Appendix

A. Psuedocodes

A.1. SAA Algorithm for p-alt Problem

Algorithm 1 Sample Average Approximation Framework

Input: M, N, N'. ▷ Specifying algorithm parameters.

for $m = 1, \ldots, M$ **do**

  Generate $\mathcal{S} \subset \Omega$, a random demand N-sample for sample problem m.

  Generate (independently) demand sample of size $N' >> N$, for evaluation.

  Solve sample problem m given by (3) to obtain a first-stage design for iteration m, $(\hat{r}^m, \hat{y}^m)$.

  for $n = 1, \ldots, N'$ **do** ▷ Evaluation phase.

    Solve second-stage subproblem (1e) given $(\hat{r}^m, \hat{y}^m)$ to compute $Q(\hat{r}^m, \hat{y}^m, \omega^n)$.

  end for

  Calculate $\hat{v}_{N'}(\hat{r}^m, \hat{y}^m) = \sum_{(i,j)\in\mathcal{A}} c_{ij}\hat{r}^m_{ij} + \frac{1}{N'}\sum_{n=1}^{N'} Q(\hat{r}^m, \hat{y}^m, \omega^n)$, the approximate expected total cost for design $(\hat{r}^m, \hat{y}^m)$.

end for

Select $(\hat{r}^*, \hat{y}^*) \in \underset{m\in\{1,\ldots,M\}}{\arg\min} \{\hat{v}_{N'}(\hat{r}^m, \hat{y}^m)\}$.

Compute the optimality gap estimate for $(\hat{r}^*, \hat{y}^*)$ and its variance using Equations (5) and (6).

return $(\hat{r}^*, \hat{y}^*)$, optimality gap estimate, and optimality gap variance estimate.

A.2. RandCut Separation Heuristic

Algorithm 2 RandCut

Input: $\mathcal{G}, \underline{r}, \underline{z}, RCMaxIter, \gamma^{RC}$. ▷ Specifying algorithm parameters.

for $\omega \in \mathcal{S}$ **do**
  $\mathcal{L} \leftarrow \emptyset$. ▷ $\mathcal{L}$ is the list of cutsets for which cut inequality is violated.
  for $t = 1, \ldots, RCMaxIter$ **do**
    $V \leftarrow \emptyset$.
    for $v \in \bar{V}$ **do**
      Choose a random byte $b \in \{0, 1\}$.
      if $b = 1$ **then**
        $V \leftarrow V \cup v$.
      end if
    end for
    if Equation (8) is violated for V **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup V$.
    end if
    if Equation (8) is violated for $\bar{V}$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \bar{V}$.
    end if
  end for
  Sort cutsets in $\mathcal{L}$ in non-increasing order according to violation of Equation (8), and add the inequalities corresponding to the first γ^{RC} cutsets to the model.
end for

A.3. RGContraction Separation Heuristic

Algorithm 3 RGContraction

Input: $\mathcal{G}, \underline{r}, \underline{x}, \underline{z}, RGMaxIter, l, \gamma^{RG}$. ▷ Specifying algorithm parameters.

for $\omega \in \mathcal{S}$ **do**
  $\mathcal{L} \leftarrow \emptyset$. ▷ $\mathcal{L}$ is the list of cutsets for which cut inequality is violated.
  for $(i,j) \in \mathcal{G}$ **do**
    Compute slack for (i,j) with respect to constraint (1f) using $\underline{r}, \underline{x}$ and $\underline{z}$.
    Compute fractional parts for arcs given by $(\underline{r}_{ij} + \underline{z}^{\omega}_{ij}) - \lfloor (\underline{r}_{ij} + \underline{z}^{\omega}_{ij}) \rfloor$.
  end for
  for $t = 1, \ldots, RGMaxIter$ **do**
    $ArcList \leftarrow$ Sort arcs by their slacks in non-increasing order, then by their fractional parts in non-increasing order.
    $\mathcal{G}' \leftarrow \mathcal{G}$. ▷ $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ represents the contracted graph.
    while $ArcList \neq \emptyset$ **and** $|\mathcal{N}'| > \mathcal{N}^{final}$ **do**
      $RCL \leftarrow$ Select first l arcs in $ArcList$.
      $(u,v) \leftarrow$ Select an arc randomly from RCL.
      Contract arc (u,v) in $\mathcal{G}'$ while updating the slacks and fractional parts as described above.
      Update $ArcList$.
    end while
    $\mathcal{V}' \leftarrow$ Enumerate cutsets on $\mathcal{G}'$.
    for $V \in \mathcal{V}'$ **do**
      if Equation (8) is violated for V **then**
        $\mathcal{L} \leftarrow \mathcal{L} \cup V$.
      end if
    end for
  end for
  Sort cutsets in $\mathcal{L}$ in non-increasing order according to violation of Equation (8), and add the inequalities corresponding to the first γ^{RG} cutsets to the model.
end for

A.4. Heuristic-SS Algorithm

Algorithm 4 Heuristic-SS

1: $SSIterate \leftarrow True$.
2: Initialize multipliers: $\rho^0_{ij\omega} = \frac{\hat{c_{ij}}}{Q} \quad \forall (i,j) \in \mathcal{G}, \omega \in \mathcal{S}$.
3: $t \leftarrow 0$ ▷ Initialize iteration counter.
4: **while** $SSIterate$ **do**
5: Solve (9) with multipliers $\rho^t_{ij\omega}$ to obtain a feasible integer set of first-stage variables for iteration t of slope scaling, (r^t, y^t).
6: Update $SSIterate$ by checking stopping criteria.
7: $\rho^{t+1}_{ij\omega} \leftarrow$ Adjust multipliers using (10).
8: $t \leftarrow t+1$ ▷ Updating iteration counter.
9: **end while**
10: $(\hat{r}^m, \hat{y}^m) \leftarrow (r^{t-1}, y^{t-1})$.
11: **return** $(\hat{r}^m, \hat{y}^m)$.

B. Comparison and Analysis of Solution Approaches

In this appendix, we present a more in-depth comparison of the exact and heuristics approaches discussed in Section 5. Specifically, we present a visual representations of the trade-off between solution quality and solution time, as well as the one between solution quality and the time to final design (TFD).

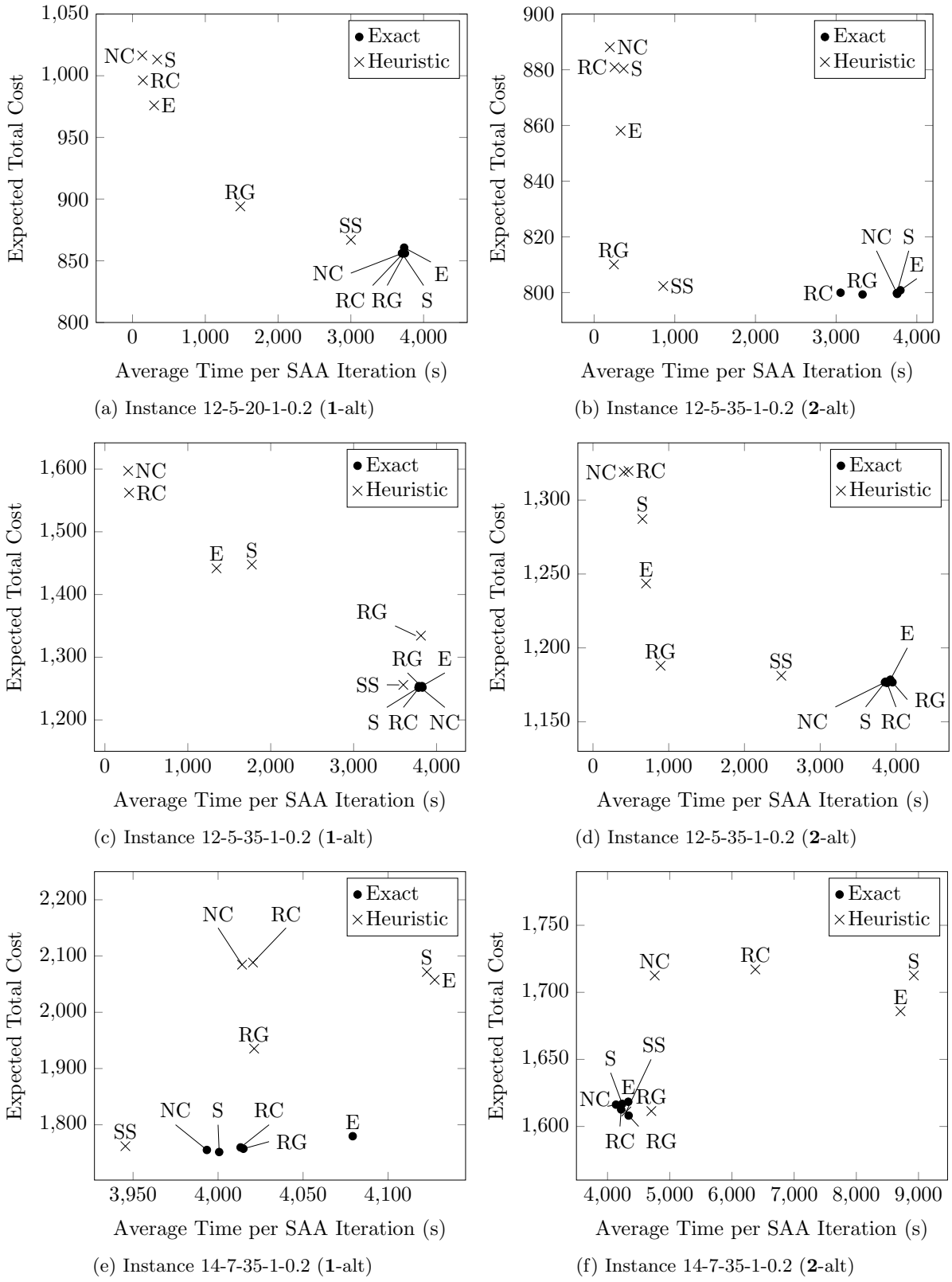
In Figure 1, we plot for each instance and each method the average *total cost estimate* taken over the $M = 10$ SAA iterations against the average *time of an SAA iteration*. We define the *total cost estimate* of a sample problem as the first-stage cost of that sample problem plus the recourse cost estimate obtained by evaluating the first-stage design using $N' = 1000$ scenarios. The *time of an SAA iteration* is defined as the time to solve the sample problem plus the total evaluation time of all the $N' = 1000$ scenarios for that sample problem. Each row of plots in the figure corresponds to a single instance, with the two plots in that row representing the different alt structures considered (**1**-alt and **2**-alt).

In terms of exact approaches, we point out that in Figure 1 that there is very little variability in terms of these approaches in both solution time and solution quality. Due to the sizes of the instances considered and the fact that almost all the sample problems time out at the hour mark, this is not a surprising observation. In the case of Figure 1b, for instance, the variability in time is a result of two of the approaches, namely Exact-RandCut and Exact-RGContraction, solving some of their sample problems to optimality before the hour mark, whereas the other three exhausted the time limit without being able to do so. This suggests that the two separation heuristics RandCut and RGContraction are effective separation heuristics for the cut inequalities, and have helped in the solution process.

Figure 2 shows that there is slightly more variability in TFD among the five exact approaches. Generally, Exact-RGContraction is the slowest to settle on a final design, and this rarely comes with an increase in design quality. In addition, Exact-Select seems to have a relatively low TFD while obtaining designs of good quality (except perhaps in Figure 2f).

For the heuristic approaches, the first thing to note from Figure 1 is that there is much more variability in terms of solution quality and runtime among the different heuristic approaches compared to their exact counterparts. Moreover the average runtime of the heuristic approaches indicates that these heuristics can be faster than the exact approaches while still finding solutions of reasonable quality. In fact, on these instances, some heuristic approaches (e.g. Heuristic-SS) are comparable in solution quality to the exact approaches.

To no surprise, we see that relaxing the integrality requirements for the second-stage variables without any method of compensating for that relaxation (NC) yields solutions that are usually much worse in quality compared to the exact approaches. In all six cases, Heuristic-NC is either the worst heuristic method in terms of solution quality or a very close second, and the difference in quality compared to the Heuristic-RG, Heuristic-E, Heuristic-S, and Heuristic-SS is significant. Furthermore, the algorithm Heuristic-RC doesn't seem to offer much improvement on the quality of the solution obtained by the Heuristic-NC approach. On the other hand, both of these algorithms are generally the among the fastest to run (especially compared to Heuristic-RG and Heuristic-SS). However, on the largest instance they lose their edge in terms of speed. This is partially explained by noting that in both settings of the smaller two instances, the sample problems were easy to solve to optimality, thereby allowing Heuristic-NC and Heuristic-RC to terminate before exhausting



(a) Instance 12-5-20-1-0.2 (**1**-alt)

(b) Instance 12-5-35-1-0.2 (**2**-alt)

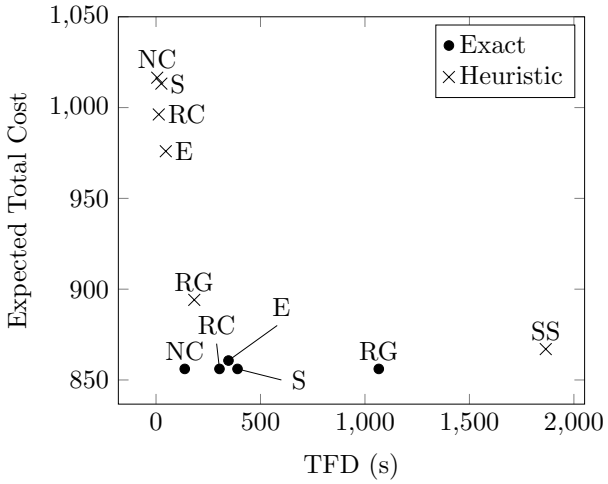
(c) Instance 12-5-35-1-0.2 (**1**-alt)

(d) Instance 12-5-35-1-0.2 (**2**-alt)

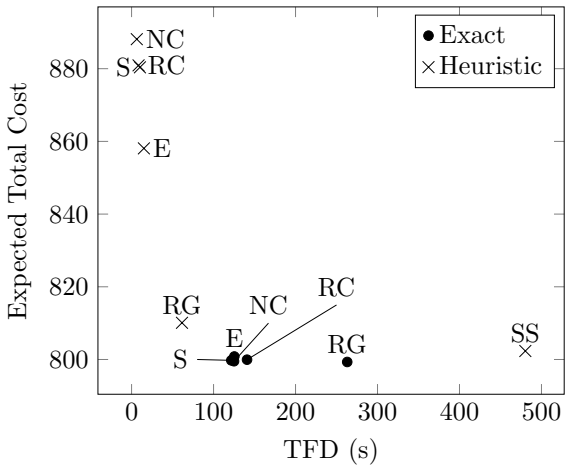
(e) Instance 14-7-35-1-0.2 (**1**-alt)

(f) Instance 14-7-35-1-0.2 (**2**-alt)

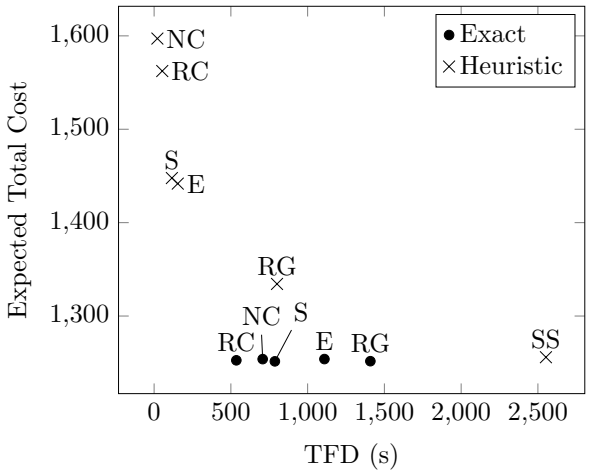
Figure 1 **Total Expected Cost vs. Average Time per SAA Iteration**



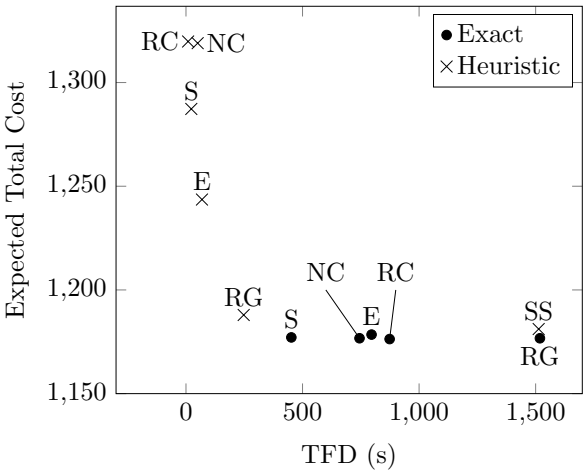
(a) Instance 12-5-20-1-0.2 (**1**-alt)



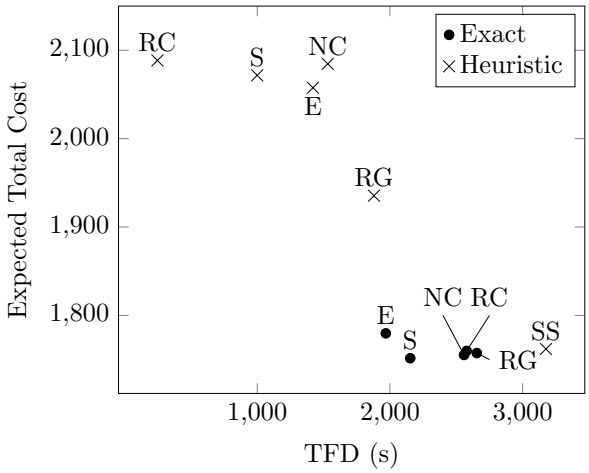
(b) Instance 12-5-35-1-0.2 (**2**-alt)



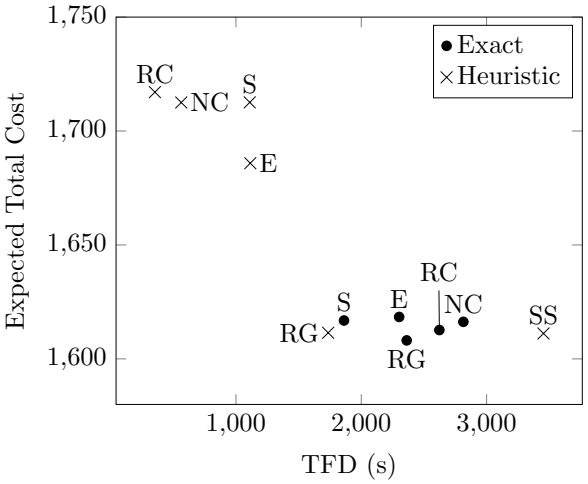
(c) Instance 12-5-35-1-0.2 (**1**-alt)



(d) Instance 12-5-35-1-0.2 (**2**-alt)



(e) Instance 14-7-35-1-0.2 (**1**-alt)



(f) Instance 14-7-35-1-0.2 (**2**-alt)

Figure 2 Total Expected Cost vs. TFD

the time limit. In the 14-7-35-1-0.2 instance, the sample problems are generally a lot more difficult to solve in the hour time limit, and even Heuristic-NC and Heuristic-RC exhaust this limit (as is the case with the other four) causing all six methods to have more comparable solution times for the sample problems. The differences seen in Figures 1e and 1f are due to time taken by the evaluation subproblems. In Figure 1f we see a curious case in which Heuristic-E and Heuristic-S take a much longer time compared to the other four. Specifically, the evaluation subproblems for those two approaches took roughly 4.5 times the total time taken by the evaluation phase of Heuristic-NC. An interesting thing to note is that Heuristic-SS is the approach that consistently finds the best solution among heuristic methods. Furthermore, this quality does not always necessarily come at the expense of more computational time; SS is competitive in terms of average time in Figures 1a–1d, but is actually the fastest heuristic method on the final two figures corresponding to the largest instance.

In Figure 2, we again notice reasonable variability in TFD between the heuristic methods. The interesting comparison is between Heuristic-RG and Heuristic-SS. Despite Heuristic-SS consistently finding the best quality solution, it takes longer to reach this final design. On the other hand, Heuristic-RG achieves a much smaller TFD value compared to Heuristic-SS, sometimes sacrificing a little bit of solution quality in the process. The biggest drop-off in solution quality occurs in the **1**-alt version of Instance 14-7-35-1-0.2 (Figure 2e).

C. SAA Parameter Choices

For completeness, we include in Table 1 the results of experimenting with the value of $M = 20$ (and a maximum of 30 minutes of computational time for each sample problem), compared with $M = 10$ (and a maximum of 60 minutes). The experiments were conducted using the instance 14-7-35-1-0.2, and we analyze both the **1**-alt and the **2**-alt models. In the table, we present the total expected cost of the chosen design and the total computational time (in hours) taken by each.

Table 1 Comparison of $M = 10$ and $M = 20$ sample problems in SAA

# of Replications	**1**-alt		**2**-alt	
	Expected Total Cost	Total Time (h)	Expected Total Cost	Total Time (h)
M=10	1729.77	11.30	1606.01	12.00
M=20	1740.88	12.61	1606.64	14.24

We observe that, in both models, despite the considerable increase in computational time, the design chosen in the case of $M = 10$ had a lower total expected cost. Based on these results, we chose $M = 10$ for the experiments in Section 6.4.

To justify our choice of $N = 10$ and to give an indication of the quality of solutions we obtained as a result of the SAA process, we include in Table 2 the optimality gap estimates computed using Equation (5), and the percent optimality gap estimate (computed relative to the lower bound) for all 18 model-instance combinations.

Table 2 Optimality gap estimates

Instance	**1**-alt		**2**-alt		Infinite-alt	
	Opt. Gap	% Gap	Opt. Gap	% Gap	Opt. Gap	% Gap
12-5-20-1-0.2	6.10	0.72%	2.77	0.35%	1.48	0.19%
12-5-20-1-0.6	10.71	1.23%	6.42	0.79%	6.44	0.79%
12-5-35-1-0.2	10.09	0.82%	4.16	0.36%	2.70	0.23%
12-5-35-1-0.6	24.09	1.90%	11.32	0.94%	8.61	0.72%
14-7-35-1-0.2	20.14	1.17%	-3.65	-0.23%	-4.64	-0.29%
14-7-35-1-0.6	22.30	1.24%	-0.13	-0.01%	1.17	0.07%

In this table, we note that all percent optimality gaps were less than 2% for all model-instance combinations. In fact, the majority of values were less than 1%, especially for the **2**-alt and Infinite-alt models. Using these observed values, we deemed that our solutions were of reasonable quality using the SAA parameter choices we selected.