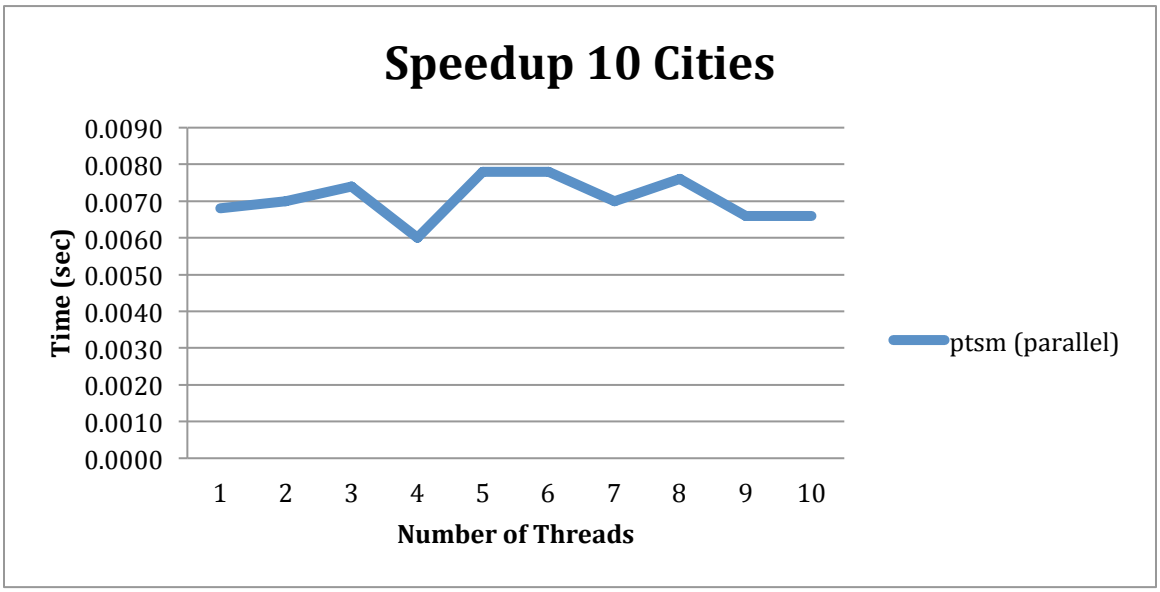
	5 Cities	
Threads	ptsm (parallel)	tsmoptimal (serial)
1	0.0248	0.0056
2	0.0058	0.0056
3	0.0054	0.0056
4	0.0044	0.0056
5	0.0040	0.0056



Speedup - 10 Cities

	10 Cities	
Threads	ptsm (parallel)	tsmoptimal (serial)
1	0.0068	0.1248
2	0.0070	0.1248
3	0.0074	0.1248
4	0.0060	0.1248
5	0.0078	0.1248
6	0.0078	0.1248
7	0.0070	0.1248

8	0. 0076	0. 1248
9	0. 0066	0. 1248
10	0. 0066	0. 1248

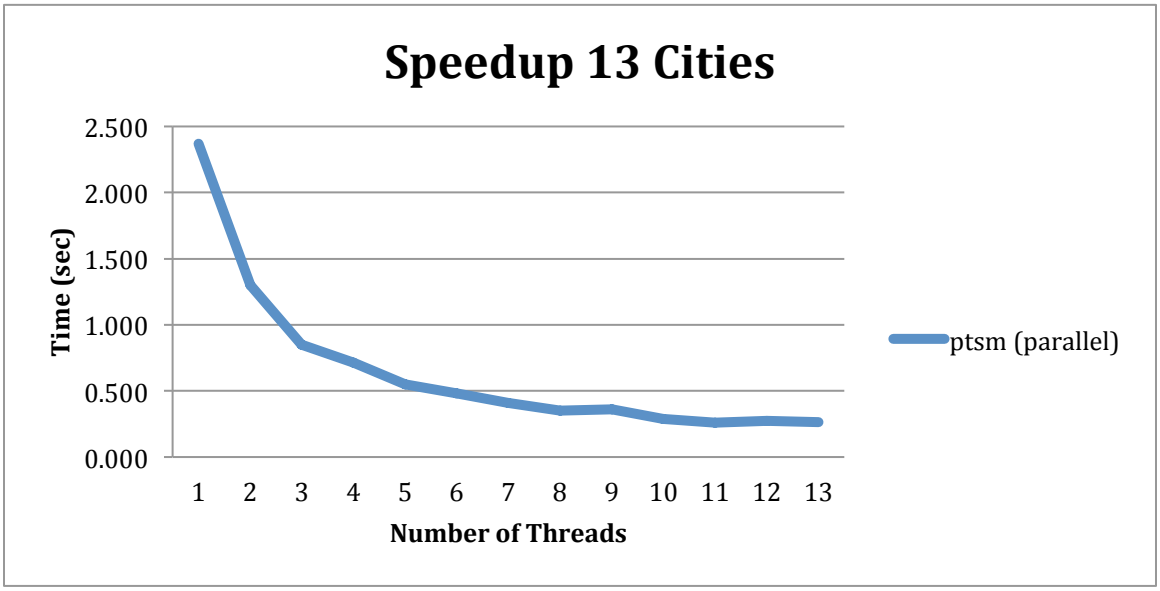


However, as you see the above total running time for my parallel version is a bit fluctuated. It is because the problem size is small for my implementation and the non-deterministic part of running time is quite significant proportional to the part determined by problem size.
So, let me show you the speed up over problem size of 13 and 14 cities, which will yield a reasonable speed up curve. The decaying factor is roughly 2 because the parallel approach is mainly task based. (not 2 because of thread overheads)

Speedup – 13 Cities

	13 Cities	
Threads	ptsm (parallel)	tsmoptimal (serial)
1	2. 370	232. 898
2	1. 298	232. 898
3	0. 849	232. 898
4	0. 714	232. 898
5	0. 549	232. 898
6	0. 481	232. 898
7	0. 407	232. 898

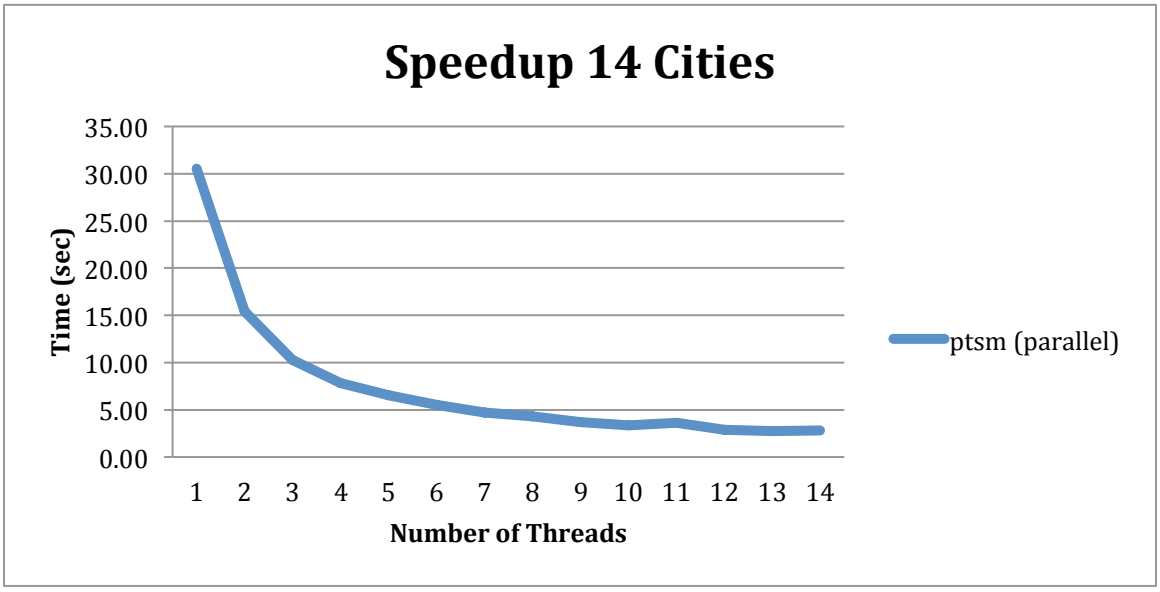
8	0.351	232.898
9	0.359	232.898
10	0.288	232.898
11	0.260	232.898
12	0.272	232.898
13	0.262	232.898



Speedup – 14 Cities

	14 Cities	
Threads	ptsm (parallel)	tsmoptimal (serial)
1	30.53	
2	15.43	
3	10.24	
4	7.81	
5	6.57	
6	5.53	
7	4.72	
8	4.29	
9	3.71	
10	3.39	
11	3.64	
12	2.86	

13	2. 72
14	2. 80



Specs of the Processors & Observations

Full details of the processors spec, please refer to the **processors_spec.txt** in the zip file.

Quick and basic summary of the processor spec:

1. Model name : AMD Opteron(TM) Processor 6272
2. cpu MHz : 2099.946
3. cache size : 2048 KB
4. There are in total 8 processors. (crunchy 1)
5. Each processor is of 8 cores, in total 64 threads available.

The above graphs doesn't make reasonable sense when the problem size is small, it is because there are some system and thread overheads and some non-deterministic running time. When the problem size is small, this portion is significant and we won't be able to observe a consistent speedup over increase of number of threads.

When the problem size is large enough (13/14 cities) that the parallelism helps, we can observe a consistent speed up over the number of threads. The running time is roughly reduced half when we double the number of threads from 1-14 threads. It is not exactly half because of some threading overheads. We are getting consistent

ratio of speed up is also because we are getting the actual number of physical threads that ptsm program requested (From the above processor spec, we see the crunchy1 machine has in total 64 threads, which is more than requested.). The task-parallelism nature ensure the consistent ratio when the actual number of threads are as requested.