

# Solving Stochastic Optimization Problems on Computational Grids

Steve Wright

Argonne National Laboratory  
University of Chicago  
University of Wisconsin-Madison

Dundee, June 26, 2001.

## Stochastic Programming

Optimization of a model with **uncertainty**.

Often formulated mathematically as

$$\min_x f(x) \stackrel{\text{def}}{=} E_{\xi} g(x; \xi) = \int_{\Omega} g(x; \xi) p(\xi) d\xi,$$

( $p$  is probability density function) subject to constraints on  $x \in R^n$ .

Arises in planning-under-uncertainty applications, where each  $\xi$  represents a possible **scenario** (a possible way in which the model could evolve). Space  $\Omega$  can contain finite or infinite scenarios.

$g(x; \xi)$  could be the value function of some second level optimization problem parametrized by  $x$ . (*Recourse*.)

## Outline

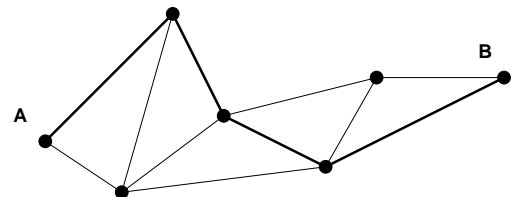
- Stochastic Programming (SP)
- Formulation and Basic Algorithms for SP
- Condor and metaNEOS
- Asynchronous Trust-Region Algorithm
- Computational Results: Algorithm Performance
- Sampling Methodology
- Computational Results: Quality of Solutions

Joint work with **Jeff Linderoth** (Axioma Inc) and **Alex Shapiro** (Georgia Tech).

## Example: Network Planning

Adding capacity on a telecommunications network for private-line services. (Sen et al., 1994.)

Shows nodes and links, Node pair A-B, and a route between A and B.



Add capacity to some links, to attempt to meet (uncertain) demand for traffic between nodes.

Sample demand profile: Third node pair:

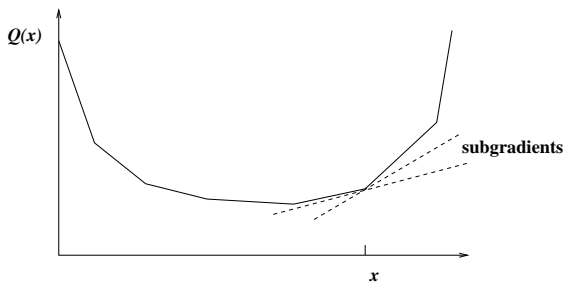
0	(prob .855)
5.39	(prob .095)
75.1	(prob .05)

- Data:
  - network topology:  $n = 89$  links.
  - point-to-point pairs:  $i = 1, 2, \dots, 86$ .
  - demands  $d_i$  for each pair  $i$  are random and independent, with 3 to 7 possible scenarios. *Total about  $10^{70}$  scenarios!*
- Decision variables:  $x_j, j = 1, 2, \dots, n$ : amount of capacity to add on link  $j$ . Total new capacity bounded by  $B$ .
- Objective: minimize the expected amount of unmet demand, summed over the  $m$  point-to-point pairs.

We can't hope to solve the problem by accounting for all the  $10^{70}$  possible scenarios exhaustively — it's much too large.

Practical approach is to use **sampling** to select a subset of  $N$  scenarios, randomly. The **sample average approximation** (SAA) is still large, but manageable.

Each  $Q(x; \omega_j)$  is convex, piecewise-linear in  $x$ .



Compute subgradients of  $Q$  by

- finding *dual* solutions  $\pi_j$  of the second-stage LP's for  $j = 1, 2, \dots, N$  (concurrently!)

$$Q(x; \omega_j) : \min_{y_j} q(\omega_j)^T y_j, \text{ subj. to } Wy = h(\omega_j) - T(\omega_j)x, y_j \geq 0;$$

- summing:

$$c - N^{-1} \sum_{j=1}^N T(\omega_j)^T \pi_j.$$

## 2-stage stochastic LP with recourse

$$\min_x Q(x) = c^T x + E_P Q(x; \omega) \\ \text{subj. to } Ax = b, x \geq 0,$$

where  $P$  is a probability measure on the space  $(\Omega, \mathcal{F})$ , and

$$Q(x; \omega) = \min_y q(\omega)^T y \text{ subject to } Wy = h(\omega) - T(\omega)x, y \geq 0.$$

$x$  = first-stage vars,  $y$  = second-stage vars.

Sampled approximation: Sample  $N$  points  $\omega_j, j = 1, 2, \dots, N$  from  $P$ , and solve

$$\min_x Q(x) = c^T x + N^{-1} \sum_{j=1}^N Q(x; \omega_j) \\ \text{subj. to } Ax = b, x \geq 0.$$

## "bundle" methods

Build up a lower bounding, piecewise linear approximation to  $Q(x)$ , based on function values  $Q(x^\ell)$  and subgradients  $g^\ell$  at iterates  $x^\ell$ .

Model function  $M_k(x)$  after  $k$  iterates is

$$M_k(x) = \sup_{\ell=0,1,\dots,k} [Q(x^\ell) + (g^\ell)^T (x - x^\ell)].$$

Choose next iterate as

$$x^{k+1} = \arg \min M_k(x), \text{ subj. to } Ax = b, x \geq 0.$$

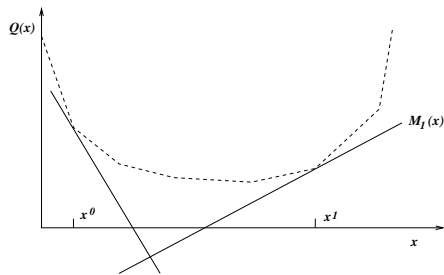
which can be formulated as:

$$\min_{x, \theta} \theta, \text{ subject to } Ax = b, x \geq 0,$$

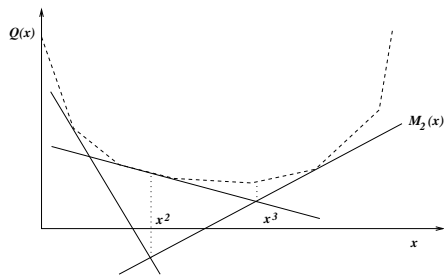
$$\theta \geq Q(x^\ell) + (g^\ell)^T (x - x^\ell), \ell = 0, 1, \dots, k.$$

(Each constraint is called a **cut**.)

Example: After first two iterations 0, 1:



$x^2$  is the minimizer of  $M_1$ ; add new subgradient to obtain  $M_2$ ; take minimizer to obtain  $x^3$ :



## trust-region (TR)

Choose next iterate as

$$x^{k+} = \arg \min M_k(x), \text{ subj. to } Ax = b, x \geq 0, \|x - x^k\|_\infty \leq \Delta_k,$$

where  $\Delta_k$  is the trust-region radius.

- Trivial to modify the LP subproblem: just add the bounds

$$-\Delta_k e \leq x - x^k \leq \Delta_k e.$$

- If candidate point  $x^{k+}$  is “significantly better” (achieves some fraction of the decrease predicted by the model) then set  $x^{k+1} \leftarrow x^{k+}$ . Possibly delete cuts, increase the trust region.
- Otherwise, set  $x^{k+1} \leftarrow x^k$  and add subgradient information from  $x^{k+}$  to improve the model. Possibly delete uninteresting cuts, decrease trust region.

## enhancements

- trust-region (allows more steady progress, exploits good starting point);
- algorithm that allows deletion of old cuts;
- group the second-stage problems  $Q(x; \omega_j)$  into  $T$  “chunks”  $\mathcal{N}_t, t = 1, 2, \dots, T$ , with
 
$$\{1, 2, \dots, N\} = \cup_{t=1,2,\dots,T} \mathcal{N}_t.$$
 and assign each  $\mathcal{N}_t$  to a worker processor;
- multiple cuts at each  $x$  (each chunk can return its own subgradients);
- **asynchronous** variant is preferred for our target parallel platform.

## TR properties

Denoting the solution set by  $\mathcal{S}$ ,

- Can delete cuts liberally, between major iterations;
- $\text{dist}(x^k, \mathcal{S}) \rightarrow 0$ ;

The algorithm may still be too synchronous: requires complete evaluation of  $Q(x)$  at a candidate iterate  $x$  before proceeding.

## related work

---

- Marsten et al “box step” (1975) builds up an exact model of the problem over the TR.
- Lemaréchal “bundle” (1975 et seq.)
- Kiwiel (1983 et seq.)
- Ruszczyński “regularized decomposition” (1986) specifically for stochastic programming.

The last three give quadratic programming sub-problems, not LP. More like a Levenberg approach than a trust-region approach.

## a challenging environment...

---

The Condor environment is powerful and inexpensive, but challenging to algorithm designers and implementers.

- *dynamic/opportunistic*: size and composition of worker pool changes unpredictably during computation
- *heterogeneous*: old workstations and fast new linux machines, different versions of Solaris, software licenses valid on only some machines.
- *latency unpredictable, generally slow*: workers can be next to each other in a rack, or separated by 6000 miles and the Internet.

*Problems that are large and compute-intensive — and algorithms that are **asynchronous** — work best on this platform.*

## Condor!

---

**High-throughput computing on pools of distributively owned computers** — Developed at Wisconsin: Livny.

- Condor pools consist of user workstations, nodes from multiprocessor systems and clusters.
- Handles scheduling, matching of user requirements to machine characteristics.
- Checkpointing and migration.
- Flocking and Glide-in mechanisms allow jobs to execute across multiple pools.

## MW (Master-Worker)

---

Many interesting optimization algorithms can be shoehorned into the master-worker paradigm.

MW is a runtime support library for implementing master-worker computations on the Condor system. (Yoder, Kulkarni, Linderoth, Goux)

MW abstracts the issues of resource management and communication.

- Condor handles resource management;
- Communication is either via shared files or Condor-PVM.

## TR may not be asynchronous enough!

---

The TR approach still synchronizes on the function evaluation at each candidate point  $x^{k+}$ .

If there are  $T$  chunks of second-stage scenarios, can't use more than  $T$  processors. For many problems of interest, we cannot make  $T$  very large (10 – 100) without making the work-per-chunk too small and creating too much contention at the master.

May wait for a long time for the last chunk to be evaluated, if its host is suspended or disappears.

An *asynchronous trust-region* (ATR) algorithm increases parallelism and throughput.

## ATR

---

- Maintain an *incumbent*  $x^I$ : the best point found so far (smallest value of  $Q$ ).
- Maintain a *basket*  $\mathcal{B}$  of 3–20 other  $x$  points — possible new incumbents — for which the second-stage LPs are currently being solved.
- When space becomes available in  $\mathcal{B}$ , generate a new candidate point by solving a TR subproblem around the current incumbent:  $\|x - x^I\|_\infty \leq \Delta$ . ( $x^I$  becomes the *parent incumbent* of the new point.)

## ATR (continued)

---

- When evaluation of a point  $x \in \mathcal{B}$  is completed, accept it as the new incumbent if
  - $Q(x) < Q(x^I)$ ; and
  - $Q(x)$  gives a significant decrease over its parent incumbent.
- Populate  $\mathcal{B}$  initially by solving TR subproblems around early incumbents, using partial subgradient information. (Synchronicity parameter  $\sigma$ .)
- Strategies for cut deletion and adjustment of trust region are adapted from the strategies for the synchronous TR algorithm.

## convergence of ATR

---

Because of the strategy for selecting incumbents, and for cut management and adjustment of  $\Delta$ , can re-use much of the theory for the serial case.

From a given incumbent, can trace back a chain through successive parents, to the initial point. A sufficient decrease condition is satisfied with respect to each link in this chain.

By applying synchronous TR theory to this chain, and assuming that all tasks terminate finitely, we have  $\text{dist}(x^I, S) \rightarrow 0$ .

## SSN: computational results

- first stage: 89 variables, 1 constraint;
- second stage: 706 variables, 175 constraints, 2284 nonzeros.

Study the effect of asynchronicity, parallelism on large sampled instances.

We report results for  $N = 10^4$  and  $N = 10^5$  scenarios, with synch parameter  $\sigma = .7$ .

run	iter	chunks	cuts/iter	av. procs	par. efficiency	wall clock (min)	
ATR	81	3	25	100	43	.38	64
ATR	81	3	25	100	39	.41	64
ATR	87	3	25	100	36	.44	66
ATR	106	3	50	100	84	.28	53
ATR	95	3	50	100	65	.26	64
ATR	94	3	50	100	23	.44	105
ATR	171	6	25	100	70	.45	61
ATR	135	6	25	100	61	.39	75
ATR	145	6	25	100	38	.35	146
ATR	177	6	50	100	87	.41	54
ATR	162	6	50	100	93	.34	66
ATR	159	6	50	100	39	.27	199
SSN, $N = 10,000$ . $1.75M \times 7.06M$ .							

run	iter	chunks	cuts/iter	av. procs	par. efficiency	wall clock (min)	
ATR	177	3	100	100	38	.52	1357
SSN, $N = 100,000$ . $17.5M \times 70.6M$ .							

run	iter	chunks	cuts/iter	av. procs	par. efficiency	wall clock (min)
ALS	282	50	50	26	.87	254
TR	47	25	100	23	.49	58
TR	44	25	100	21	.31	97
TR	45	25	100	20	.23	158
TR	51	50	100	37	.33	48
TR	51	50	100	45	.14	135
TR	46	50	100	41	.15	135
SSN, $N = 10000$ . $1.75M \times 7.06M$ .						

## storm: computational results

Cargo flight scheduling problem (Mulvey and Ruszczyński).

- first stage: 121 variables;
- second stage: 1259 variables.

For a 250000 scenario sampled approx, LP has size

$$132,000,185 \times 314,750,121$$

Started from a solution for a 3000-scenario approximation, whose quality is very good. TR takes a single step and terminates, ATR doesn't take any steps, just verifies quality of starting point.

(For a chunk of 2000 scenarios, task size is about 150 seconds.)

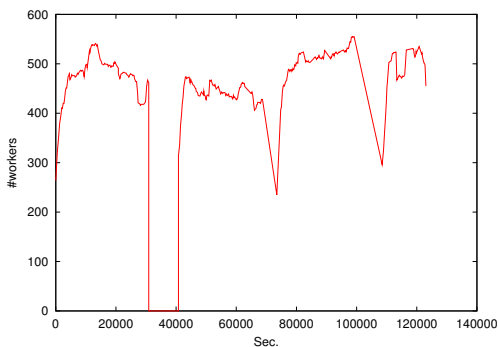
run	iter	s	chunks	cuts/iter	av. procs	par. efficiency	wall clock (min)
TR	17	-	125	125	106	.55	146
ATR	25	3	125	125	106	.90	116
storm, $N = 250,000$ . $132M \times 315M$ .							

## storm with $10^7$ scenarios

LP has size approximately

$$5.5 \times 10^9 \text{ rows, } 1.3 \times 10^{10} \text{ columns.}$$

Used machines at Wisconsin, NCSA (Illinois), New Mexico, Argonne, Italy, Columbia. 800 machines requested, 556 actually used during the run (average of 433 at any one time).



run	iter	s	chunks	cuts/iter	av. procs	par. efficiency	wall clock (hrs)
ATR	39	4	1024	1024	433	.67	31.9
storm, $N = 10^7$ . Columns: $1.3 \times 10^{10}$ .							

Solved  $4 \times 10^8$  second-stage linear programs during the run (3472 per second). Average task 774 seconds.

Total computation time 9014 hours (more than one year).

## solution quality

---

- Can we get useful estimates for the optimal objective values of the true problem from the sampled problem? Can we get confidence intervals?
- How do the *solutions* of the sampled approximation relate to those of the real problem?

Using ATR, along with relevant theory (some recent), we have performed computational and statistical studies of these issues for some difficult problems from the literature.

## lower bound for $Z^*$

---

It's well known that

$$EZ_N \leq Z^*.$$

This is true for any unbiased estimator. In particular can use certain variance reduction techniques (e.g. Latin hypercube) to select the sample  $\{\omega_1, \omega_2, \dots, \omega_N\}$ .

Generate  $M$  batches — each a sampled approximation of size  $N$  of the form  $\{\omega_1^{(i)}, \omega_2^{(i)}, \dots, \omega_N^{(i)}\}$ ,  $i = 1, 2, \dots, M$  — and solve the  $M$  SAAs to obtain optimal values

$$Z_N^{(1)}, Z_N^{(2)}, \dots, Z_N^{(M)}.$$

Then estimate  $EZ_N$  by

$$L_M = M^{-1} \sum_{i=1}^M Z_N^{(i)}.$$

## notation reminder

---

$$\min_{Ax=b, x \geq 0} Q(x) = c^T x + \sum_{i=1}^K p_i Q(x; \omega^i),$$

(where  $K \approx 10^{70}$ , say) while sample average approximation (SAA) can be formed by sampling  $N$  points  $\{\omega_1, \omega_2, \dots, \omega_N\}$  from the distribution  $P$  and solving

$$\min_{Ax=b, x \geq 0} Q_N(x) = c^T x + N^{-1} \sum_{j=1}^N Q(x; \omega_j).$$

Denote

$$Z^* = \min_{Ax=b, x \geq 0} Q(x), \quad Z_N = \min_{Ax=b, x \geq 0} Q_N(x).$$

Can use Monte Carlo sampling (sampling with replacement). Can also use variance reduction techniques to reduce  $\text{Var}(Z_N)$ . We implemented Latin hypercube sampling (sampling without replacement).

## confidence interval for lower bound

---

Have in the limit that

$$\sqrt{M} [L_M - EZ_N] \sim N(0, \sigma_L^2)$$

where

$$\sigma_L^2 = \text{Var} Z_N.$$

Can approximate  $\sigma_L^2$  by the sample variance estimator

$$s_L(M)^2 = \frac{1}{M-1} \sum_{i=1}^M \left[ Z_N^{(i)} - L_M \right]^2$$

Then defining  $z_\alpha$  such that

$$P\{N(0, 1) \leq z_\alpha\} = 1 - \alpha,$$

our estimate of the width of the  $(1 - 2\alpha)$  confidence interval is

$$z_\alpha s_L(M) / \sqrt{M}.$$

(For  $\alpha = .025$  have  $z_\alpha \approx 1.96$ .)

## upper bound for $Z^*$

Given any feasible point  $\hat{x}$ , we have

$$Q(\hat{x}) \geq Q(x^*).$$

Choose an  $\hat{x}$  that appears to be nearly optimal, e.g. minimizer of some  $Q_N$ .

Choose  $T$  i.i.d. samples, each of size  $\bar{N}$  (using Monte Carlo or Latin Hypercube):

$$\{\bar{\omega}_1^{(i)}, \bar{\omega}_2^{(i)}, \dots, \bar{\omega}_{\bar{N}}^{(i)}\}, \quad i = 1, 2, \dots, T,$$

Defining

$$\hat{Q}_{\bar{N}}^{(i)}(\hat{x}) = c^T \hat{x} + \bar{N}^{-1} \sum_{j=1}^{\bar{N}} Q(\hat{x}; \bar{\omega}_j^{(i)}).$$

we get an unbiased estimator:

$$U_{\bar{N}, T} = T^{-1} \sum_{i=1}^T \hat{Q}_{\bar{N}}^{(i)}(\hat{x}).$$

## test problems

Performed experiments with five problems from the literature.

Solved SAA's for sample sizes  $N$  ranging from 50 to 5000.

Solved between 9 and 12 SAAs ( $M$ ) for each value of  $N$ .

In upper-bound evaluation, for each optimizing  $\hat{x}$  from the SAAs,  $T = 50$  and  $\bar{N} = 20,000$ . Report the value of  $\hat{x}$  for which the estimate  $U_{\bar{N}, T}$  is lowest, together with its confidence interval.

In selecting samples of size  $N$  (for SAA) and  $\bar{N}$  (for evaluation), used both Monte Carlo and Latin Hypercube distribution, as reported in the tables.

## confidence interval for upper bound

Since the batches are i.i.d. we have

$$\sqrt{T} [U_{\bar{N}, T} - Q(\hat{x})] \sim N(0, \sigma_U^2)$$

where  $\sigma_U^2 = \text{Var} \hat{Q}_{\bar{N}}(\hat{x})$ . Approximate  $\sigma_U^2$  by sample variance estimator

$$s_U(T)^2 = \frac{1}{T-1} \sum_{i=1}^T \left[ \hat{Q}_{\bar{N}}^{(i)}(\hat{x}) - U_{\bar{N}, T} \right]^2.$$

Get confidence interval width

$$z_\alpha s_U(T) / \sqrt{T}.$$

name	application	scenarios	stage 1	stage 2
Lands	hydro power	$10^6$	$2 \times 4$	$7 \times 12$
gbd		$6.5 \times 10^5$		
storm	cargo flights	$6 \times 10^{81}$	$185 \times 121$	$528 \times 1291$
20term	vehicle assignment	$1.1 \times 10^{12}$	$1 \times 5$	$71 \times 102$
ssn	network design	$10^{70}$	$1 \times 89$	$175 \times 706$

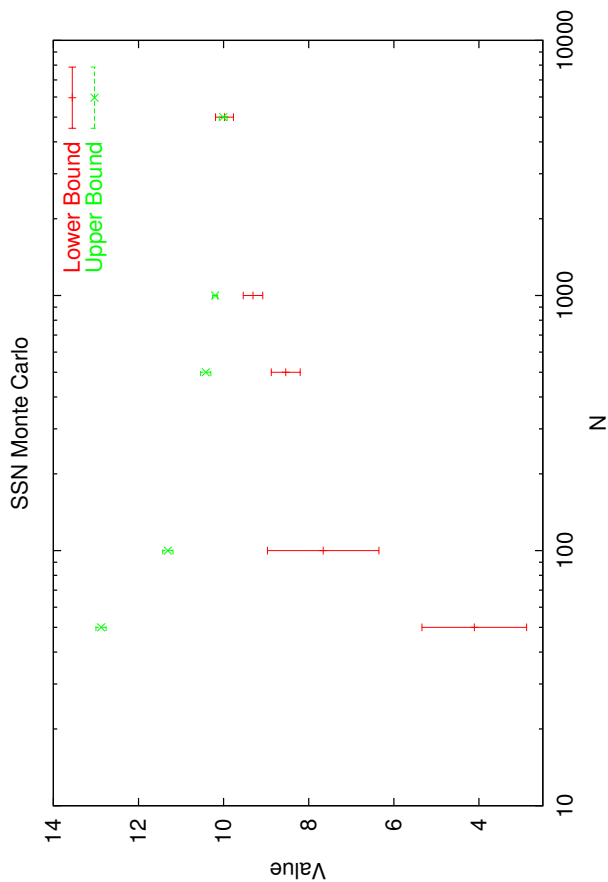
## results on bounds: SSN

Results of Mak et al (1999).

	lower	upper
batch/sample	$30 \times 1000$	$1 \times 100000$
estimate	9.22	9.98
95% confidence	$\pm 0.21$	$\pm 0.11$

Using different techniques, Mak et al, generate an approximate solution  $\hat{x}$  using  $N = 2000$ , and obtain

- upper bound  $10.06 \pm 0.12$  (95% confidence interval);
- with 95% likelihood, the optimal  $Z^*$  is within 0.77 of this value.



## solution estimates for SSN

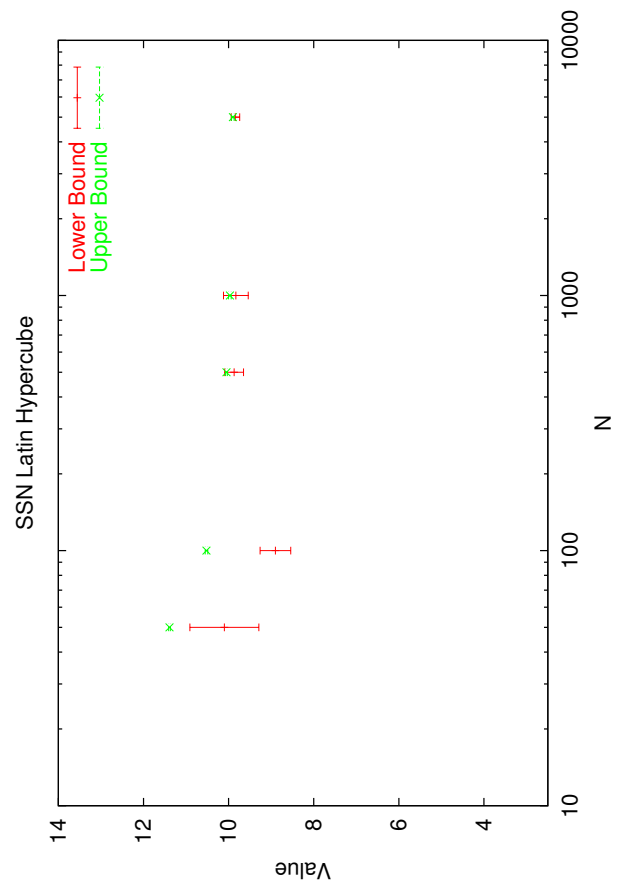
95% confidence intervals.

Monte Carlo:

N	Lower	Upper
50	$4.11 \pm 1.23$	$12.88 \pm 0.12$
100	$7.66 \pm 1.31$	$11.31 \pm 0.12$
500	$8.54 \pm 0.34$	$10.42 \pm 0.12$
1000	$9.31 \pm 0.23$	$10.20 \pm 0.06$
5000	$9.98 \pm 0.21$	$10.01 \pm 0.09$

Latin Hypercube:

N	Lower	Upper
50	$10.10 \pm 0.81$	$11.39 \pm 0.02$
100	$8.90 \pm 0.36$	$10.52 \pm 0.03$
500	$9.87 \pm 0.22$	$10.05 \pm 0.02$
1000	$9.83 \pm 0.29$	$9.97 \pm 0.03$
5000	$9.84 \pm 0.10$	$9.90 \pm 0.03$



### solution estimates for gbd

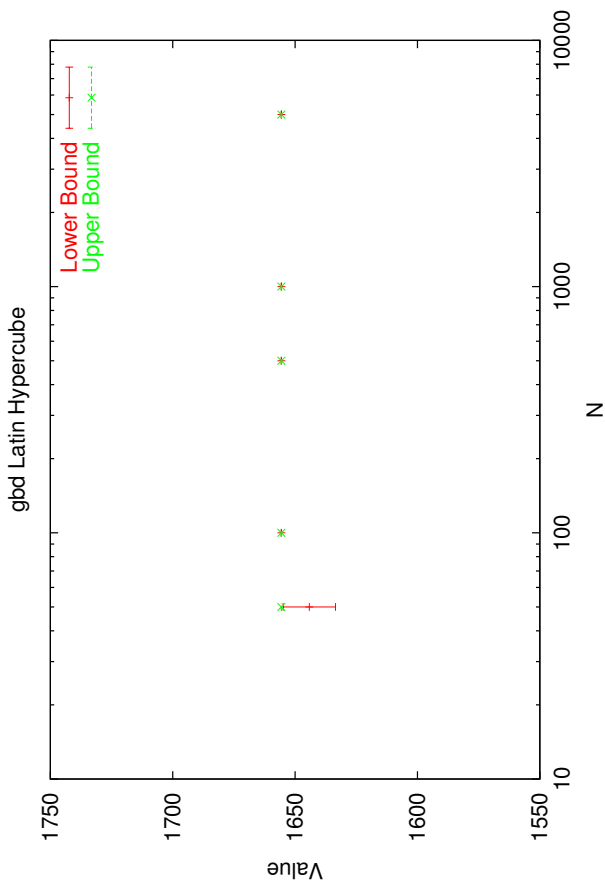
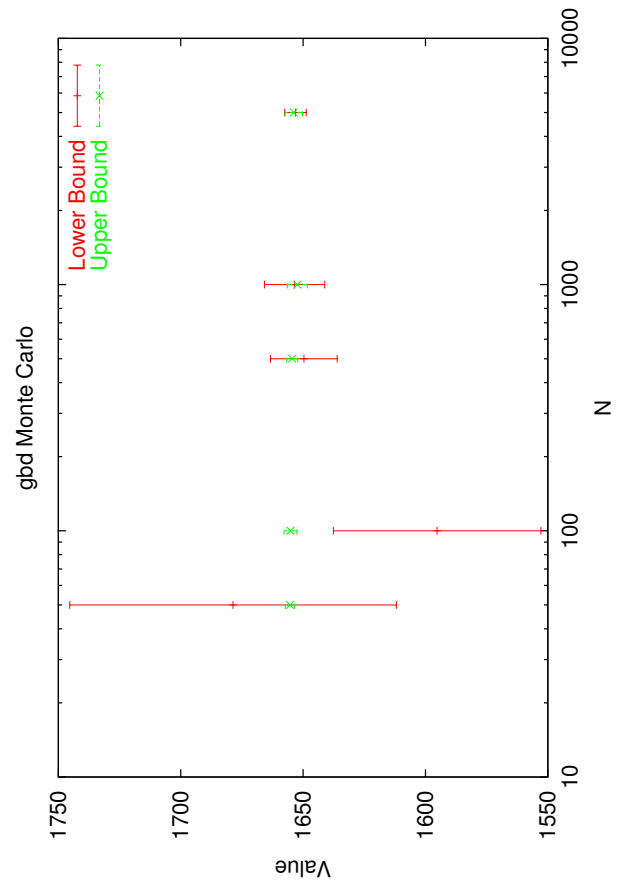
95% confidence intervals.

Monte Carlo:

N	Lower	Upper
50	1678.62 ± 66.73	1655.33 ± 1.93
100	1595.24 ± 42.41	1655.13 ± 2.64
500	1649.66 ± 13.60	1654.42 ± 2.27
1000	1653.50 ± 12.32	1652.35 ± 4.09
5000	1653.13 ± 4.37	1653.81 ± 3.56

Latin Hypercube:

N	Lower	Upper
50	1644.21 ± 10.71	1655.62 ± 0.00
100	1655.62 ± 0.00	1655.62 ± 0.00
500	1655.62 ± 0.00	1655.62 ± 0.00
1000	1655.62 ± 0.00	1655.62 ± 0.00
5000	1655.62 ± 0.00	1655.62 ± 0.00



### solution estimates for storm

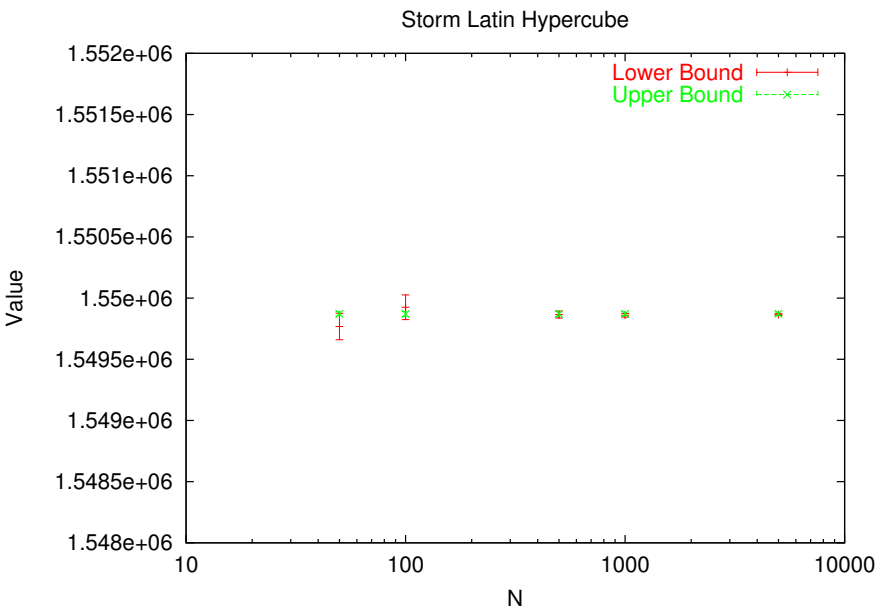
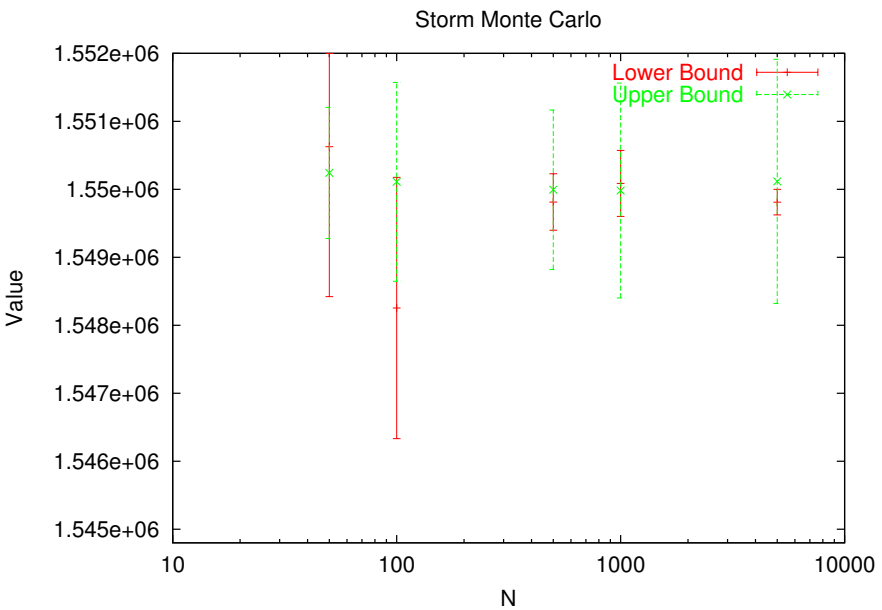
95% confidence intervals.

Monte Carlo:

N	Lower	Upper
50	1550626.75 ± 2204.35	1550241.00 ± 964.33
100	1548254.58 ± 1921.34	1550110.00 ± 1462.08
500	1549813.58 ± 415.30	1549995.00 ± 1172.70
1000	1550087.42 ± 484.79	1549982.00 ± 1581.07
5000	1549811.67 ± 188.01	1550117.00 ± 1795.66

Latin Hypercube:

N	Lower	Upper
50	1549767.90 ± 107.94	1549872.00 ± 17.05
100	1549924.90 ± 101.09	1549871.00 ± 20.10
500	1549865.60 ± 28.18	1549871.00 ± 22.71
1000	1549859.20 ± 14.90	1549871.00 ± 17.30
5000	1549865.10 ± 7.46	1549872.00 ± 15.04



## solution estimates for 20term

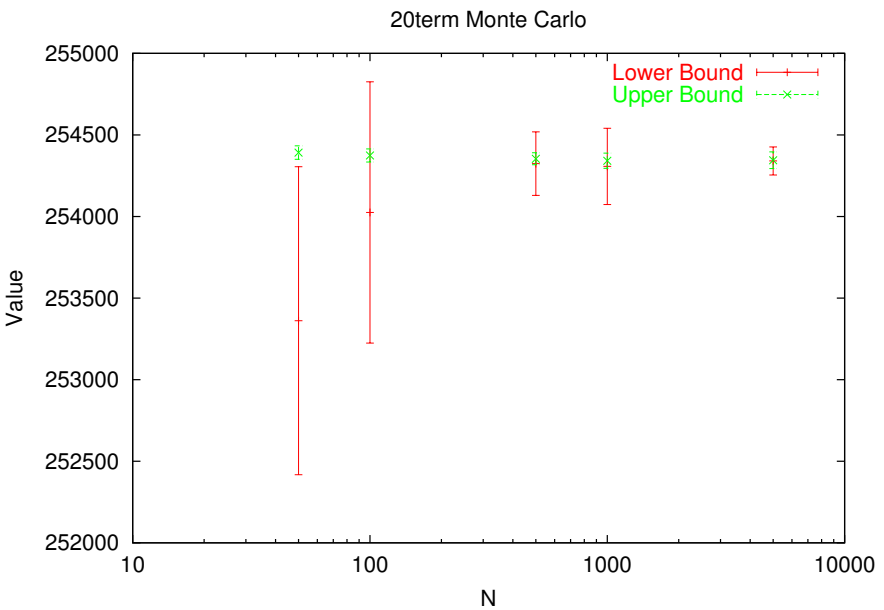
95% confidence intervals.

Monte Carlo:

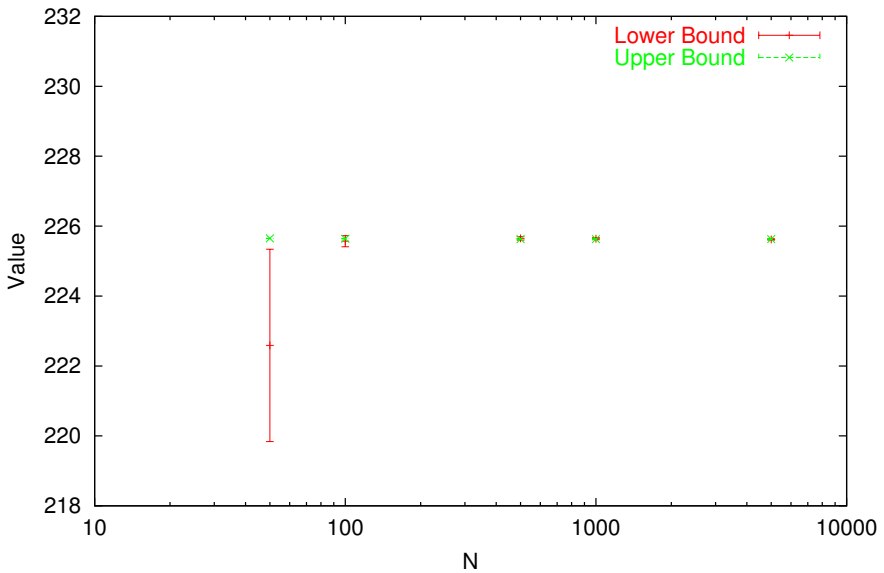
N	Lower		Upper	
	50	253361.33 ± 944.06	254392.00 ± 41.55	254392.00 ± 41.55
100	254024.89 ± 800.88	254374.00 ± 40.11	254374.00 ± 40.11	254354.00 ± 36.55
500	254324.33 ± 194.51	254354.00 ± 36.55	254354.00 ± 36.55	254342.00 ± 46.60
1000	254307.22 ± 234.04	254342.00 ± 46.60	254342.00 ± 46.60	254345.00 ± 51.05
5000	254340.78 ± 85.99	254345.00 ± 51.05	254345.00 ± 51.05	254345.00 ± 51.05

Latin Hypercube:

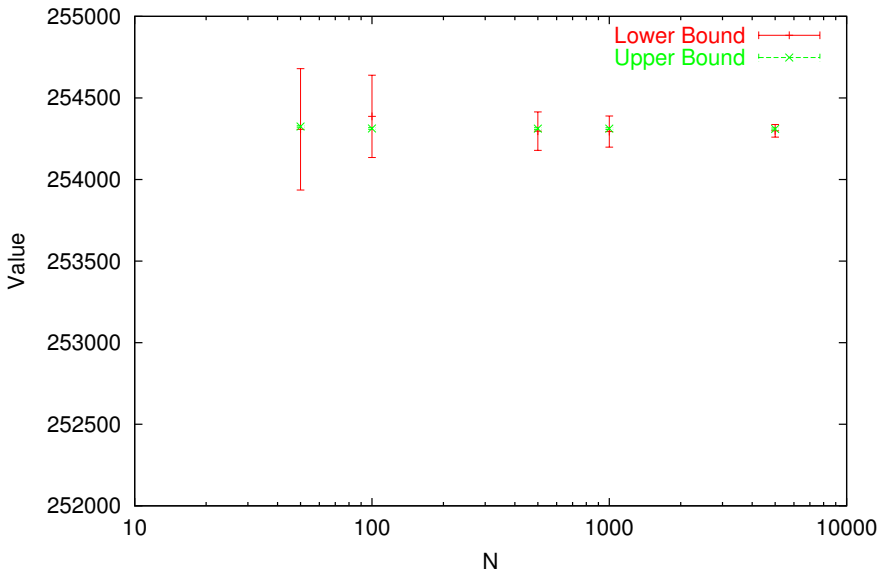
N	Lower		Upper	
	50	254307.57 ± 371.80	254325.00 ± 4.58	254325.00 ± 4.58
100	254387.00 ± 252.13	254313.00 ± 5.43	254313.00 ± 5.43	254296.43 ± 117.95
500	254296.43 ± 117.95	254312.00 ± 5.58	254312.00 ± 5.58	254294.00 ± 95.22
1000	254294.00 ± 95.22	254312.00 ± 5.14	254312.00 ± 5.14	254298.57 ± 38.74
5000	254298.57 ± 38.74	254310.00 ± 5.80	254310.00 ± 5.80	254298.57 ± 38.74



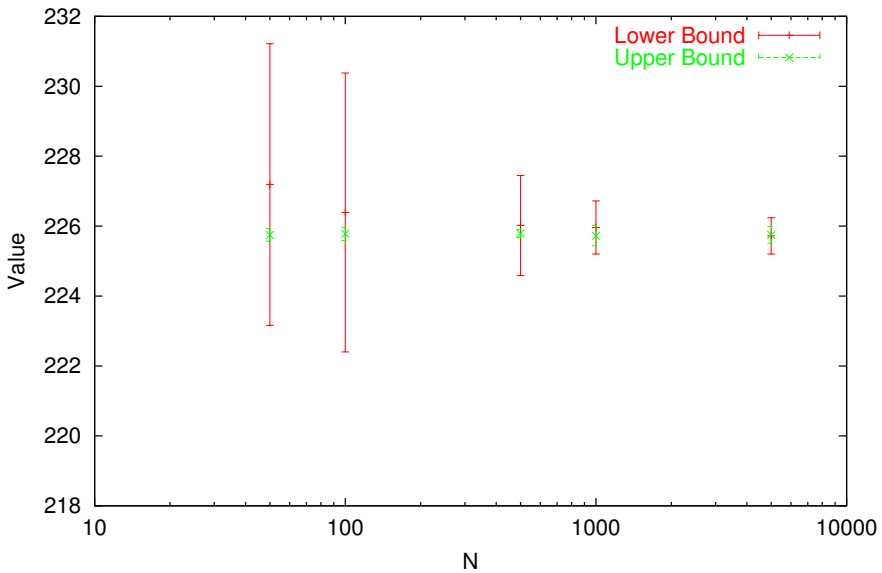
LandS Latin Hypercube



20term Latin Hypercube



LandS Monte Carlo



### conditioning and exact solutions

Recent results (Shapiro and Homem-de-Mello, 2000) indicate that a discrete SAA can identify the exact solution of a stochastic linear program over a discrete probability space. If solution is unique, chance of identifying it exactly approaches 1 exponentially rapidly in  $N$ :

$$P(\hat{x} = x^*) \geq 1 - e^{-\beta N}, \text{ some } \beta > 0.$$

We investigated the solutions obtained for the finest SAAs ( $N=5000$ ) for each problem instance, using Latin Hypercube sampling. We plotted distance matrices for six SAA solutions for each problem.

**distance of SSN solutions**

0.00	396.72	176.90	481.92	266.11	477.05
396.72	0.00	465.13	743.21	528.69	326.39
176.90	465.13	0.00	501.36	381.06	495.92
481.92	743.21	501.36	0.00	698.67	934.41
266.11	528.69	381.06	698.67	0.00	712.62
477.05	326.39	495.92	934.41	712.62	0.00

Solutions are far apart, though their objective values appear to be similar. Indicates a shallow minimum.

**distance of gbd solutions**

0.00e+00	2.49e-26	1.42e-27	6.68e-27	4.10e-28	1.47e-27
2.49e-26	0.00e+00	1.61e-26	6.72e-27	2.64e-26	3.22e-26
1.42e-27	1.61e-26	0.00e+00	3.17e-27	1.73e-27	2.96e-27
6.68e-27	6.72e-27	3.17e-27	0.00e+00	8.30e-27	1.19e-26
4.10e-28	2.64e-26	1.73e-27	8.30e-27	0.00e+00	8.17e-28
1.47e-27	3.22e-26	2.96e-27	1.19e-26	8.17e-28	0.00e+00

Sharp, well defined minimizer.

**distance of 20term solutions**

0.00	259.36	700.39	87504.49	77043.47	68975.66
259.36	0.00	413.07	88080.16	77726.57	69723.88
700.39	413.07	0.00	84761.87	74631.07	67052.76
87504.49	88080.16	84761.87	0.00	2419.97	4485.82
77043.47	77726.57	74631.07	2419.97	0.00	1017.77
68975.66	69723.88	67052.76	4485.82	1017.77	0.00

**distance of storm solutions**

0.00e+00	3.51e-04	1.34e-05	1.11e-04	4.79e-05	5.27e-05
3.51e-04	0.00e+00	2.35e-04	8.94e-05	1.54e-04	1.47e-04
1.34e-05	2.35e-04	0.00e+00	4.73e-05	1.07e-05	1.30e-05
1.11e-04	8.94e-05	4.73e-05	0.00e+00	1.31e-05	1.08e-05
4.79e-05	1.54e-04	1.07e-05	1.31e-05	0.00e+00	1.12e-07
5.27e-05	1.47e-04	1.30e-05	1.08e-05	1.12e-07	0.00e+00

Also a well defined minimizer.

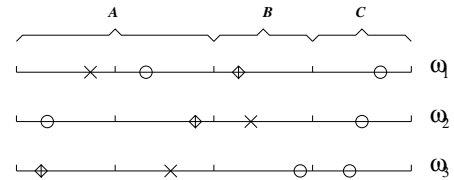
## Latin hypercube sampling

---

Aim to reduce variance in  $Z_N$ .

Example: Scenario space  $\omega_1 \times \omega_2 \times \omega_3$ , where each  $\omega$  is distributed according to:

$$P(\omega = A) = .5, \quad P(\omega = B) = .25, \quad P(\omega = C) = .25.$$



Sample size  $N = 4$ . Divide  $[0, 1]$  into 4 intervals, allow exactly one sample in each interval for each random variable.

	1	2	3	4
$\omega_1$	B	C	A	A
$\omega_2$	A	A	B	C
$\omega_3$	A	C	A	B

The following slides will not be used in this talk but are retained in the file for reference.

We use Latin hypercube sampling, larger sample sizes.

- Lower bound: 39 batches of size  $N = 5000$ .  
Obtained

$$L_M = 9.9163, \quad \text{standard error} = .0273.$$

95% confidence interval is [9.8610, 9.9716].

- Upper bound: For each of the SA solutions  $\hat{x}$  obtained from the lower bound calculation, took 21 batches of size  $P = 20,000$ . Used these to estimate  $U_P(\hat{x})$  for each  $\hat{x}$ , together with its std error. For the "best"  $\hat{x}$ , obtained

$$9.9001, \quad \text{standard error} .0190.$$

95% confidence interval is [9.8614, 9.9397].

Suggests strongly that the optimal  $Z^*$  is close to 9.91.