The Mystery of Signing - Colgen without pruning

Within the colgen loop is a mysterious line of code which causes much havoc.

```
@constraint(modelcg, unique[i in 1:n_jobs], sum(u[i][q] * z[q] for q in 1:Q) == 1);
```

The full formulation specifies that this should be a strict equality, as only exactly one path, formed by one vehicle, can cover a job node. It is $\sum_q u_i^q z^q = 1$.

When I performed colgen, I started with $\sum_q u_i^q z^q \geq 1$.

This was to ensure the solver would print a feasible solution, and in principle, relaxing constraints works well for the solver. Unfortunately, I would run into trouble with certain cases of n_jobs in which the objective would be just above the true minimum, and the routes ever so slightly off from the optimal set.

This was the case for n = 20, 22, 24, 28, 31, 32, 33, 34. (We were able to measure up to n = 35 inclusive because SP took over 10 minutes after that.)
With >=, here are the route errors.

n = 20: true answer 3'028; false answer 3'077
4, 19, 5, 20 18, 11, 12, 13, **14**
4, 19, 5, 20, **14** 18, 11, 12, 13

n = 22: true answer 3'364; false answer 3'383

1, 2, **13, 14**	4, 19, 5, **6, 7**	8, 9, **10**	18, 11, 12, **20**	21, 22, **3**	15, 16, 17
1, 2, **3**	4, 19, 5, **20**	8, 9	18, 11, 12, **13, 14**	21, 22, **10**	15, 16, 17, **6, 7**

n = 24: true answer 3'445; false answer 3'518
1, 2, 3 4, 19, 5, 20 15, 16, 17, 6, 7 8, 9, 10 18, 11, 12, 13, 14 21, 22, 23, 24
* There is route sharing in the >= case! * (There's a bunch of 1/3's and 2/3's)

n = 28: true answer 2'933; false answer 2'949
11, 12, 15 16, 17, 18, 19, 13, 14
11, 12, 15 * route sharing! *

Route-sharing occurs at the 13-14; 16-19 jobs: there is a 0.5 z-probability associated with 11, 12, 15 (italicized), and also with:
11, 12, 13, 14, 15
16, 17, 18, 19, 13
16, 17, 18, 19, 14

n = 31: true answer 3'284; false answer 3'522
Lots of route-sharing takes place (1/3's and 2/3's)

n = 32: true answer 3'292; false answer 3'356
13, 14, 15, 16, 17, 18 23, 24, 25, 26
13, 14, 15, 16, 17, 18 23, 24, 25, 26, **18**
All routes in the >= case have z = 1, indicating a double visit at job 18!

n = 33: true answer 3'406; false answer 3'740
Extensive route-sharing

n = 34: true answer 3'649; false answer 3'691
Extensive route-sharing

Observe that with >=, only n = 20, 22 yield "true wrong answers"; the rest, which have route-splitting/route-sharing, can be explained through deficiencies in the column generation model itself. This phenomenon is further explained in the large paragraph on page 2 of "Timing the Big 4.pdf".

However, changing the >= to == is not a perfect patch. While it does avoid many issues accrued in the >= formulation, which has too much route sharing, it was not a bulletproof solution. New errors were introduced that weren't present before! The problematic cases were n = 29, 30, 31, 32, 33, and 34.

n = 29: true answer 3'003; false answer 3'019
11, 12, 15 **16**, 17, 18, 19, 13, 14
16, 11, 12, 15 17, 18, 19, 13, 14

n = 30: true answer 3'282; false answer 3'438
5, 6, 7, **8, 9** 13, 14, 15, 16, **17, 18** 23, 24, 25, 26
5, 6, 7, **17, 18** 13, 14, 15, 16 23, 24, 25, 26, **8, 9**

n = 31: true answer 3'284; false answer 3'359
Lots of route-sharing (1/7's and 2/7's)

n = 32: true answer 3'292; false answer 3'345
13, 14, 15, 16, 17, **18** 23, 24, 25, 26
13, 14, 15, 16, 17 23, 24, 25, 26, **18**

n = 33: true answer 3'406; false answer 3'443
5, 6, 7, 8, **9** 23, 24, 25, 26
5, 6, 7, 8 23, 24, 25, 26, **9**

n = 34: true answer 3'649; false answer 3'703
13, 14, 15, 16, 17, **18** 23, 5, 24, 25, 26
13, 14, 15, 16, 17 23, 5, 24, 25, 26, **18**

To summarize (light red = route-sharing issue; dark red = true wrong answer):

Case	20	22	24	28	29	30	31	32	33	34
True	3'028	3'364	3'445	2'933	3'003	3'282	3'284	3'292	3'406	3'649
>=	**3'077**	**3'383**	**3'518**	**2'949**	**3'003**	**3'282**	**3'522**	**3'356**	**3'740**	**3'691**
==	**3'028**	**3'364**	**3'445**	**2'933**	**3'019**	**3'438**	**3'359**	**3'345**	**3'443**	**3'703**

For smaller cases == offers much better performance than >=, avoiding both barely-incorrect answers and route-splitting. However, at n=29 this shifts, with >= providing many correct answers that == misses. After n=31 both >= and == give wrong answers, but == often gives incorrect answers without route-splitting, while >= at least has 'an excuse' in that it always splits the route.

Let's see how we do when we prune routes (covered in another document).