Graphing Iterations and Speed

We examine what factors are responsible for the time taken to run a primal loop iteration: is it just the number of routes, or is the circumstance also responsible? The following 5 cases are considered:

Jobs only
Cov only (technically, after jobs-only burden)
DCG only
DCG in MCG after jobs-only burden
DCG in SCG after jobs-only and cov-only burdens

The x-variable is the number of routes present in the system total (for jobs+cov together, we sum the two up). The y-variable is the amount of time taken.

The reason I use all 5 cases is because I want to test if it is only the number of routes, and not other characteristic differences (e.g. cov with jobs already is easier than cov without jobs; heuristic makes primal easier to run) that affects time.

Results

<u>Jobs Only:</u> See *Figure 1*.

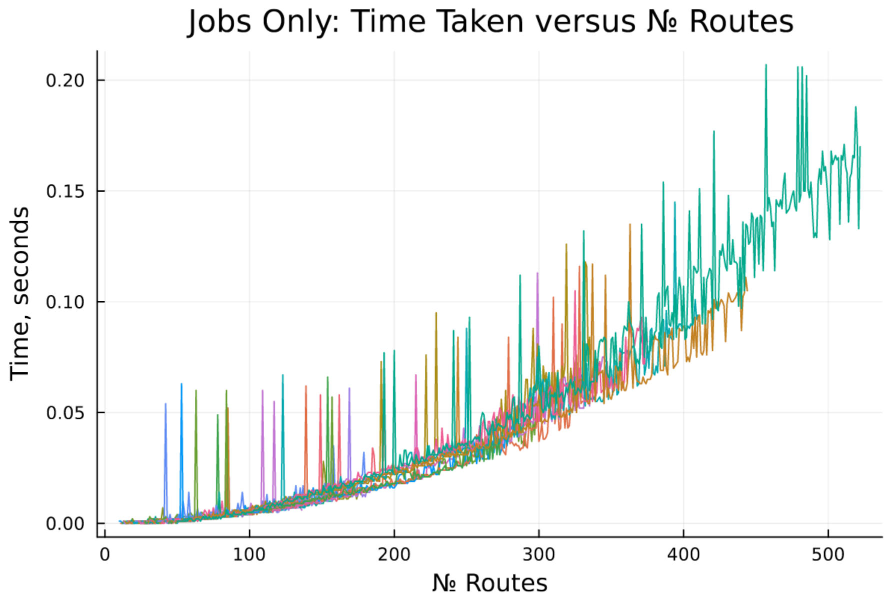

Figure 1: Each colored line represents a certain number of jobs. The progression on the x-axis shows the number of routes in the system at the time of primal optimization.

For each job number, as the number of routes increases, so does the time required to execute the primal. The correlation between # routes and time taken is not perfect; the higher # jobs we start with, the more elevated the time taken for the same # of routes as a lower-job primal. Reason: The # of constraints for the higher-job case is higher.

There is also a large number of spikes which occur every so often: a total of 51, with 18 of them in the longest case generating over 520 routes.

While the "elevation" phenomenon described above proves that circumstance is partially responsible for primal runtime, the main driver for runtime is the number of routes used.

Benchmarks:
100 routes: .006s (total .3s) | 200 routes: .025s (total 2s) | 300 routes: .05s (total 13s)

<u>Covs Only:</u> See *Figure 2* for results.

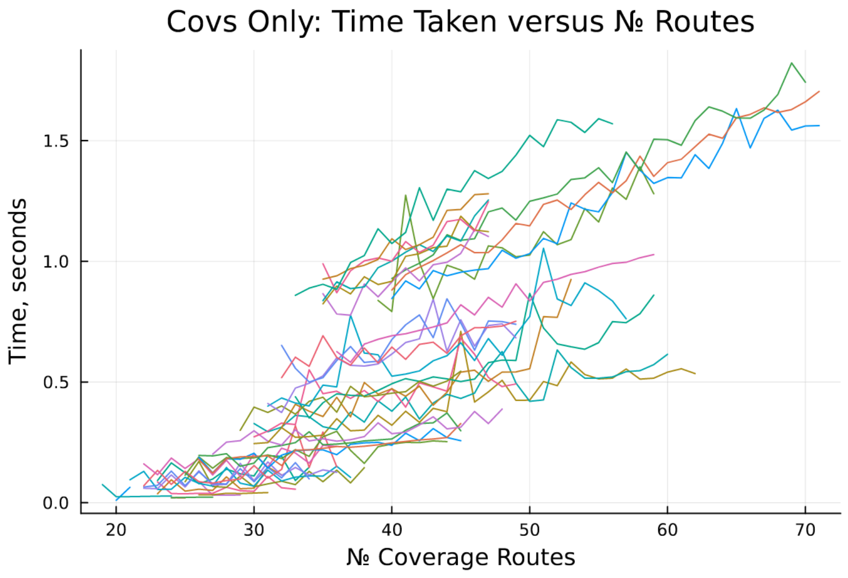

Figure 2: Solving coverage after given optimal jobs-only solution.

The same elevation we saw with jobs only is plain to see here as well. It is no surprise, as the complexity of cov constraints increases with more jobs.

This was done with treasure-hunting. It is interesting that jobs SP has far more routes with much shorter complete times, while cov SP is the opposite.

<u>DCG Only:</u> See *Figure 3* for results.

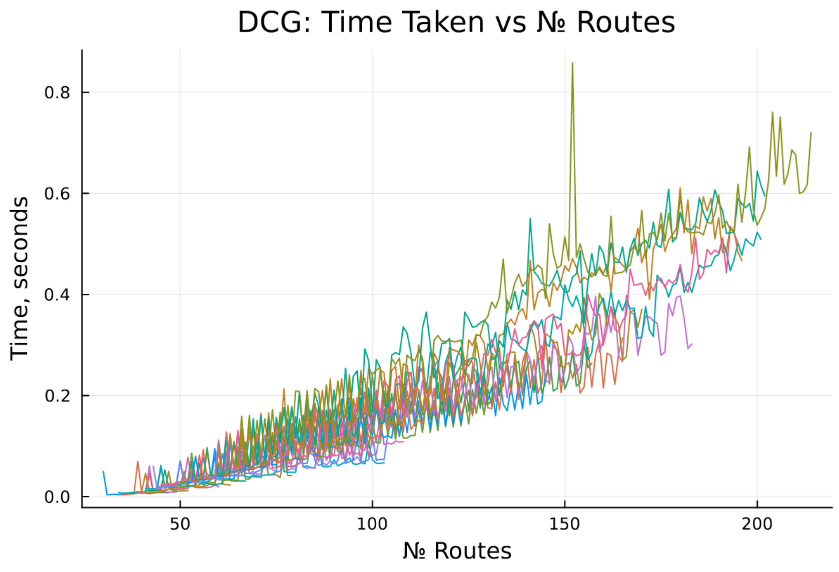

Figure 3: Time taken for DCG primal versus number of routes in primal.

Elevation is visible, but a correlation between time taken and number of routes (jobs+cov) is clear. Thus, if we have less routes to start with in DCG, we will do better, as the primal is the bottleneck from previous studies: see page 6 of "Bottleneck Analysis of mcg TH.pdf", which shows a whopping 95% of time is burned on the primal!

Benchmarks:
100 routes: .2s (total 5s) | 150 routes: .4s (total 20s) | 200 routes: .6s (total 40s)

<u>DCG in MCG:</u> See *Figure 4* for results.

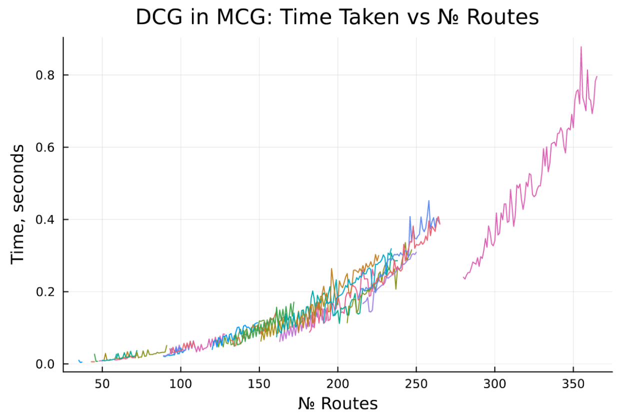

Figure 4: Time taken for DCG within MCG.

We consider the number of routes given by both jobs and coverage. Here, elevation is less noticeable, and time spent solving the primal is very well correlated with number of routes because there is no significant elevation. We can take a closer look in *Figure 5* with the magenta "outlier" removed.

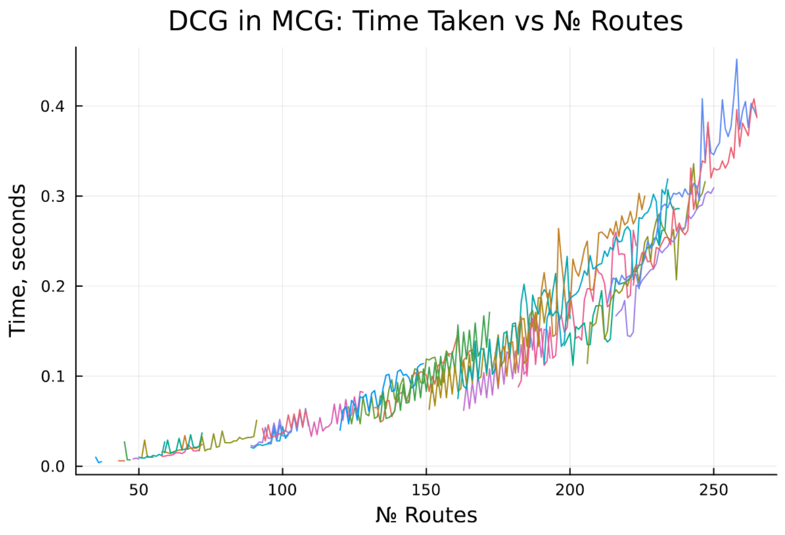

Figure 5: Clearer DCG in MCG.

Benchmarks:
100 routes: 0.04s | 150 routes: 0.1s | 200 routes: 0.2s | 250 routes: 0.35s

<u>DCG in SCG Only:</u> See *Figure 6* for results.

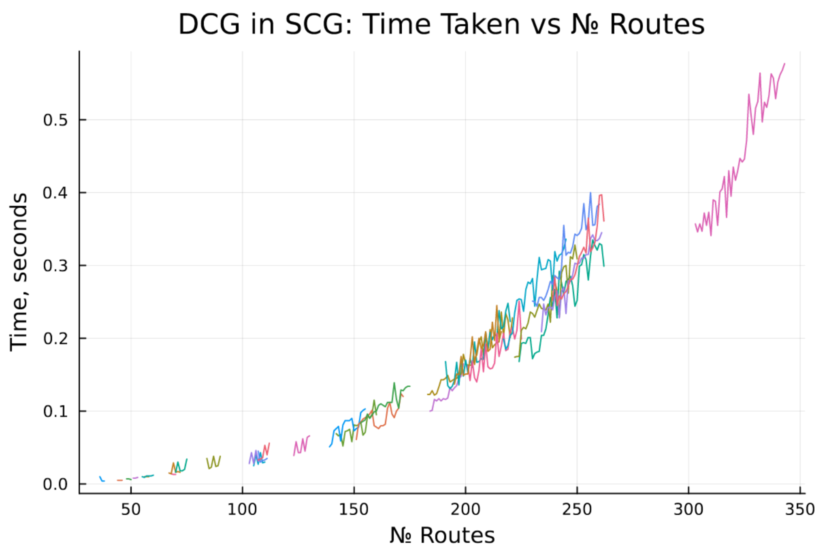

Figure 6: DCG in SCG.

Notice how the plots for DCG in: DCG, MCG, SCG progressively get less cluttered. This is because warm starts from jobs and/or cov reduce the number of iterations spent on dcg, which is computationally the most expensive by far. The pattern is plain to see: elevation is still a factor, but its impact grows progressively less as we initiate more warm starts.

Benchmarks are largely the same as for DCG in MCG, though slightly lower.

General Conclusions
The correlation between runtime and number of routes is far more powerful than elevation effects. If we can trim the number of routes we use for our warm start into dcg, we will probably get to the answer faster.

In scenario A, we use *all* the routes from warm starts and put them into dcg to complete the problem. In scenario B, we use only routes with z-value 1 (including fractional values) from warm starts and put them into dcg. My hypothesis is the following. The time potentially "wasted" in B "re-deriving" better routes that would've been present in A is far outweighed by the much longer RMP times in A. In particular, my guess is that the warm starts are sufficiently good and only minor tweaks to them are necessary. For example, we saw how for n_jobs = 50, the jobs-only optimum was 3'532 while the jobs optimum for jobs+cov was 3'569.

We soon perform the following two experiments, whose algorithms I will call XJW (exclusive jobs warm start) and XJCW (exclusive jobs + cov warm start). They are the same as mcg and scg, respectively, just with only positive z- and x-values put into dcg.

The less elevation there is in the scenario (dcg/mcg/scg), the better my argument, because it means "dragging" the number of total routes in a primal iteration will have a larger effect.