Stochastic Optimization on Computational Grids

Steve Wright
Argonne National Laboratory
University of Chicago

http://www.mcs.anl.gov/~wright/

Outline

- Stochastic Programming (SP)
- Formulation and Basic Algorithms for SP
- Condor and metaNEOS
- Asynchronous Trust-Region Algorithm
- Computational Results: Algorithm Performance, Solution Quality

Joint work with Jeff Linderoth (Argonne).

Stochastic Programming

Optimization of a model that contains **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.

x is a vector of real numbers.

Arises in planning-under-uncertainty applications, where each ξ represents a possible **scenario** (a possible way in which the model could evolve).

Space Ω can be finite or infinite.

Example: Natural Gas Supplier

(See NEOS Guide Case Studies:
www.mcs.anl.gov/otc/Guide/)

A supplier buys natural gas from a distributor and retails it to customers.

- Makes a purchase at the start of each month. Wants to plan over a two-month period.
- Knows the price and demand for the current month, but has the option of buying extra gas *now* and storing it, to hedge against price and demand in the next month — both uncertain
- There's a cost associated with storing it.

Decision: Given a set of possible scenarios for next month's prices and demands, how much gas should I buy now to minimize total expected cost?

First month data: Demand 10 units, Price \$1.00/unit, Storage Cost \$0.05/unit.

Second month data:

scenario	prob.	demand	price
1 (normal)	.50	10.0	1.00
2 (warm)	.30	8.0	0.85
3 (cold)	.15	14.0	1.50
4 (very cold)	.05	27.0	1.80
weighted av.		10.85	1.07

"Average case" approach: Assume that demand in second month is a (fixed) 10.85 units, with cost \$1.07; make decision accordingly.

Stochastic programming approach: Figure the cost of a decision to purchase x units in each of these 4 scenarios; find the weighted average over the scenarios; choose the x that minimizes this *expected cost* $f(x)$.

Solutions (units to buy now):

"average case"	20.85
stochastic programming	18.00

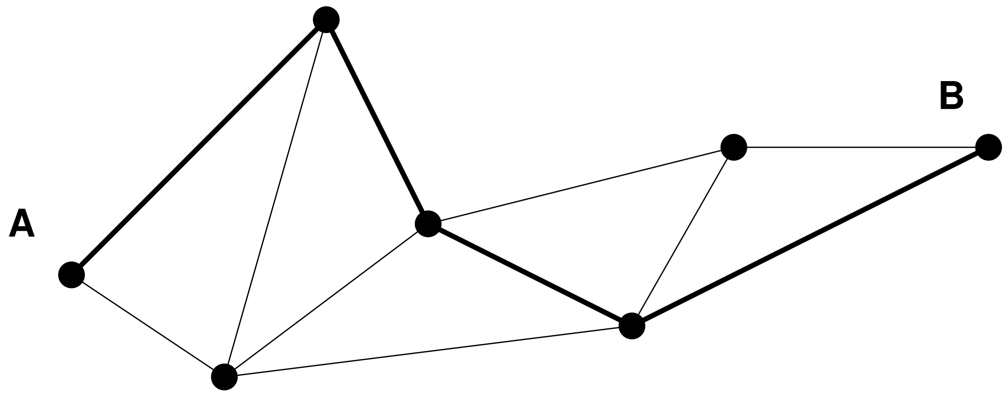
Costs:

scenario	probability	av. case	stoch prog
normal	.50	21.44	20.40
warm	.30	21.54	18.40
cold	.15	26.12	27.40
very cold	.05	50.46	52.60
expected cost		23.62	22.46

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: $i = 1, 2, \ldots, 86$.
  - 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, $j = 1, 2, \ldots, n$: 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** (SA) is still large, but manageable.

stochastic LP formulation

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

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

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

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

When $\Omega = \{\omega^1, \omega^2, \dots, \omega^K\}$ is finite (i.e. distribution is discrete), can write as

$$\min_x \hat{\mathcal{Q}}(x) = c^T x + \sum_{i=1}^K p_i \mathcal{Q}(x;\omega^i)$$
$$\text{subj. to } Ax = b, \;\; x \ge 0,$$

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

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

solving it: the deterministic equivalent

Lump first- and second-stage variables together to get one big (structured) problem:

$$\begin{gathered}\min_{x,y_1,\ldots,y_N} c^T x + N^{-1} \textstyle\sum_{i=1}^N q_i^T y_i \\ \text{subj. to } Ax = b, \ \ x \geq 0, \\ W y_i = h_i - T_i x, \ \ y_i \geq 0, \ \ i = 1, 2, \ldots, N.\end{gathered}$$

Note: one instance of second-stage variables y_i for each scenario.

Could treat this problem as a big linear program and solve it directly, taking account of the structure if possible.

$$\min_{\bar{x}} \bar{c}^T \bar{x}, \ \ \text{s.t.} \ \ \bar{A}\bar{x} = \bar{b}, \ \ \bar{x} \geq 0,$$

where constraint matrix, unknown vector are

$$\bar{A} = \begin{bmatrix} A & & & & \\ T_1 & W & & & \\ T_2 & & W & & \\ \vdots & & & \ddots & \\ T_N & & & & W \end{bmatrix}, \ \ \bar{x} = \begin{bmatrix} x \\ y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}.$$

Interior-point methods can exploit structure well. However they need significant data movement and synchronization, so are not ideal for the grid computing platforms we consider here.

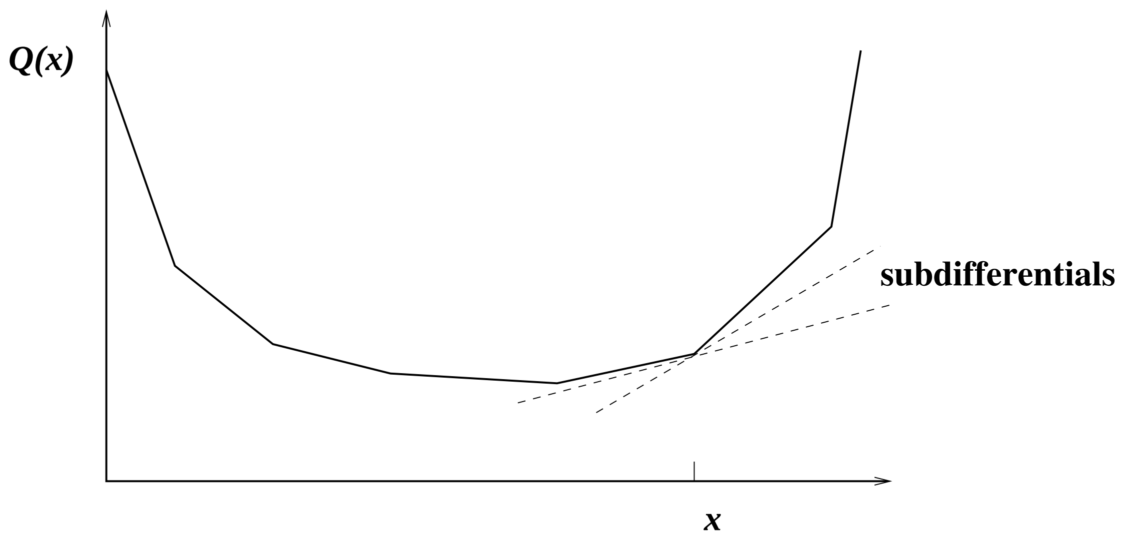
It's better to attack the problem in the x variables directly:

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

This is much smaller, but much more complex: each term $\mathcal{Q}(x; \omega_j)$ is convex, piecewise-linear in x.

Achieve parallelism via concurrent handling of the N second-stage terms $\mathcal{Q}(x; \omega_j)$, $j = 1, 2, \ldots, N$.

properties of $\mathcal{Q}(x;\omega_j)$ and $\mathcal{Q}(x;\omega)$



Compute subdifferentials of $\mathcal{Q}$ by

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

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

- summing:

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

L-shaped method

Build up a lower bounding, piecewise linear approximation to $\mathcal{Q}(x)$, based on function values $\mathcal{Q}(x^\ell)$ and subdifferentials g^ℓ at iterates x^ℓ, $\ell = 0, 1, 2, \ldots$.

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

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

Choose next iterate as

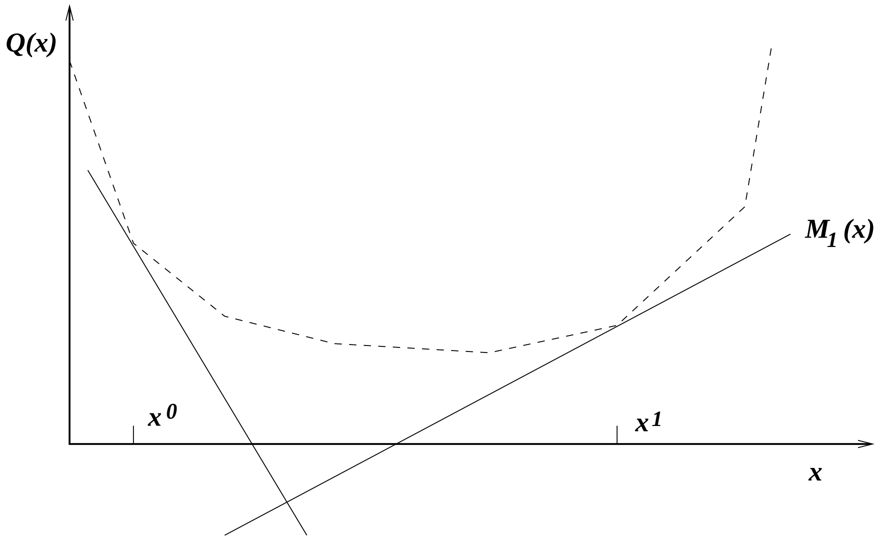
$$x^{k+1} = \arg\min M_k(x), \;\; \text{subj. to } Ax = b, \; x \geq 0.$$

Can formulate this as a linear program:

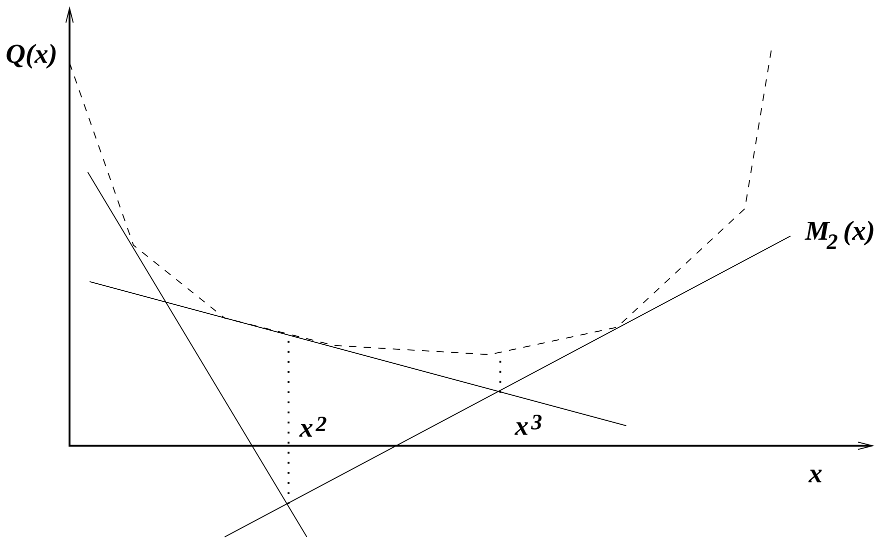
$$\begin{gathered} \min_{x,\theta} \theta, \;\; \text{subject to} \\ Ax = b, \;\; x \geq 0, \\ \theta \geq \mathcal{Q}(x^\ell) + (g^\ell)^T (x - x^\ell), \;\; \ell = 0, 1, \ldots, k. \end{gathered}$$

(Each additional 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:



L-shaped implementation

$k \leftarrow 0$, $\mathcal{Q}_{\min} \leftarrow \infty$, choose `tol`;

repeat

  $x^{k+1} = \arg\min M_k(x)$;

  for $i = 1, 2, \ldots, N$ (concurrently)

    evaluate $\mathcal{Q}_i(x^{k+1})$ and π_i;

  assemble $\mathcal{Q}(x^{k+1})$, subgradient g^{k+1};

  if $\mathcal{Q}(x^{k+1}) < \mathcal{Q}_{\min}$

    $\mathcal{Q}_{\min} \leftarrow \mathcal{Q}(x^{k+1})$;

  if $\mathcal{Q}_{\min} - M_k(x^{k+1}) \le$ `tol`

    stop;

end (repeat)

Group the second-stage problems into T "chunks" $\mathcal{N}_t$, $t = 1, 2, \ldots, T$, with

$$\cup_{t=1,2,\ldots,T} \mathcal{N}_t = \{1, 2, \ldots, N\},$$

so that calculation of $\mathcal{Q}_i(x^{k+1})$ for all $i \in \mathcal{N}_t$ represents a unit of work of reasonable size.

drawbacks and enhancements

- slow convergence;
- synchronizes on the function evaluation;
- for typical problems, scalability is limited;
- cannot exploit good starting point — the first few iterates take one far from the starting point;
- need to store all the cuts.

Some enhancements:

- *Multicut:* Return one or more cuts per chunk ($10 - 100$ cuts per iteration);
- *Trust-Region Strategy:* Stay within a certain distance of the current best point; move only if new point has a significantly better value of $\mathcal{Q}$;

trust-region (TR) strategy

Choose next iterate as

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

where Δ_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 decrease trust region.

trust region properties

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

- Can delete cuts liberally, between iterations;
- Iterates cannot get stuck at an nonoptimal point;
- $\text{dist}(x^k, \mathcal{S}) \to 0$.

For the piecewise linear function here, minimum is *weak sharp*, so can prove slightly stronger results about rate of convergence, lower bound on trust region.

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.)
- Ruszczyǹski "regularized decomposition" (1986) specifically for stochastic programming.

The last three give quadratic programming sub-problems, not LP.

Making it work: metaNEOS

metaNEOS is a collaboration of Wisconsin, Argonne, Northwestern, Columbia. Computational grid and optimization groups.

Aim to solve large optimization problems on computational grids by developing and deploying

- metacomputing infrastructure,
- optimization algorithms.

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.

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 very large and compute-intensive — and algorithms that are **asynchronous** — *are best suited.*

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 files or Condor-PVM.

MWApplications in metaNEOS

- MWFATCOP: Linear integer programming. (Chen, Ferris, Linderoth)
- MWMINLP: Nonlinear integer programming. (Goux, Leyffer)
- MWQAP: QAPs via QP relaxation, branch-and-bound. (Anstreicher, Brixius, Goux, Linderoth)
- MWATR: Asynchronous trust-region algorithm for stochastic linear programming. (This talk)

TR may not be synchronous 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.

We propose an *asynchronous trust-region* (ATR) algorithm to increase parallelism and throughput.

ATR

- Maintain an *incumbent* x^I: the best point found so far (smallest value of $\mathcal{Q}$).
- Maintain a *basket* $\mathcal{B}$ of 5 or 10 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: $\|x - x^I\|_\infty \le \Delta$. ($x^I$ becomes the *parent incumbent* of the new point.)
- When evaluation of a point $x \in \mathcal{B}$ is completed, accept it as the new incumbent if
  - $\mathcal{Q}(x) < \mathcal{Q}(x^I)$;
  - $\mathcal{Q}(x)$ gives a significant decrease over its parent incumbent.

ATR (continued)

- Populate $\mathcal{B}$ initially by solving TR subproblems around early iterates, using partial subgradient information.

- Can delete a cut once both (i) the point at which it was generated and (ii) the parent incumbent of that point are no longer in $\mathcal{B}$.

- Adjust trust region using similar heuristics to synchronous case.

convergence of ATR

Because of the strategy for selecting incumbents, can reduce the theory to that of 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

- If $x^I \notin \mathcal{S}$, it is eventually replaced by a new incumbent.
- $\text{dist}(x^I, \mathcal{S}) \to 0$.

SSN: Computational Results

- first stage: 89 variables, 1 constraint;
- second stage: 706 variables, 175 constraints, 2284 nonzeros.
- total 10^{70} scenarios.

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

(For comparison, the regularized decomposition code of Ruszczyński and Świetanowski takes 220 mins (109 iterations) on a fast sparc uniprocessor for a 5000-scenario problem.)

1. 10000 scenarios, cut deletion after 50 iterations. Tasks 50-90 seconds each.
2. 20000 scenarios, cut deletion after 100 iterations.

run	iter	$\vert\mathcal{B}\vert$	chunks	cuts/iter	av. procs	par. efficiency	wall clock (min)
L	255	-	10	50	19	.46	398.
TR	47	-	10	50	19	.35	130.
TR	45	-	10	50	11	.69	95.
ATR	99	5	10	50	71	.22	52.
ATR	164	10	10	50	71	.57	43.
ATR	223	15	10	50	52	.91	57.
ATR	107	10	20	100	31	.83	42.
SSN, $N = 10000$. 1.75M $\times$ 7.06M.							

run	iter	$\|\mathcal{B}\|$	chunks	cuts/iter	av. procs	par. efficiency	wall clock (min)
TR	83	-	20	60	37	.37	106
ATR	151	5	20	60	36	.86	100
ATR	204	10	20	60	66	.92	69
SSN, $N = 20000$. 3.50M $\times$ 14.1M.							

storm: comptuational results

Cargo flight scheduling problem (Mulvey and Riszczyński).

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

For a 250000 scenario sampled approx, LP is enormous!

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

Started from a solution for a 3000-scenario approximation.

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

Starting point is very good: Only 3 steps in TR, 5 in asynchronous TR.

run	iter	$\vert\mathcal{B}\vert$	chunks	cuts/iter	av. procs	par. efficiency	wall clock (min)
TR	25	-	125	125	67	.57	211
ATR	57	5	125	125	86	.96	229
storm, $N = 250,000$. 132M $\times$ 315M.							

some comments on solution quality

- How are the optimal objective values of the sampled approximations related to those of the real problem (with its zillions of scenarios)? Can we get confidence intervals?
- How do the *solutions* of the SA relate to those of the real problem?

Using the horsepower of MW-ATR, along with relevant theory (some recent), we can perform computational/statistical studies of these issues, for some difficult problems.

Report on just one issue: solution quality for SSN.

Recall that exact problem is

$$\min_{x \in X} \mathcal{Q}(x) = c^T x + \sum_{i=1}^{K} p_i \mathcal{Q}(x; \omega^i),$$

(where X is the polyhedral feasible set) while sampled approximation (SA) is

$$\min_{x \in X} \mathcal{Q}_{(N)}(x) = c^T x + N^{-1} \sum_{j=1}^{N} \mathcal{Q}(x; \omega_j).$$

Denote

$$Z^* = \min_{x \in X} \mathcal{Q}(x), \qquad Z_N = \min_{x \in X} \mathcal{Q}_{(N)}(x).$$

lower bound for Z^*

It's well known that

$$EZ_N \leq Z^*.$$

True for any unbiased estimator — in particular can use variance reduction techniques, e.g. Latin hypercube sampling to select the sample $\{\omega_1, \omega_2, \ldots, \omega_N\}$.

Can generate M batches — each a sampled approximation of size N — and solve the M SA's to obtain optimal values

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

and then estimate EZ_N by

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

upper bound for Z^*

Take any guess $\hat{x}$ of the solution, and a sample size P: $\{\omega_1, \omega_2, \ldots, \omega_P\}$. Estimate is

$$U_P(\hat{x}) = c^T \hat{x} + P^{-1} \sum_{j=1}^{P} \mathcal{Q}(\hat{x}, \omega_j).$$

By central limit theorem,

$$P^{-1/2} [U_P(\hat{x}) - \mathcal{Q}(\hat{x})] \sim N(0, \sigma^2),$$

where σ^2 is $\text{var}\mathcal{Q}(\hat{x}; \omega)$.

results on bounds: SSN

Results of Mak et al (1997).

	lower	upper
batch/sample	30×1000	1×100000
estimate	9.22	9.98
97.5% confidence	± 0.21	± 0.11

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

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

We use Latin hypercube sampling, larger sample sizes.

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

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

  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

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

  95% confidence interval is $[9.8614, 9.9397]$.

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

an exact solution?

Recent theory (Shapiro and Homen-de-Mello, 1998) suggests that for discrete distributions, it is possible to find the *exact solution of the underlying problem* by solving a SA with relatively few scenarios. Why?

- $\mathcal{Q}$ is piecewise linear and solution set A is sharp: for all $x \in X$, have $c > 0$ such that

  $$\mathcal{Q}(x) \geq Z^* + c\,\text{dist}(x, A).$$

  Hence for all $x^* \in A$, $d \in N_A(x^*) \cap T_X(x^*)$, $\|d\| = 1$, the directional derivative has

  $$\mathcal{Q}(x^*; d) \geq c.$$

- Directional derivatives converge: For any x, have

  $$\lim_{N\to\infty} \sup_{\|d\|=1} \left| \mathcal{Q}'(x; d) - \mathcal{Q}'_{(N)}(x; d) \right| = 0,$$

  with probability 1.

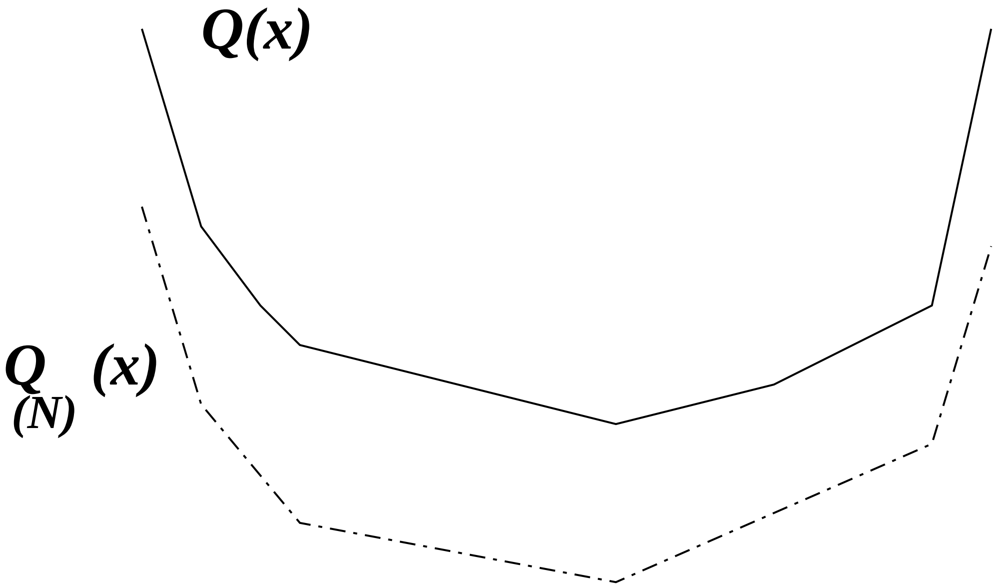
In fact, convergence of the directional derivative with N is fast enough to guarantee the following result: If $\hat{x}_N$ is the minimizer of $\mathcal{Q}_{(N)}$ then there is $\beta > 0$ such that

$$\limsup_{N\to\infty} \frac{1}{N} \log P(\hat{x}_N \notin A) \le -\beta.$$

Hence for large N we have

$$P(\hat{x}_N \in A) \ge 1 - e^{-\beta N}.$$

The chances of $\hat{x}_N$ *not* being a minimizer of $\mathcal{Q}$ go to zero exponentially!

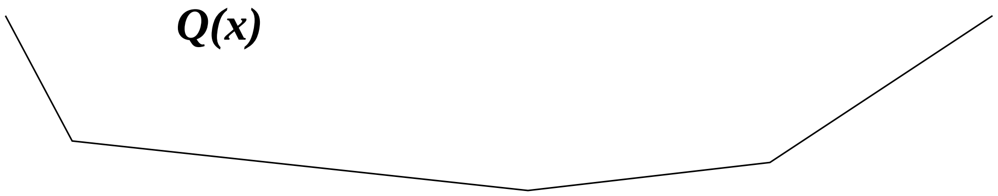


what can go wrong?

Small β!

Shapiro and Homen-de-Mello (2000) show that when $A = \{x^*\}$ (a singleton), we have

$$\beta \sim \sup_{d \in T_X(x^*), \|d\|=1} \frac{[\mathcal{Q}'(x^*; d)]^2}{\text{Var}\,[\mathcal{Q}'(x, \omega; d)]}.$$



case study: 20term

Test the convergence of $\hat{x}_N$ to A on a real problem with zillions of scenarios.

20term models a motor freight carrier's operations (Hackney and Infanger, 1994).

- first stage: 63 variables, 3 constraints, 63 nonzeros;
- second stage: 764 variables, 124 constraints, 4404 nonzeros.
- $\approx 10^{12}$ scenarios.

methodology

- Fix N, solve the problem with objective $\mathcal{Q}_{(N)}$ for two different samples of size N. Obtain candidates $\hat{x}_N^1$ and $\hat{x}_N^2$. (Use solution from smaller N as starting point.)

- If $\hat{x}_N^1 = \hat{x}_N^2$, take more samples of size N to check that the same minimizer works for these too. If so, indicates that $\hat{x}_N = x^*$.

- If not, could still have $\hat{x}_N^1, \hat{x}_N^2$ both in A. Use statistical tests (with new samples of large size) to check whether

$$\mathcal{Q}\left(\hat{x}_N^1\right) = \mathcal{Q}\left(\hat{x}_N^2\right).$$

  Do this by evaluating

$$\mathcal{Q}^j_{(N')}(\hat{x}_N^1) - \mathcal{Q}^j_{(N')}(\hat{x}_N^2), \quad j = 1, 2, \ldots, 10,$$

  where

$$\mathcal{Q}^j_{(N')}(x) = c^T x + (N')^{-1} \sum_{i=1}^{N'} \mathcal{Q}_i(x; \omega_i^j).$$

results for 20term

Took N from 1000 to 20000. Have $\widehat{x}_N^1 \neq \widehat{x}_N^2$ for each N, though for the second sample $\mathcal{Q}_{(N)}$ with $N = 20000$ we have

$$\begin{aligned}\mathcal{Q}_{(N)}(\widehat{x}_N^1) &= 254307.471886\\ \mathcal{Q}_{(N)}(\widehat{x}_N^2) &= 254307.441239.\end{aligned}$$

However using 10 batches of size $50,000$, we obtain an estimate of

$$\mathcal{Q}\left(\widehat{x}_N^1\right) - \mathcal{Q}\left(\widehat{x}_N^2\right)$$

of

$$.0170729$$

with standard error

$$.00229.$$

This suggests that with high probability, $\widehat{x}_N^1$ and $\widehat{x}_N^2$ do *not* have the same value of $\mathcal{Q}$, so $N = 20000$ is not large enough to have identified the exact solution.