Supplementary Material for Parametric Stochastic Programming with One Chance Constraint: Gaining Insights from Response Space for Analysis

H.J. Greenberg[a], J-. P. Watson[b,*], D.L. Woodruff[c]

[a] *Deceased.*
[b] *Sandia National Laboratories, Albuquerque, New Mexico, USA*
[c] *University of California Davis, Davis, California, USA*

Abstract

In order to give examples of the calculations, we applied our methods to three models. Two of them are similar models for electricity generation expansion planning (GEP) and the third is a network flow, capacity-planning model. We refer to the models as Midwest GEP, Korean GEP, and Network Flow, respectively.

1. Empirical Anlaysis

1.1. Experimental Environment

The Midwest GEP model optimizes long-term generation capacity as described in Jin et al. (2011). We solve two instances, which use data from the Midwest Independent System Operator (MISO) with uncertain demand and fuel costs. The objective is to minimize expected total cost in a two-stage model where capacity decisions are made before scenarios are realized and generation expenses are based on the realization. The model includes the possibility that not all demand for electricity will be met and this so-called "load shedding" is penalized proportionally to the unmet demand. For

*Corresponding author
Email addresses: jwatson@sandia.gov (J-. P. Watson), dlwoodruff@ucdavis.edu (D.L. Woodruff)

the experiments reported here, we added a chance constraint that requires demand be met with probability at least β.

The Korean GEP model is described in Lee et al. (2012). Its basic structure is similar to Midwest GEP with the addition of an explicit service-level threshold that mandates electricity generation meet most of the demand for those scenarios, s, for which $\delta_s = 0$. The probability distributions for both GEP models were generated by a fit to historical data, which yields scenario probabilities.

The Network Flow model is based on a two-stage model created by RuszczynskiRuszczyński (2002). For testing purposes we extended it to include second-stage binary variables that allow for arc activation. Both arc activation costs and demands are stochastic. The model as used here is described in Watson et al. (2010), except here we use non-uniform scenario probabilities. However, the probabilities used were generated randomly, just to have a simple well-understood model for testing purposes. Unlike the GEP models, the Network Flow model has no penalty for unmet load service-level constraints.

In all three of these models, the δ-variables constrain the model as:

$$D \leq M(1 - \delta),$$

where D is the total unmet demand, defined in the model such that $0 \leq D < M$. Thus, $\delta = 0$ renders the constraint redundant, and $\delta = 1$ forces $D = 0$.

The three models are written in Pyomo. The stochastic elements are encoded using PySP, which also provides libraries used by the Python scripts for the algorithms described in this paper. Pyomo and PySP are available freely for download at http://pyomo.org.

All three models have integer variables in their basic structure, so adding δ does not change their theoretical complexity. For the 100-scenario GEP examples, the fully specified instances with explicit non-anticipativity constraints have tens of thousands of rows and columns, hundreds of thousands of nonzeros and thousands of integer variables.

For our experiments, we used a parameter for the minimum probability considered to be nonzero, which is $\tau^{\text{prob}} = 10^{-5}$. Eliminating low probability scenarios of the Korean GEP problems reported in Lee et al. (2012) left us with 40 scenarios. Property **??** fixes six scenarios, leaving only 34 of the 100 scenario-selection values to be determined. The 280-scenario instance found $\left|\{p_s \leq \tau^{\text{prob}}\}\right| = 160$, and we were able to use Property **??** to fix three

scenarios. These reductions, though based on very simple pre-processing, have a dramatic effect on computation time and, more importantly, offer insights to analysts as to why these scenarios were eliminated.

1.2. Response Space of Examples

Figure 1 shows the result of parametric search for a 10-scenario instance of the Midwest GEP model. This problem serves to demonstrate that the objective function may not be very sensitive to β. The difference between the minimum cost with no scenarios selected and that with all scenarios selected is 0.04%. (Note the scale: each cost is on the order of $19 billion, so a difference of $7.7 million is relatively small, making the chance constraint have relatively little impact.)

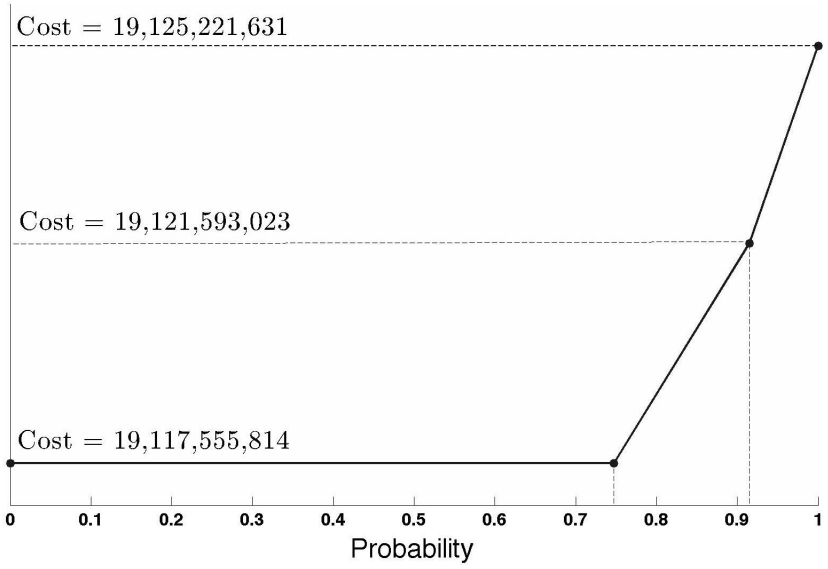

Figure 1: Envelope Function, F^*, of Midwest 10-Scenario Instance

Figure 2 shows the result of parametric search for a 10-scenario instance of the Network Flow problem. Although it might appear that the RS point (0.88, 118774) is on the line segment, as an alternative Lagrangian minimum, its segment is distinct with $\lambda^* = 43,521$. Using the solution values in Table 1, we can verify that (0.88, 118774) is generated as a new point in RS as follows:

$$\lambda = \frac{f^*(0.967) - f^*(0.747)}{0.967 - 0.747} = \frac{122,560 - 113,853}{0.22} = 39,577$$

$$\Rightarrow f^*(0.88) - \lambda\, 0.88 = 118,774 - 34,828 = 83,946 < f^*(0.967) - \lambda\, 0.967 = 84,289.$$

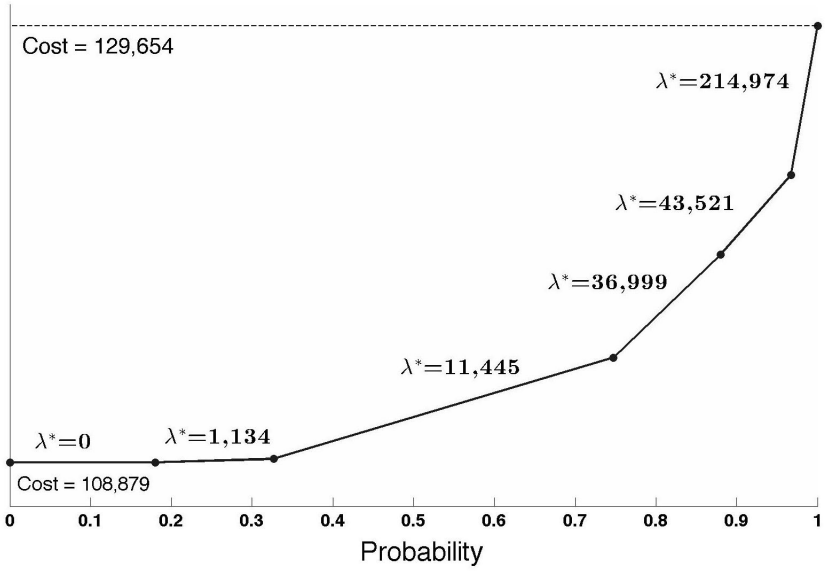

Figure 2: Envelope Function, F^*, of Network Flow 10-Scenario Instance

Table 1: Response Space Points with Optimal Multipliers for Network Flow 10-Scenario Instance

	λ^*	b	$f^*(b)$
1	0	0	108,879
2	0	0.18	108,879
3	1,134	0.327	109,046
4	11,445	0.747	113,853
5	36,999	0.880	118,774
6	43,524	0.967	122,560
7	214,974	1.000	129,654

As indicated in Table 2, it took 14 iterations to overcome the large λ_1 needed to ensure $b = 1$ is the Lagrange minimum. (As with tangential approximation, the actual initial solution, $(b_1 = 1, z_1)$ was obtained by $\delta \stackrel{\text{fix}}{=} 1$.) Once z_0 and z_1 are known, the value of λ_1 was set to scale: $\lambda_1 = (z_1 - z_0)/(1 - \tau^{\text{prob}})$, which ensures $(1, z_1)$ is the unique Lagrangian minimum, ignoring points that might be in the strip for $b \in (1 - \tau^{\text{prob}}, 1)$ since they are within the probability tolerance, τ^{prob}. From Table 1, we see $\lambda^* = 36999$ and $(b^L, b^U) = (0.747, 0.88)$. Bisection was able to terminate with $\lambda^* \in (36901, 37024)$. The width of this interval is 115, so the relative tolerance to declare convergence would have to be at least 115/37024, or 0.31%. The default relative tolerance is 10^{-5}, so that stopping condition was not met. We declare termination with a confirmed gap region if both endpoints are optimal solutions to the Lagrangian (within tolerance):

$$\max\left\{ \left| z^L - \lambda b^L \right|, \left| z^U - \lambda b^U \right| \right\} \leq \tau^{\text{opt}} L^*(\lambda).$$

Table 2: Bisection Iterations for $\beta = 0.85$

iteration	λ	b	z	elapsed time (sec)
0	0	0	108879	11
1	2077483200	1	129654	17
2	1038741600	1	135379	21
3	519370800	1	135379	26
4	259685400	1	135379	31
5	129842700	1	135379	36
6	64921350	1	135379	41
7	32460675	1	129654	47
8	16230338	1	129654	53
9	8115169	1	129654	59
10	4057584	1	129654	65
11	2028792	1	129654	72
12	1014396	1	129654	81
13	507198	1	129654	101
14	253599	1	129654	134
15	126800	0.967	122560	186
16	63400	0.967	122560	245
17	31700	0.747	113852	355
18	47550	0.967	122560	436
19	39625	0.880	118773	542
20	35662	0.747	113852	658
21	37644	0.880	118773	754
22	36653	0.747	113852	863
23	37148	0.880	118773	957
24	36901	0.747	113852	1072
25	37024	0.880	118773	1168

Table 3: Bisection Solutions for Network Flow 10-Scenario Instance

Specified	Terminal Values						Time	Number
β	λ^L	λ^U	b^L	b^U	z^L	z^U	(min)	iterations
0.96	39625	47550	0.880	0.967	118774	122560	12.0	19
0.93	39625	43587	0.880	0.967	118774	122560	12.1	20
0.90	39625	43587	0.880	0.967	118774	122560	12.3	20
0.85	36901	37024	0.747	0.880	113852	118773	19.5	25
0.80	36901	37024	0.747	0.880	113852	118773	12.0	25
0.60	11392	11454	0.327	0.747	109046	113852	21.9	26
0.40	11392	11454	0.327	0.747	109046	113852	22.8	26
0.20	1114	1238	0.180	0.327	108879	109046	11.5	25

Figure 3 is a visualization of results from the four highest target probabilities, which took about 56 minutes. The targets are displayed as the vertical lines, based at the best computed Lagrangian bound for multipliers, $\{\lambda_i\}_0^K$:

$$f^*(\beta) \geq \widehat{F^*}(\beta) \stackrel{\text{def}}{=} \max_{i=0,\ldots,K} \{L^*(\lambda_i)+\lambda_i\beta\} = \max\left\{L^*(\lambda^L)+\lambda^L\beta,\, L^*(\lambda^U)+\lambda^U\beta\right\}.$$

Tangential approximation obtains λ^*, so its bound is $L^*(\lambda^*) + \lambda^*\beta$, which cannot be less than any Lagrangian bound. Moreover, we can connect the dots in Figure 2 to show the full envelope function. We cannot do this using target probabilities; there could be other RS points below each line segment (though not the case here).

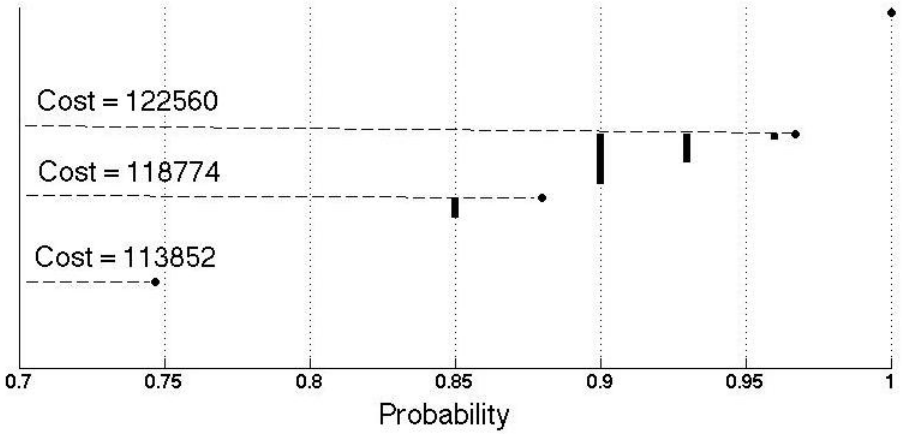

Figure 3: Visualization of Bisection Results of Four Target Probabilities (Vertical lines go from Bisection Bound to Cost of Best Feasible Solution)

Compare this with the decision support provided by Figure 2, which took about 12 minutes. Computing bisection solutions for all eight target β took more than two hours, and the highest four took nearly an hour, solving each β separately. Note that $(b = 0.967, z = 122560)$ was generated by several multipliers, as shown in Table 2. Because RS points are re-generated for different multiplier values, even solving just one β takes longer than the complete parametric tangential approximation (c.f., Table 3). For $\beta \in (b^L, b^U)$, bisection converges geometrically to the unique multiplier, $\lambda_k \to \lambda^* = (z^U - z^L)/(b^U - b^L)$. The multiplier sequence does get sufficiently close to the λ^* that it identifies the gap region, (b^L, b^U). Table 4 shows the final numerical values for each β.

Table 4: Bisection Approximations Compared with Tangential Approximation

	Bisection		Tangential Approximation	
β	$\widehat{F^*}(\beta)$	$\widehat{g}(\beta)$	$F^*(\beta)$	$g(\beta)$
0.20	108901	0.0013	108902	0.0013
0.40	109878	0.0349	109881	0.0349
0.60	112168	0.0148	112170	0.0148
0.80	115811	0.0249	115813	0.0249
0.85	117652	0.0094	117663	0.0093
0.90	119640	0.0238	119644	0.0238
0.93	120947	0.0132	120950	0.0131
0.96	122255	0.0025	122256	0.0025

When β is in a gap, λ^* is unique, and $\lambda_i \to \lambda^*$. The bisection bounds converge to the best Lagrangian bound, $\widehat{F^*}_i(\beta) \to F^*(\beta)$, which is the exact bound computed by tangential approximation. Thus, the approximate gap value from bisection converges: $\widehat{g}_i(\beta) \to g(\beta)$. As shown in Table 4, bisection provides nearly equal bounds and gap values for the targets. The real difference is the computing time. Our parametric tangential approximation algorithm computes the complete convex envelope is less time than it takes to solve just one β using bisection even if we discount the time to overcome λ_1 (i.e., reach $b = 0.967$). This speed empowers an analyst with more information, which can be visualized, as in Figure 2. For β not in a gap, bisection converges to one of the optimal multiplier values in the interval $[\lambda^L, \lambda^U]$, where the endpoints, computed exactly by tangential approximation, are the left and right derivatives of the envelope (**??**).

These rates do not appear in Figure 3, and other models may have $\lambda^U - \lambda^L$ so large (even relative to scale) that just knowing $\lambda^* \in (\lambda^L, \lambda^U)$ does not provide a visualization of variation between targets as do the extreme-λ values. In short, *parametric tangential approximation provides more information in less time.*

Using linear interpolation (another interval reduction method considered in Greenberg (1977)), the situation is worse because for high β values, termination is due to reaching a (default) limit on the number of iterations before getting close to λ^*.

The λ-sequence for $\beta = 0.85$, shown in Table 5, takes many more iterations than bisection. The situation is worse for the three highest values of

β, shown in Table 6, where it reached the maximum number of iterations permitted (100). For $\beta \in \{0.2, \ldots, 0.85\}$, linear interpolation generally takes more iterations than bisection, again reinforcing Property **??**. As shown in Table 5, the 78 iterations for $\beta = 0.85$ include 45 iterations to reach the search interval $(0, 0.967)$.

Table 5: Linear Interpolations Iterations for $\beta = 0.85$

iteration	λ	b	z	elapsed time (sec)
0	0	0	108879	16.0
1	2077483200	1	129654	27.4
⋮				⋮
57	231738	1	129654	624.1
58	196977	0.967	122560	648.0
59	173145	0.967	122560	673.4
60	152195	0.967	122560	695.0
61	133781	0.967	122560	720.2
62	117594	0.967	122560	746.9
63	103366	0.967	122560	776.8
64	90860	0.967	122560	806.4
65	79866	0.967	122560	834.8
66	70203	0.967	122560	871.3
67	61709	0.967	122560	904.9
68	54243	0.967	122560	945.5
69	47680	0.967	122560	987.4
70	41911	0.880	118774	1036.7
71	40482	0.880	118774	1086.2
72	39101	0.880	118774	1150.1
73	37769	0.880	118774	1204.1
74	36481	0.747	113853	1263.7
75	37478	0.880	118774	1318.0
76	37254	0.880	118774	1386.9
77	37079	0.880	118774	1438.2
78	36944	0.747	113853	1503.1

Table 6: Linear Interpolation Solutions for Network Flow 10-Scenario Instance

Specified	Terminal Values						Time	Number
β	λ^L	λ^U	b^L	b^U	z^L	z^U	(min)	iterations
0.96	39625	36508130	0.967	1	122560	129654	12.9	100*
0.93	0	1575250	0	1	108879	129654	16.4	100*
0.90	0	88686	0	0.967	108879	122560	21.1	100*
0.85	36944	37079	0.747	0.880	113852	118773	25.1	78
0.80	36998	37506	0.747	0.880	113852	118773	16.5	55
0.60	11392	11454	0.327	0.747	109046	113852	19.6	34
0.40	11425	11483	0.327	0.747	109046	113852	19.1	25
0.20	1119	1684	0.180	0.327	108879	109046	6.3	14

*Limit (default)

As shown in Table 7, the three highest values of β do not reach the gap region, so the bound is very coarse, giving a gap value exceeding tolerance. It also gives the full interval $[0, 1]$ as the final region of uncertainty, whereas the gap region is $(0.880, 0.967)$. Note the gross bound:

$$\widehat{F^*}(0.93) = 108879 < F^*(0.93) = 120590.$$

Table 7: Linear Interpolation Approximations Compared with Tangential Approximation

	Interpolation		Tangential Approximation	
β	$\widehat{F^*}(\beta)$	$\widehat{g}(\beta)$	$F^*(\beta)$	$g(\beta)$
0.20	108901	0.0013	108902	0.0013
0.40	109880	0.0349	109881	0.0349
0.60	112168	0.0148	112170	0.0148
0.80	115813	0.0249	115813	0.0249
0.85	117661	0.0094	117663	0.0093
0.90	116618	0.0485	119644	0.0238
0.93	108879	0.1602	120950	0.0131
0.96	117319	0.9514	122256	0.0025

Figure 4 shows the result of parametric search for a 50-scenario instance of the Network Flow model. One thing to notice is that the chance constraint

becomes binding at $\beta \approxeq 0.6684$, whereas this happens for the 10-scenario instance at $\beta \approxeq 0.18$. Intuitively, having more scenarios seems to provide low-cost selections that have a greater probability of satisfaction without an explicit constraint.

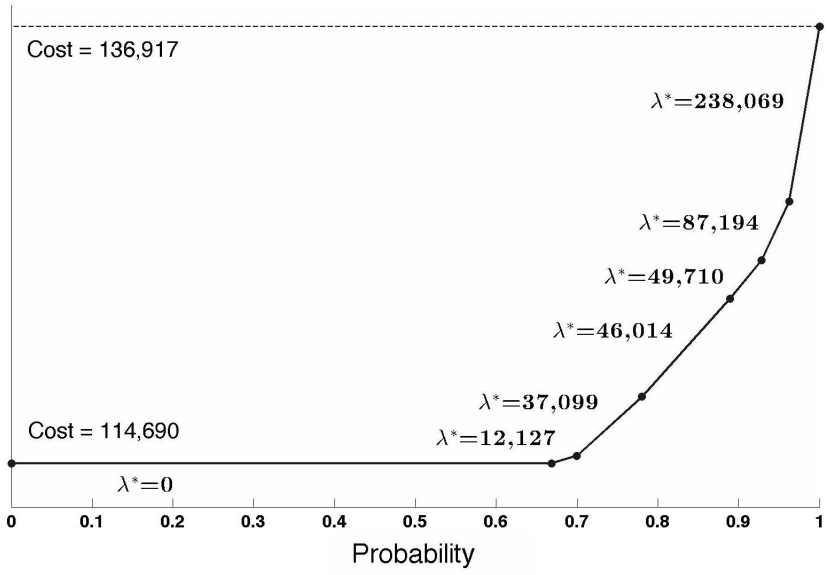

Figure 4: Envelope Function, F^*, of Network Flow 50-Scenario Instance

Figure 5 shows the result of parametric search for a 70-scenario Korean GEP instance. There is only one gap region, the entire $(0, 1)$ interval. All response space points are above (or on) that line segment, whose slope is $\lambda^* = z^U - z^L$. The relative gap for any $\beta > 0$ is $1 - \frac{z^L + \lambda^* \beta}{z^U} < 1 - \frac{z^L}{z^U} = 0.0127$. In the next section we consider closing the gap and will return to these chance-constraint instances.

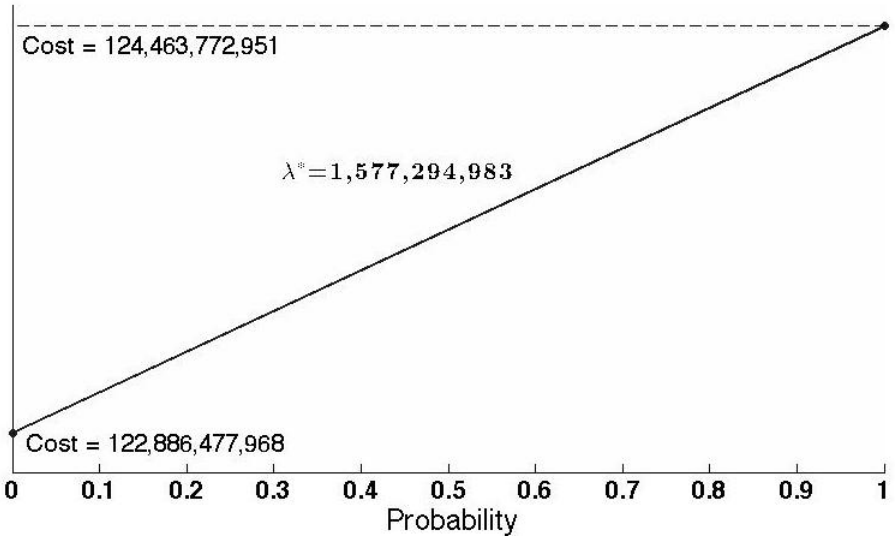

Figure 5: Envelope Function, F^*, of Korean 70-Scenario Instance

References

References

Greenberg, H. J., 1977. The one dimensional generalized Lagrange multiplier problem. Operations Research 25 (2), 338–345.

Jin, S., Ryan, S. M., Watson, J.-P., Woodruff, D. L., 2011. Modeling and solving a large-scale generation expansion planning problem under uncertainty. Energy Systems 2 (3–4), 209–242.

Lee, G.-C., Höhenrieder, M., Watson, J., Woodruff, D., 2012. Chance and service level constraints for stochastic generation expansion planning. Tech. rep., GSM, UC Davis, Davis CA 95616 USA.

Ruszczyński, A., 2002. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. Mathematical Programming 93 (2), 195–215.

Watson, J.-P., Wets, R. J.-B., Woodruff, D., 2010. Scalable heuristics for a class of chance-constrained stochastic programs. INFORMS Journal on Computing 22 (4), 543–554.
URL http://dx.doi.org/10.1287/ijoc.1090.0372