Simple Multiplication Goes Awry

I hit a mysterious error when I ran my double column generation loop: After awhile, the same route was being pumped into both the list of job routes and list of coverage routes, and for both job and coverage, the reduced cost of each of those routes was the same every time.

I was also frustrated with the impossibility of modeling concave problems. A week ago, the solver kept splitting routes for several n_jobs cases, and I decided to punish splitting by using the ceiling function: $\sum_q C^q \lceil z^q \rceil$.

That way, if a route was used partially, it would be counted the same as if it were used entirely, thereby destroying split routes altogether and allowing convergence to the correct value.

Sadly, Gurobi wouldn't allow this, nor my other idea of the square root function $\sum_q C^q \sqrt{z^q}$.

Since the model told me 0th, 1st, and 2nd powers were allowed, I created a function $1-(1-z)^2$, which had 0 at 0, 1 at 1, and a similar curve to the square root function. Of course, Gurobi would then tell me that nonconvex problems were not solvable. That marked the end of all attempts to "punish" route-splitting.

But while I was playing around with the code, something else caught my eye.

Consider my formulation for the job routes only. For n = 25, the correct answer is 2'845. (The PT NyVA >= gives 2'853, but this is not the primary concern.) So if I have:

<variables, constraints>

Objective: MIN $\sum_q C^q z^q$

the answer is 2'845.

Now, what should I get if I said this instead?

<the same variables and the same constraints, nothing changed at all>

Objective: MIN 2 * $\sum_q C^q z^q$

Obviously it should be 2 * 2'845 = 5'690, right?

According to Gurobi (and HiGHS optimizer too), no.

What follows is the loop's running forever, eventually pushing out the same generated column with the same reduced cost, over and over. I had to terminate the loop and the objective was 6'784. Divide by 2 to get 3'392, far above the correct answer.

It goes without saying that if I have an objective and the answer is S, an objective with a scaled factor f should yield answer fS, and in around the same time.

Apparently Gurobi and HiGHS can't do this.

When f is smaller than 1, the optimizers do the job as quickly as when f is 1 (the default). But the answer is completely wrong. Even worse, when f is any bigger than 1, the optimizer never halts, eventually printing out an endless stream of the same route and the same negative reduced cost. Here is a table of: f, outputted objective, scaled answer.

0.1	929	9'290
0.2	1'357	6'785
0.3	1'562	5'206
0.4	1'323	3'307
0.5	1'709	3'418
0.6	1'940	3'233
2/3	2'095	3'142
0.75	2'307	3'076
1	**2'853**	**2'853**
1.2	3'446	2'871
1.5	4'562	3'041
1.7	5'910	3'476
2	6'784	3'392
2.2	8'258	3'753
2.5	9'603	3'841
2.7	10'231	3'789
3	12'187	4'062
3.5	14'479	4'136

The same type of error I saw with my double column generation scheme is the same type of error I am observing here whenever the scaled factor is greater than 1.

It doesn't make any sense.

And yes, I have tried every shortcut I could search up. Setting the objective with sum coefficient 1 (i.e. no coefficient) and then set_objective_coefficient to 2 doesn't work. Nor does using MIN sum(...) + sum(...).

This is evidence that there is nothing wrong with my column generation scheme, or the code I've written, because it all works without issue when I use coefficient 1. There is probably something wrong within the Gurobi and HiGHS black box. This might also include the calculation of dual variables!