RANKING & SELECTION FOR SIMULATION OPTIMIZATION

Barry L Nelson

Northwestern University
Lancaster University
STOR-i

Spring 2020

Northwestern | McCORMICK SCHOOL OF ENGINEERING

What you need for this class

- Nothing; you can just watch and listen.
- But if you want to do the exercises…
  - E-mail me nelsonb@northwestern.edu for the R code.
  - Install RStudio from https://rstudio.com/
  - Create an RStudio project for this class.
  - Open the RScript file I sent you called `FirstChallenge.R`.
  - If you decide to continue after the first video you should…
    - Open `Procedures.R`, `Simulations.R` and `ParallelProcedures.R`.
    - Install the `parallel` package from CRAN to your RStudio.

Hands-on example

We have the ability to simulate $k = 4$ different system designs that use redundancy to be resistant to system failure.

Let $Y(x)$ be the time to failure (TTF) of design type $x = 1, 2, 3, 4$. Your job is to find $x^\star = \text{argmax}_x \mathrm{E}[Y(x)]$. You have 10 minutes.

```
MySim <- function(x, n=1, RandomSeed=-1){
  # function to simulate CTMC TTF example
  # x in {1,2,3,4} is the system index
  # n is number of replications
  # RandomSeed sets the initial seed
  # output is time to system failure
```

Postmortem

Compile class results & supporting arguments.

If we were going to create an algorithm what would we want it to do?

- Control sample size for us.
- Provide statistical guarantees (such as?)
- Work for large k
- Exploit parallel computation
- Be efficient (how measured?)
- Other?

Optimizing simulated systems

Maximize E[Performance]
Subject to: Budget constraint on staff & machines

- Stochastic, dynamic, often nonstationary.
- Can only evaluate *instances*.
- May be computationally expensive.
- **The 3 errors:**
  - Don't visit the optimal solution.
  - Don't recognize the best solution visited.
  - Optimistic estimate of the performance of the selected solution.

A tiny bit of history

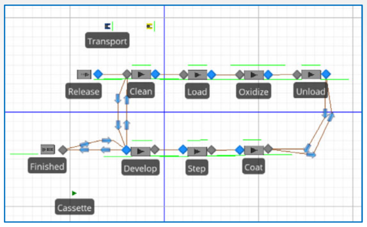
This class will address methods known collectively as Ranking & Selection (R&S).

- R&S originated with Bechhofer (Cornell) and Gupta (Purdue) in the 1950s for biostatistics types of applications:
  - Evaluate the efficacy of 3 drug treatments and a placebo.
- Characteristics included
  - small number of treatments k
  - normally distributed response
  - relatively equal (maybe even known) variances
  - need to be easy to implement
  - sampling done in batches, not sequentially

Then simulation adopted R&S…

At WSC 1983 Goldsman presented a tutorial on R&S and organized a session with Bechhofer & Gupta arguing that R&S was useful for optimizing simulated systems.

Simulation folks had grander delusions:



- Much larger numbers of “treatments” k.
- Non-normal (nominal) output data.
- Significantly unequal variances.
- Intentionally induced dependence due to Common Random Numbers (CRN).
- As complex as we want to be if it reduces number of “simulations.”

R&S: A simulation success story

R&S has been a theoretical and practical success for simulation:

- Strong theory; asymptotic regimes for non-normal data; effective use of “statistical learning.”
- Widely applied in real problems; included in many commercial languages.
  - Ex: KN and GSP in Simio Forward Thinking
- Can control all 3 errors.

Clearly there is a R&S limit since all feasible solutions must be simulated: much research has been on pushing that limit (e.g., statistically efficient; using parallel computing).

Outline

Simulation optimization (SO)

$\max_{\mathbf{x}} \mathrm{E}[Y(\mathbf{x})] = \mu(\mathbf{x})$	distribution of output objective depends on $\mathbf{x}$
subject to:	
$\mathbf{x} \in \mathrm{X}$	deterministic constraints
$\mathrm{E}[\mathbf{C}(\mathbf{x})] \in \mathrm{C}$	stochastic constraints

"Expectation" includes optimizing probabilities and chance constraints.

As in math programming, the nature of X (finite, countable, uncountable) has a huge impact on the approach.

Clear overlap with stochastic programming, however SO assumes $\mathrm{E}[Y(\mathbf{x})]$ and $\mathrm{E}[\mathbf{C}(\mathbf{x})]$ cannot be evaluated, but $Y(\mathbf{x})$ and $\mathbf{C}(\mathbf{x})$ can be simulated.

Approaches

1. **Finite X, simulate them all:** R&S; this masterclass.
2. **Finite or countable X, can't simulate them all:** Adaptive random search; statistical learning.
3. **Uncountable (continuous) X:** Stochastic gradient descent; statistical learning; sample-average approximation.

The guarantees (if any) for 2 & 3 are typically asymptotic (infinite effort). Dimension of X tends to be the limiting factor.

To date commercial software is dominated by R&S and metaheuristics.

Basics of the "best mean" R&S problem

If the true system performance expected values are

$$\mu(1) \le \mu(2) \le \cdots \le \mu(k-1) \le \mu(k)$$

then we refer to system k, or any system tied with system k, as the best.

For system x we can only estimate $\mu(x)$ with a consistent estimator such as the sample mean of $n(x)$ replications:

$$\bar{Y}(x) = \frac{1}{n(x)} \sum_{j=1}^{n(x)} Y_j(x)$$

The R&S procedure returns something like $\widehat{x}^{\star} = \mathrm{argmax}_{x \in \{1,2,\ldots,k\}} \bar{Y}(x)$.

Objective: Fixed precision

Simulate until a fixed inference is achieved; ideally PCS: $\Pr\{\widehat{x}^\star = k\} \geq 1 - \alpha$.

Since this can be computationally infeasible, a compromise is made such as...

- **Indifference zone:** $\Pr\left\{\widehat{x}^\star = k \mid \mu(k) - \mu(k-1) \geq \delta\right\} \geq 1 - \alpha$
- **Good selection:** $\Pr\left\{\mu(k) - \mu(\widehat{x}^\star) \leq \delta\right\} \geq 1 - \alpha$
- **Top m:** $\Pr\left\{\widehat{x}^\star \in [k, k-1, \ldots, k-m+1]\right\} \geq 1 - \alpha$
- **Subset:** Find $\widehat{\mathcal{S}} \subseteq \{1, 2, \ldots, k\}$ such that $\Pr\{k \in \widehat{\mathcal{S}}\} \geq 1 - \alpha$

These are typically *frequentist* guarantees to be achieved as efficiently as possible.

Objective: Fixed budget

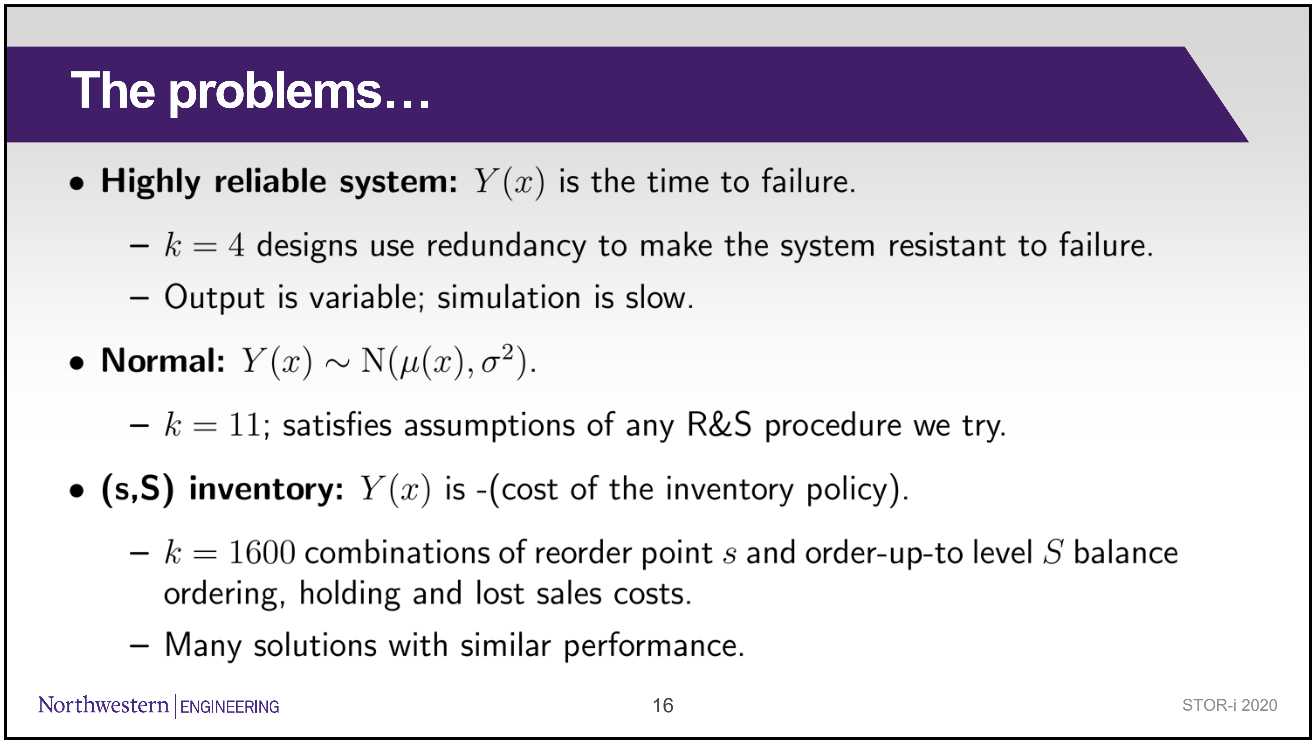
Obtain as strong an inference as possible within a fixed computation budget.

Formulated as minimizing some expected loss for the chosen solution: $\mathrm{E}[\mathcal{L}(\widehat{x}^\star)]$.

Inference is typically *Bayesian* in nature:

- **0-1 Loss:** Maximize posterior PCS
- **Opportunity cost:** Minimize posterior expected optimality gap

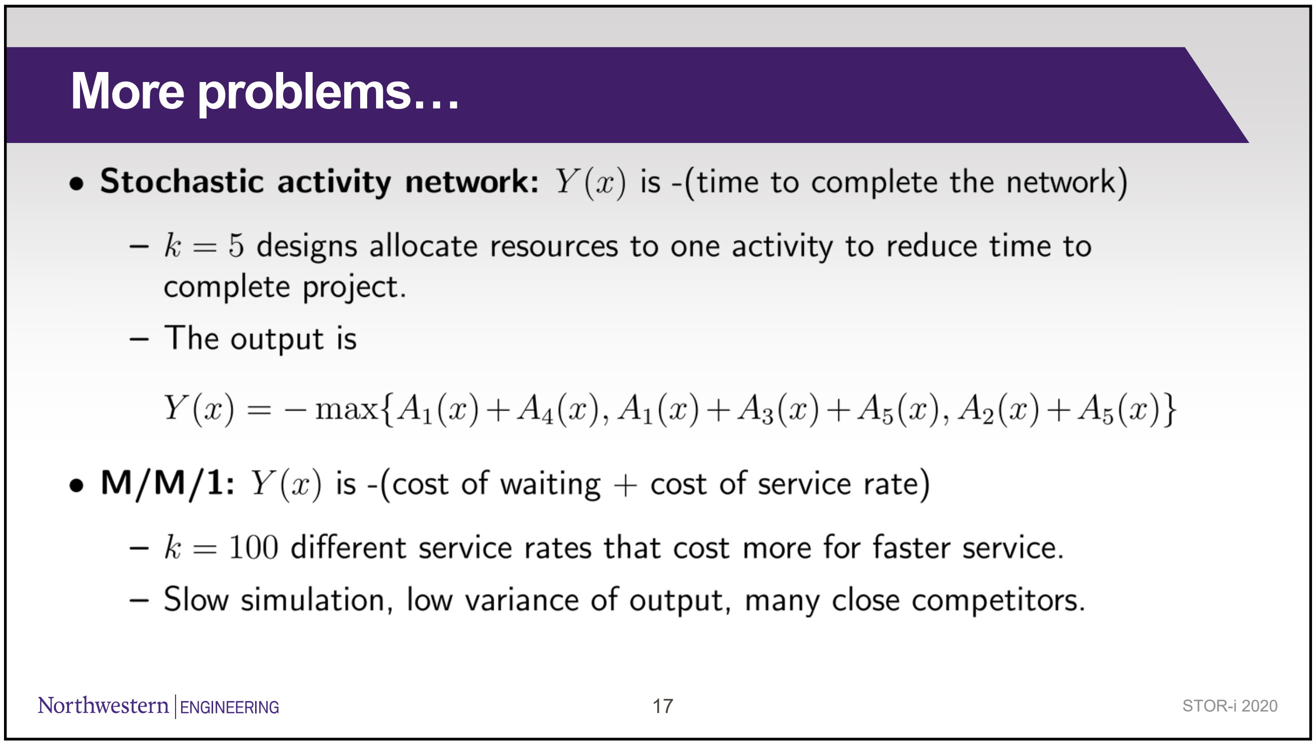
Approaches include "Expected Improvement," "Knowledge Gradient," etc.



The problems…

- **Highly reliable system:** $Y(x)$ is the time to failure.
  - $k = 4$ designs use redundancy to make the system resistant to failure.
  - Output is variable; simulation is slow.
- **Normal:** $Y(x) \sim \mathrm{N}(\mu(x), \sigma^2)$.
  - $k = 11$; satisfies assumptions of any R&S procedure we try.
- **(s,S) inventory:** $Y(x)$ is -(cost of the inventory policy).
  - $k = 1600$ combinations of reorder point s and order-up-to level S balance ordering, holding and lost sales costs.
  - Many solutions with similar performance.

Northwestern | ENGINEERING 16 STOR-i 2020



More problems…

- **Stochastic activity network:** $Y(x)$ is -(time to complete the network)
  - $k = 5$ designs allocate resources to one activity to reduce time to complete project.
  - The output is

$$Y(x) = -\max\{A_1(x)+A_4(x), A_1(x)+A_3(x)+A_5(x), A_2(x)+A_5(x)\}$$

- **M/M/1:** $Y(x)$ is -(cost of waiting + cost of service rate)
  - $k = 100$ different service rates that cost more for faster service.
  - Slow simulation, low variance of output, many close competitors.

Northwestern | ENGINEERING 17 STOR-i 2020

The normal means case

From system x, $Y_1(x), Y_2(x), \ldots$ are i.i.d. $\mathrm{N}(\mu(x), \sigma^2(x))$.

It may be that $\mathrm{Cov}(Y(x), Y(x')) \neq 0$ if we use common random numbers.

Question: Is normally distributed output really relevant? **Answers:**

- Output Y is often the average of more basic outputs (e.g., daily average customer waiting time).
- Sample sizes are large, so we can group or "batch" outputs.
- Sample sizes are large, so normal-theory methods apply asymptotically.

Initially we assume we can only simulate one system at a time; later we parallelize.

Rinott's Procedure

1. Choose confidence level $1-\alpha$, initial sample size $n_0 \geq 2$ and "indifference zone" $\delta > 0$. Set $h = h(k, 1-\alpha, n_0)$. Note $h(4, 0.95, 50) = 3.074$.
2. For each system $x = 1, 2, \ldots, k$ do the following:
   (a) Simulate n_0 replications and compute the sample variance $S^2(x)$.
   (b) Compute $N(x) = \left\lceil \dfrac{h^2 S^2(x)}{\delta^2} \right\rceil$
   (c) Simulate $\max\{0, N(x) - n_0\}$ additional replications.
   (d) Compute the sample mean $\bar{Y}(x)$.
3. Choose $\widehat{x}^\star = \mathrm{argmax}_x \bar{Y}(x)$

Using your birthday as your seed [set.seed(211256)], run Rinott on the TTF problem with $n_0 = 50$, $\alpha = 0.05$ and $\delta = 1000$ hours.

```
Rinott <- function(k, alpha, n0, delta){
  # implements Rinott's procedure
  # k = number of systems
  # 1-alpha = desired PCS
  # n0 = first-stage sample size
  # delta = indifference-zone parameter
  # note: uses 99% UCB for Rinott's h
  h <- Rinotth(k,n0,1-alpha,0.99,10000)$UCB
  Ybar <- NULL
  Vars <- NULL
  Ns <- NULL
  # loop through the k systems
  for (x in 1:k){
    Y <- MySim(x, n0)
    S2 <- var(Y)
    N <- ceiling(h^2*S2/delta^2)
    if (N > n0){
      Y <- c(Y, MySim(x, N-n0))
    }
    Ybar <- c(Ybar, mean(Y))
    Vars <- c(Vars, S2)
    N <- max(N, n0)
    Ns <- c(Ns, N)
  }
  list(Best = which.max(Ybar), Ybar = Ybar,
       Var = Vars, N = Ns)
}
```

Rinott guarantees

- Rinott assumes the outputs are i.i.d. normally distributed, unknown and possibly unequal variances, and independent across systems.
  - Implies distinct random number seeds.
  - Does our data look normally distributed?
- Let $\mu(1) \le \mu(2) \le \cdots \le \mu(k)$, so system k is best. Rinott guarantees

$$\mathrm{PCS} = \Pr\{\widehat{x}^\star = k \mid \mu(k) - \mu(k-1) \ge \delta\} \ge 1 - \alpha$$

  Later we will learn how Rinott-like procedures provide this guarantee.
- δ is often interpreted as the "smallest practically significant difference" but is called the *indifference-zone parameter*.

Rinott pro & con

- Rinott is easy to implement, and requires no coordination among systems (easy to parallelize).
- But it is pessimistic: it assumes the means are in the "least favorable configuration" $\mu(1) = \cdots = \mu(k-1) = \mu(k) - \delta$.
- What happens if there are other good (closer than δ) systems?
  - Turns out it also has $1-\alpha$ guarantee of selecting a "good" system (within δ of the best); not true for all procedures.
- $N(x)$ grows as h^2/δ^2. How does $h(k, 1-\alpha, n_0)$ grow with the number of systems k? (Answer: too fast).

Foundation

Since we assume $\mu(k) - \mu(x) \geq \delta$, $x \neq k$,

$$\begin{aligned}
&\Pr\left\{\bar{Y}(k) > \bar{Y}(x)\right\} \\
&\quad = \Pr\left\{\bar{Y}(k) - \bar{Y}(x) > 0\right\} \\
&\quad = \Pr\left\{\bar{Y}(k) - \bar{Y}(x) - [\mu(k) - \mu(x)] > -[\mu(k) - \mu(x)]\right\} \\
&\quad \geq \Pr\left\{\bar{Y}(k) - \bar{Y}(x) - [\mu(k) - \mu(x)] > -\delta\right\}.
\end{aligned}$$

The statistic

$$\bar{Y}(k) - \bar{Y}(x) - [\mu(k) - \mu(x)]$$

has mean 0, so we can find the number of replications needed to provide the desired probability guarantee considering only δ and the variances.

Indifference-zone paradigm

This formulation—where we want $\mathrm{PCS} \geq 1-\alpha$ when $\mu(k)-\mu(x) \geq \delta$ and we assume the LFC—has been dominant in frequentist R&S.

- Frees the probability statements from dependence on the true means.

There are two challenges:

1. When $\mu(k)-\mu(x) \gg \delta$ the LFC does not exploit it.
2. What happens if $\mu(k)-\mu(x) < \delta$ for some inferior system x? We would like a "good selection" guarantee:

$$\mathrm{PGS} = \Pr\{\mu(k)-\mu(\widehat{x}^\star) < \delta\} \geq 1-\alpha.$$

It is not always the case that indifference-zone PCS implies PGS.

Northwestern | ENGINEERING 24 STOR-i 2020

R&S based on "statistical learning"

These are ideas based (formally or informally) on Bayesian reasoning.

Frequentist: $\mu(1),\ldots,\mu(k)$ are *fixed* performance measures and probability statements (e.g., PCS) are with respect to repeated experiments.

Bayesian: Reduce our uncertainty by updating our knowledge.

$$\underbrace{\mu(1),\ldots,\mu(k)}_{\text{your problem}} \quad \sim \quad \underbrace{M(1),\ldots,M(k)}_{\text{r.v.'s with a "prior" distribution}}$$

After observing some data $\mathcal{H} = \{Y_j(x)\}$ we update our knowledge based on the conditional ("posterior") distribution $[M(1),\ldots,M(k)]|\mathcal{H}$.

Northwestern | ENGINEERING 25 STOR-i 2020

Generic Bayesian R&S

1. For $x \in \{1, 2, \ldots, k\}$, set $n(x) = 0$, $\bar{Y}(x) = \text{null}$, $\mathcal{H}_0 = \emptyset$, $j = 0$
2. $x^{(j)} = \pi(\mathcal{H}_j)$ and simulate $Y_{j+1}(x^{(j)})$ [policy $\pi(\cdot)$ based on posterior]
3. Update $n(x^{(j)}) = n(x^{(j)}) + 1$ and $\bar{Y}(x^{(j)}) = \dfrac{1}{n(x^{(j)})} \displaystyle\sum_{i:\, x^{(i)} = x^{(j)}} Y_{i+1}(x^{(i)})$

   $\mathcal{H}_{j+1} = \mathcal{H}_j \cup \left\{x^{(j)}, Y_{j+1}(x^{(j)})\right\}$
4. If budget is exhausted then return $\widehat{x}^\star = \text{argmax}_x \bar{Y}(x)$
   else $j = j + 1$ and go to 2

The policy

Clearly the action is in the policy $\pi(\cdot)$.

Typically the policy is expressed as some sort of "acquisition" function; e.g.,

$$\text{argmax}_{x \neq \widehat{x}^\star}\, a(x, \widehat{x}^\star) = \text{argmax}_{x \neq \widehat{x}^\star}\, \mathrm{E}\left[\max\left\{0, M(x) - M(\widehat{x}^\star)\right\} \middle| \mathcal{H}\right]$$

which is the solution with the largest posterior expected improvement.

An additional goal is to learn "optimally," meaning as efficiently as we can.

Finally, the policy has to be computable, which often means it cannot look too many steps ahead.

Fun facts

Gaussian processes provide a very useful framework for this sort of approach.

If $(Z_1, Z_2) \sim \mathrm{BVN}(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$ then $Z_1 \sim \mathrm{N}(\mu_1, \sigma_1^2)$, But

$$Z_1 | Z_2 = z \sim \mathrm{N}\underbrace{\left(\mu_1 + \rho\frac{\sigma_1}{\sigma_2}(z - \mu_2),\ \sigma_1^2(1-\rho^2)\right)}_{\text{“learning”}}$$

If $Z \sim \mathrm{N}(0,1)$ then $\mathrm{E}\left[\max\{0, \mu + \sigma Z\}\right] = \mu\Phi\left(\frac{\mu}{\sigma}\right) + \sigma\phi\left(\frac{\mu}{\sigma}\right)$ where Φ and ϕ are the cdf and density of Z.

Complete Expected Improvement policy

If $\widehat{x}^\star$ is the current sample best solution, select as the next solution to simulate

$$\pi(\mathcal{H}) = \mathrm{argmax}_{x \neq \widehat{x}^\star}\ \mathrm{E}\left[\max\left\{0, M(x) - M(\widehat{x}^\star)\right\} \middle| \mathcal{H}\right] = \mathrm{argmax}_{x \neq \widehat{x}^\star} \mathrm{CEI}(x, \widehat{x}^\star)$$

When the posterior is normal, then using the fun fact we have

$$\begin{aligned}\mathrm{CEI}(x, \widehat{x}^\star) &= (m(x) - m(\widehat{x}^\star))\Phi\left(\frac{m(x) - m(\widehat{x}^\star)}{\sqrt{\mathrm{Var}(x, \widehat{x}^\star)}}\right) \\ &+ \sqrt{\mathrm{Var}(x, \widehat{x}^\star)}\phi\left(\frac{m(x) - m(\widehat{x}^\star)}{\sqrt{\mathrm{Var}(x, \widehat{x}^\star)}}\right)\end{aligned}$$

where $m(x) = \mathrm{E}(M(x))$, $\mathrm{Var}(x, \widehat{x}^\star) = \mathrm{Var}(M(x) - M(\widehat{x}^\star))$. Is this a good idea?

A convergence-rate perspective

Suppose that the best system is unique: $\mu(k) > \mu(k-1)$.

Then as all the $n(x) \to \infty$, even if not equal, we will eventually correctly select $\widehat{x}^\star = k$ due to the strong law of large numbers.

But what is the best way to get to ∞?

A formulation:

Let $n(x) = \beta_x N$ where $\beta_x \geq 0$ and $\sum_x \beta_x = 1$.

What choice of $\beta_1, \ldots, \beta_k$ makes $\lim_{N\to\infty} \Pr\{\widehat{x}^\star \neq k\}$ go to 0 the fastest?

A third pillar of statistics: Large deviations

$Z_1, Z_2, \ldots, Z_N$ i.i.d. (μ, σ^2), plus.... Then as $N \to \infty$,

1. **SLLN:** $\bar{Z}(N) \xrightarrow{a.s.} \mu$
2. **CLT:** $\sqrt{N}(\bar{Z}(N) - \mu) \xrightarrow{\mathcal{D}} \sigma \mathrm{N}(0,1)$
3. **LDP:** $\lim_{N\to\infty} \frac{1}{N} \ln[\Pr\{\bar{Z}(N) > z\}] = -I(z)$ where $I(\cdot)$ is the *rate function* that depends on the distribution of Z. The LDP can be interpreted as

$$\Pr\{\bar{Z}(N) > z\} \approx e^{-NI(z)} \text{ for large } N$$

For R&S we want to choose $\beta_1, \ldots, \beta_k$ to maximize the rate of decay of

$$\mathrm{PICS} = \Pr\{\bar{Y}_x(\beta_x N) - \bar{Y}_k(\beta_k N) > 0\} \approx \exp(-NI(0, \beta_x, \beta_k))$$

LD optimal allocation

Glynn and Juneja (2004) showed that if the outputs are normally distributed then the LD rate-optimal allocation satisfies [let $\mu_x = \mu(x), \sigma_x = \sigma(x)$].

$$\left(\frac{\beta_k}{\sigma_k}\right)^2 = \sum_{x \neq k} \left(\frac{\beta_x}{\sigma_x}\right)^2$$

$$\frac{(\mu_x - \mu_k)^2}{\frac{\sigma_x^2}{\beta_x} + \frac{\sigma_k^2}{\beta_k}} = \frac{(\mu_{x'} - \mu_k)^2}{\frac{\sigma_{x'}^2}{\beta_{x'}} + \frac{\sigma_k^2}{\beta_k}}, \quad \forall x, x' \neq k$$

Obvious problem: This expression involves things we don't know, and just plugging in estimates does not give the best possible rate (although it is not horrible). Things get harder for unknown distributions (estimating LD rates is difficult).

Connections

Optimal Computer Budget Allocation (OCBA) arrives at this result through a Bayesian-inspired approximation to the posterior PCS. OCBA uses plug-in estimates and nonlinear optimization to allocate batches of runs to achieve this balance; it is a heuristic. See Chen and Lee (2011).

CEI: Chen and Ryzhov (2017) showed that a slight modification of the CEI policy is asymptotically equivalent to the rate-optimal allocation! This result is remarkable because CEI comes from unrelated reasoning: the Bayes-optimal allocation of the next simulation run if it will be your last.

We will play with a version of the CEI algorithm now....

```
cei <- function(k, n0, Nmax){
  f <- function(z){z*pnorm(z) + dnorm(z)}
  Ybar <- rep(0, k)
  Sum2 <- rep(0, k)
  N <- rep(0, k)
  Y <- rep(0, k)
  CEI <- rep(0, k)
  systems <- 1:k

  # get n0 reps from each system
  for (i in 1:k){
    for (j in 1:n0){
      Y[j] <- MySim(i)
    }
    Ybar[i] <- mean(Y)
    Sum2[i] <- (n0-1)*var(Y)
    N[i] <- n0
  }
  # start sequential allocation
  while(sum(N) < Nmax){
    xstar <- which.max(Ybar)    # sample best
  # check if sample best has too few reps
      if (2*N[xstar]^2/(Sum2[xstar]/(N[xstar]-1))
         < sum(N^2/(Sum2/(N-1)))){
       x <- xstar}
     else{            # calculate CEIs
       S2 <- Sum2/(N - 1)/N
       for (i in systems){
         v <- sqrt(S2[xstar] + S2[i])
         CEI[i] <- v*f(-abs(Ybar[i]
                        - Ybar[xstar])/v)
         CEI[xstar] <- 0}
       x <- which.max(CEI)
     }
   # simulate x and update statistics
    Yx <- MySim(x)
    difference <- Yx - Ybar[x]
    Ybar[x] <- Ybar[x] + difference/(N[x]+1)
    Sum2[x] <- Sum2[x] + difference*(Yx - Ybar[x])
    N[x] <- N[x] + 1 }
}
```

STOR-i 2020

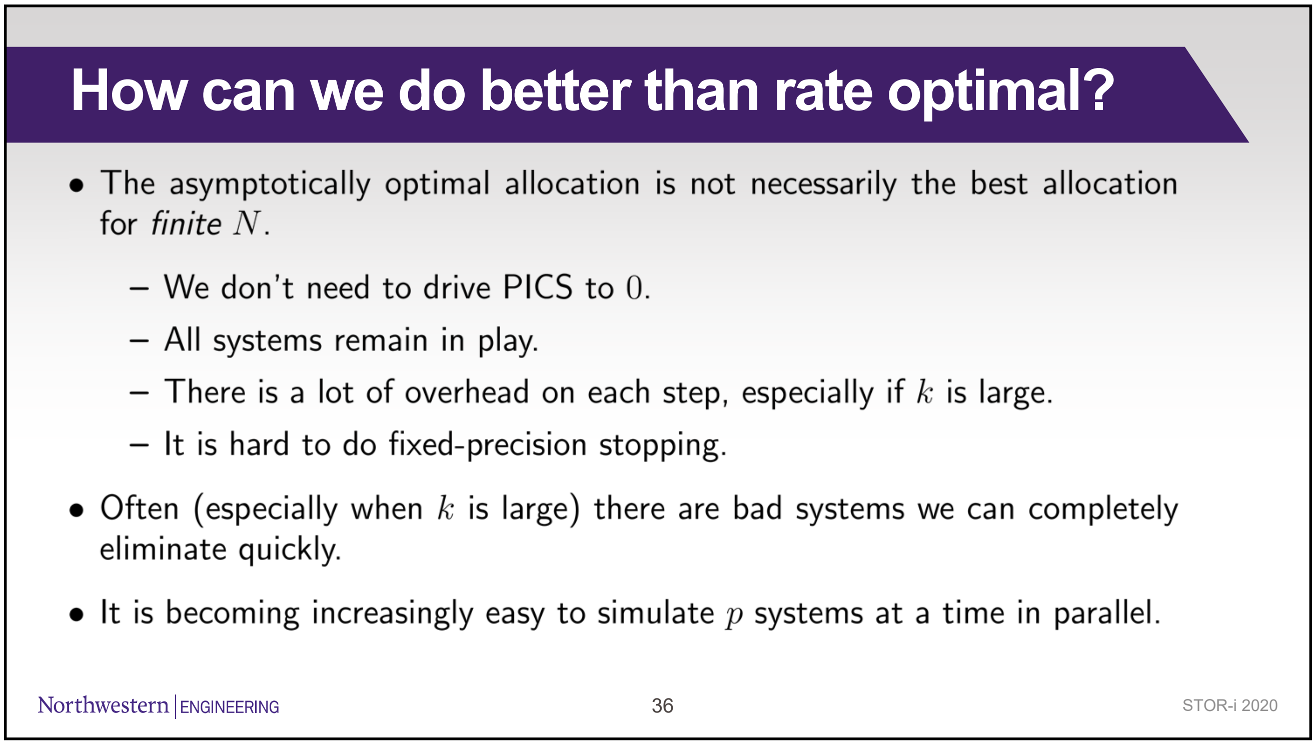
Trying out CEI

1. Load the SAN example, which has $k = 5$ alternatives.
2. Run CEI with $n_0 = 20$ and maximum observations 5000. Remember to set the seed to your birthday before starting.

```
result <- cei(5, 20, 5000)

result$xstar          # estimated optimal
result$N              # observations allocated to each system
plot(results$xpath)   # sequence of solutions x simulated
plot(result$Ypath)    # sequence of estimated optimal value
```

3. Which solution did you get as optimal? Which solutions were simulated most?

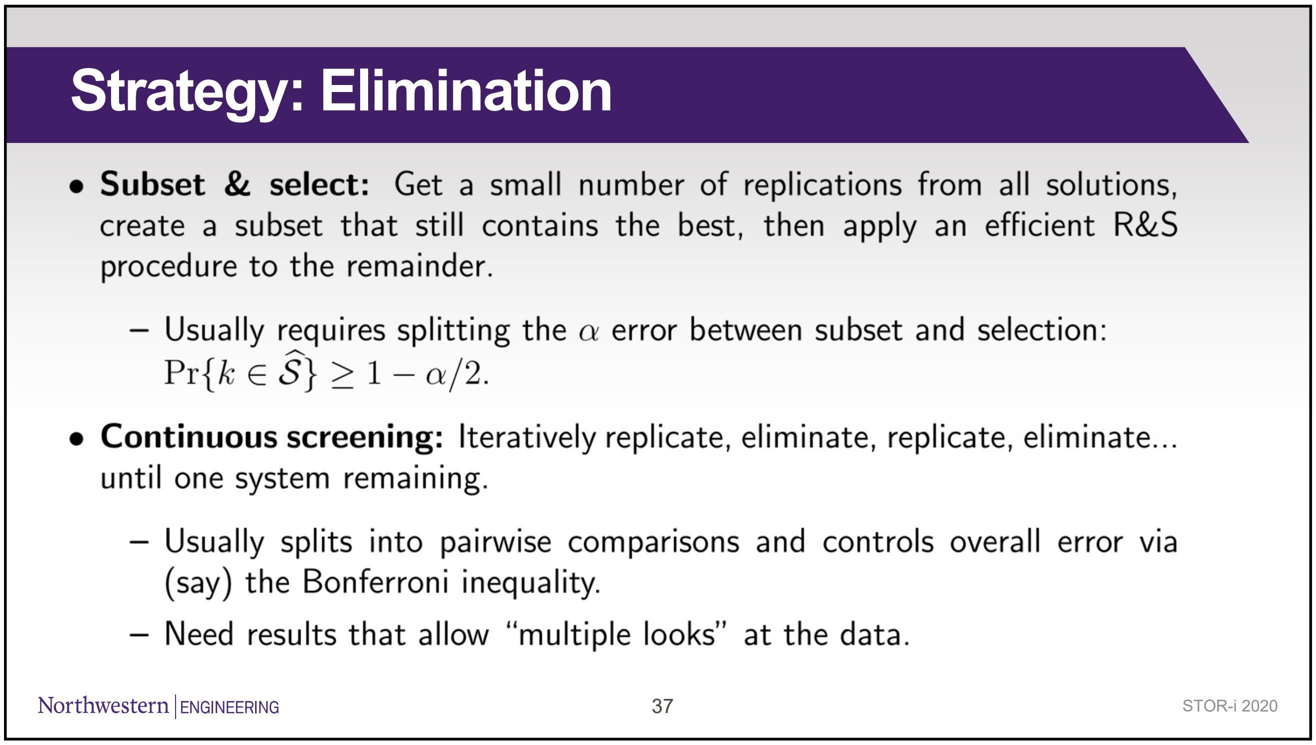
Northwestern | ENGINEERING 35 STOR-i 2020



How can we do better than rate optimal?

- The asymptotically optimal allocation is not necessarily the best allocation for *finite* N.
  - We don't need to drive PICS to 0.
  - All systems remain in play.
  - There is a lot of overhead on each step, especially if k is large.
  - It is hard to do fixed-precision stopping.
- Often (especially when k is large) there are bad systems we can completely eliminate quickly.
- It is becoming increasingly easy to simulate p systems at a time in parallel.

Northwestern | ENGINEERING 36 STOR-i 2020



Strategy: Elimination

- **Subset & select:** Get a small number of replications from all solutions, create a subset that still contains the best, then apply an efficient R&S procedure to the remainder.
  - Usually requires splitting the α error between subset and selection: $\Pr\{k \in \widehat{\mathcal{S}}\} \geq 1 - \alpha/2$.
- **Continuous screening:** Iteratively replicate, eliminate, replicate, eliminate... until one system remaining.
  - Usually splits into pairwise comparisons and controls overall error via (say) the Bonferroni inequality.
  - Need results that allow "multiple looks" at the data.

Northwestern | ENGINEERING 37 STOR-i 2020

Basic subset selection

1. Given $n(x) \geq 2$ observations from solution x, set $t(x) = t_{(1-\alpha)^{\frac{1}{k-1}},\, n(x)-1}$ the $(1-\alpha)^{\frac{1}{k-1}}$ quantile of the t distribution with $n(x)-1$ degrees of freedom, for $x = 1, 2, \ldots, k$.
2. Calculate the sample means $\bar{Y}(x)$ and sample variances

$$S^2(x) = \frac{1}{n(x)-1} \sum_{j=1}^{n(x)} \left(Y_j(x) - \bar{Y}(x)\right)^2$$

for $x = 1, 2, \ldots, k$, and also for all $x \neq x'$

$$W(x, x') = \left(t(x)^2 \frac{S^2(x)}{n(x)} + t(x')^2 \frac{S^2(x')}{n(x')} \right)^{1/2}$$

3. Form the subset

$$\widehat{\mathcal{S}} = \left\{x \colon \bar{Y}(x) \geq \bar{Y}(x') - W(x, x') \text{ for all } x' \neq x\right\}.$$

Subset foundation

The following is behind many subset selection procedures:

$$\begin{aligned}
&\Pr\{k \in \widehat{\mathcal{S}}\} \\
&\quad = \Pr\left\{\bar{Y}(k) \geq \bar{Y}(x) - W(k, x),\ x \neq k\right\} \\
&\quad = \Pr\left\{\bar{Y}(k) - \bar{Y}(x) - [\mu(k) - \mu(x)] \geq -W(k, x) - [\mu(k) - \mu(x)],\ x \neq k\right\} \\
&\quad \geq \Pr\left\{\bar{Y}(k) - \bar{Y}(x) - [\mu(k) - \mu(x)] \geq -W(k, x),\ x \neq k\right\}.
\end{aligned}$$

The statistic

$$\bar{Y}(x) - \bar{Y}(x') - [\mu(x) - \mu(x')]$$

has mean 0 for all $x \neq x'$, allowing the $W(x, x')$'s to be derived that give the desired probability based only on their variances.

Subset procedure

```
Subset <- function(k, alpha, n){
  Yall <- NULL
  for (x in 1:k){Yall <- cbind(Yall, MySim(x, n))}
  Ybar <- apply(Yall, 2, mean)
  S2 <- apply(Yall, 2, var)/n
  tval <- qt((1-alpha)^(1/(k-1)), df = n-1)
  Subset <- 1:k
  for (i in 1:k){
    for (j in 1:k){
      if (Ybar[i] < (Ybar[j]-tval*sqrt(S2[i] + S2[j]))){
        Subset[i] <- 0
        break
      }
    }
  }
  list(Subset = Subset[Subset != 0], Ybar = Ybar, S2 = S2)}
```

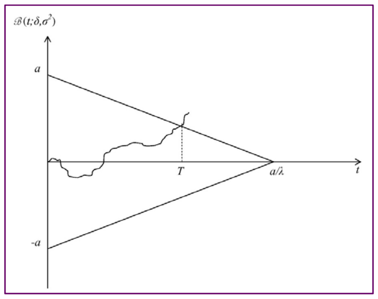
Load the M/M/1 example which has $k = 100$ solutions and run Subset first with $n_0 = 10$, then $n_0 = 100$ at $\alpha = 0.05$. Remember to set the seed to your birthday each time.

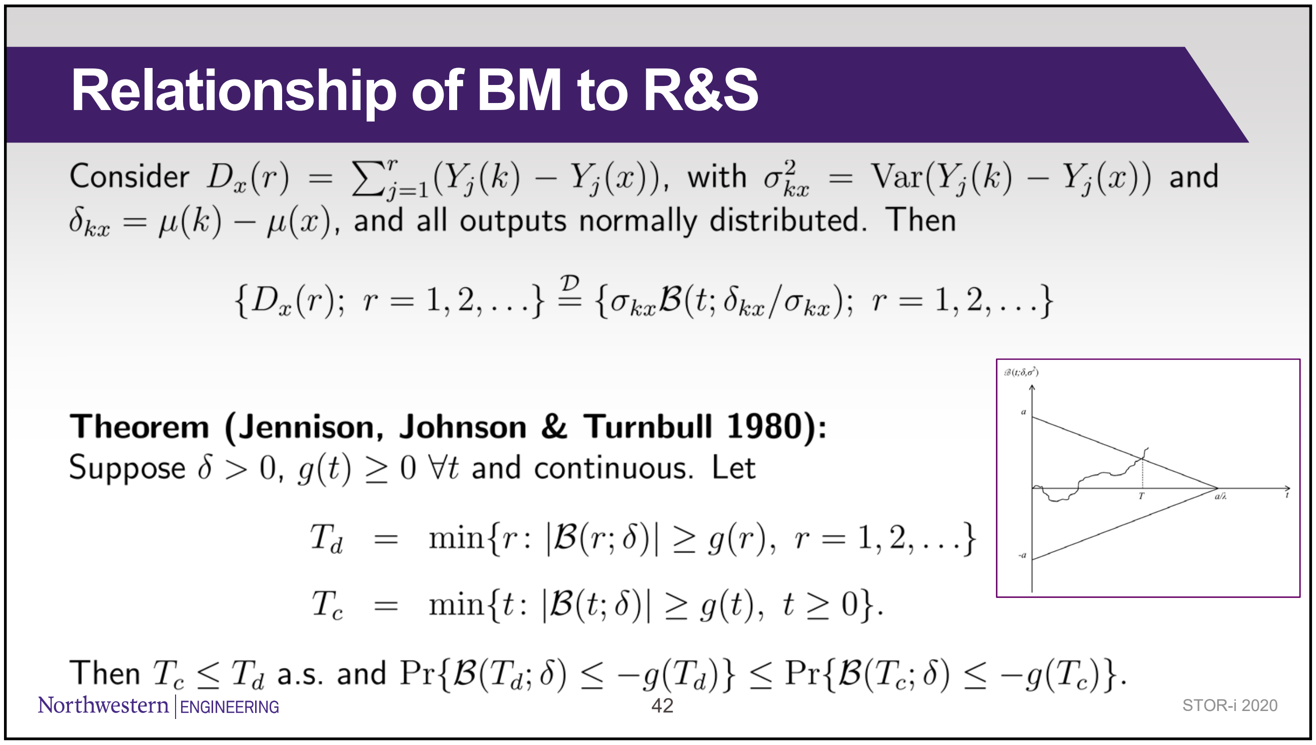
The role of Brownian motion

Let $\{\mathcal{B}(t);\ t \geq 0\}$ be *standard Brownian motion (BM)*:

1. $\mathcal{B}(0) = 0$
2. $\mathcal{B}(t)$ is almost surely continuous
3. $\mathcal{B}(t)$ has independent increments: $\mathcal{B}(t) \perp \mathcal{B}(t+s) - \mathcal{B}(t)$
4. $\mathcal{B}(t) - \mathcal{B}(s) \sim \mathrm{N}(0, t-s),\ 0 \leq s \leq t$
5. BM with *drift*: $\mathcal{B}(t;\delta) = \mathcal{B}(t) + \delta t$
6. Scaling: $\sigma\mathcal{B}(t;\delta/\sigma) = \sigma\mathcal{B}(t) + \delta t$

A lot is known about BM exiting regions like $\rightarrow$





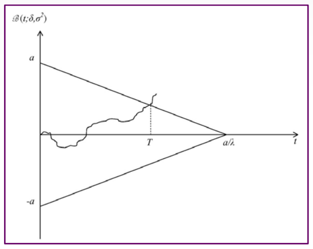
Relationship of BM to R&S

Consider $D_x(r) = \sum_{j=1}^{r}(Y_j(k) - Y_j(x))$, with $\sigma_{kx}^2 = \mathrm{Var}(Y_j(k) - Y_j(x))$ and $\delta_{kx} = \mu(k) - \mu(x)$, and all outputs normally distributed. Then

$$\{D_x(r);\ r = 1, 2, \ldots\} \overset{\mathcal{D}}{=} \{\sigma_{kx}\mathcal{B}(t; \delta_{kx}/\sigma_{kx});\ r = 1, 2, \ldots\}$$

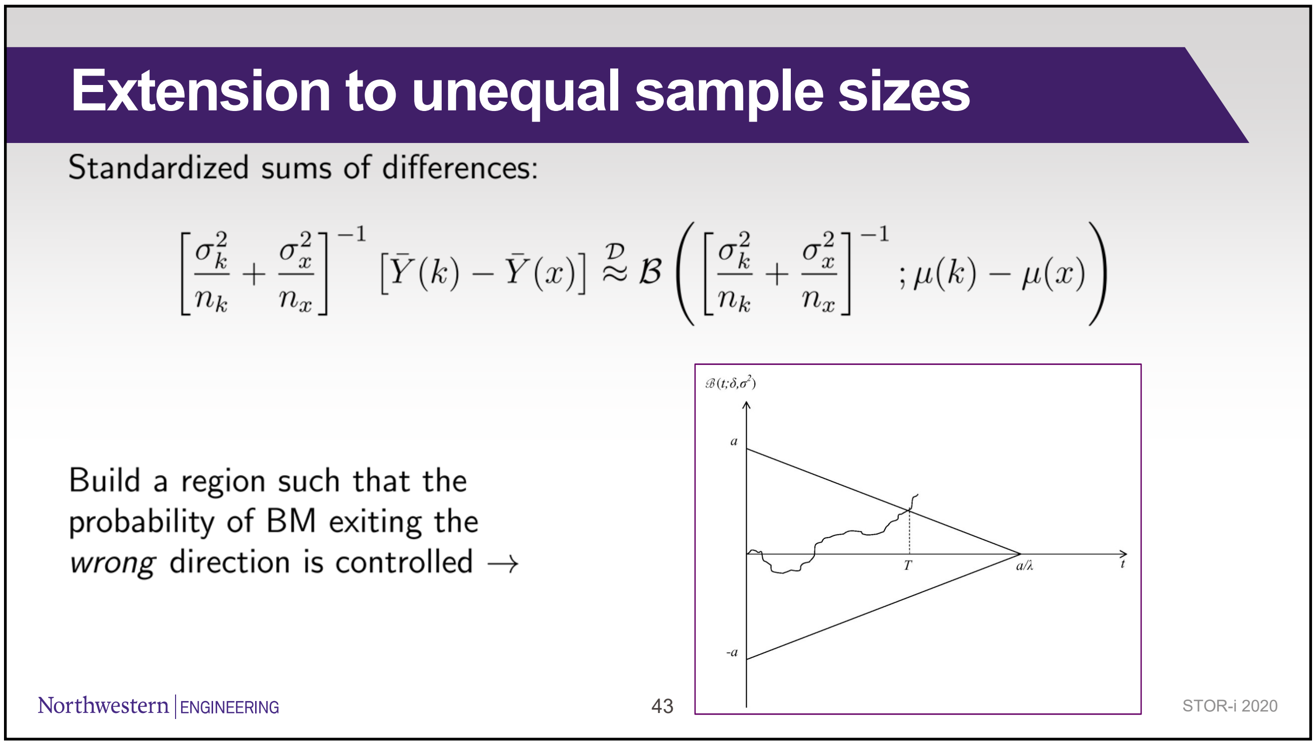
Theorem (Jennison, Johnson & Turnbull 1980):
Suppose $\delta > 0$, $g(t) \geq 0\ \forall t$ and continuous. Let

$$\begin{aligned} T_d &= \min\{r\colon |\mathcal{B}(r;\delta)| \geq g(r),\ r = 1, 2, \ldots\} \\ T_c &= \min\{t\colon |\mathcal{B}(t;\delta)| \geq g(t),\ t \geq 0\}. \end{aligned}$$



Then $T_c \leq T_d$ a.s. and $\Pr\{\mathcal{B}(T_d;\delta) \leq -g(T_d)\} \leq \Pr\{\mathcal{B}(T_c;\delta) \leq -g(T_c)\}$.

Northwestern | ENGINEERING 42 STOR-i 2020

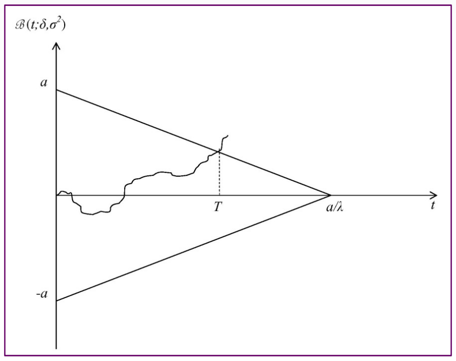


Extension to unequal sample sizes

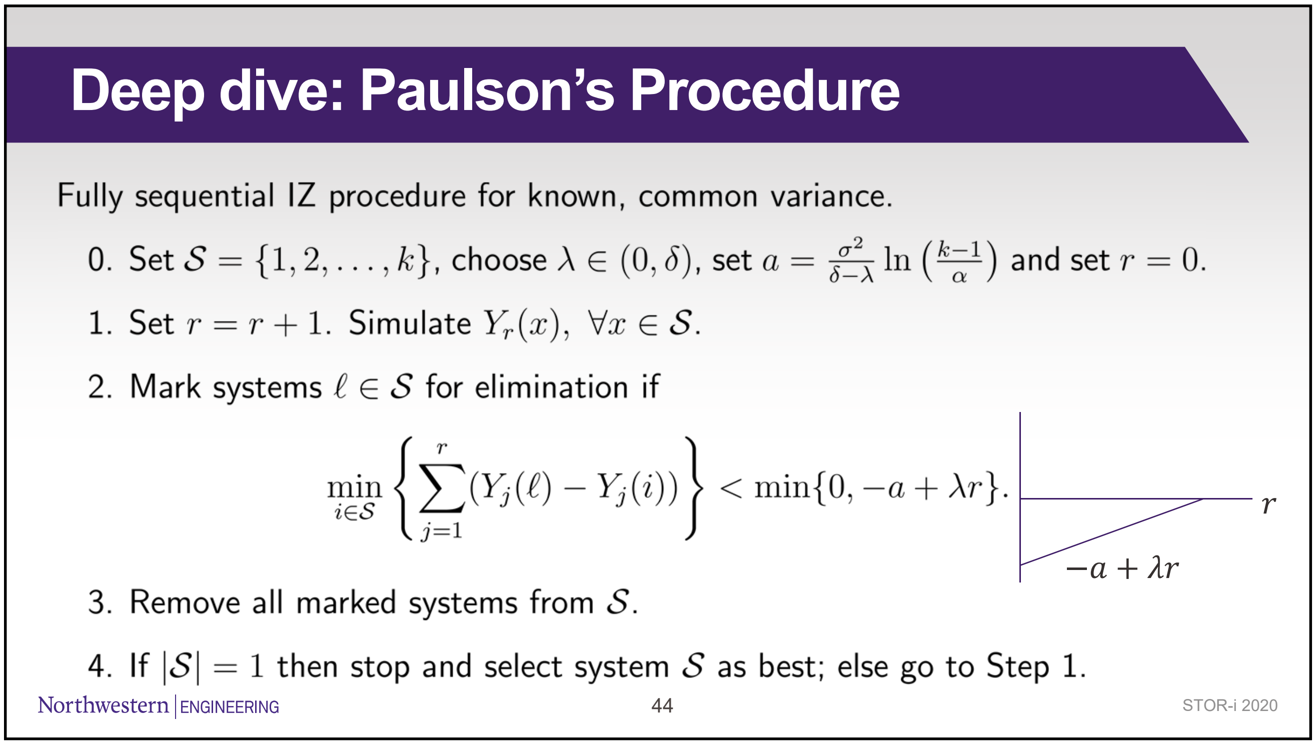
Standardized sums of differences:

$$\left[\frac{\sigma_k^2}{n_k} + \frac{\sigma_x^2}{n_x}\right]^{-1}\left[\bar{Y}(k) - \bar{Y}(x)\right] \overset{\mathcal{D}}{\approx} \mathcal{B}\left(\left[\frac{\sigma_k^2}{n_k} + \frac{\sigma_x^2}{n_x}\right]^{-1}; \mu(k) - \mu(x)\right)$$

Build a region such that the probability of BM exiting the *wrong* direction is controlled $\rightarrow$



Northwestern | ENGINEERING 43 STOR-i 2020

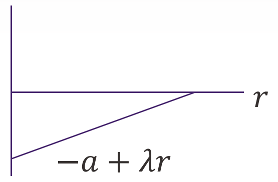


Deep dive: Paulson's Procedure

Fully sequential IZ procedure for known, common variance.

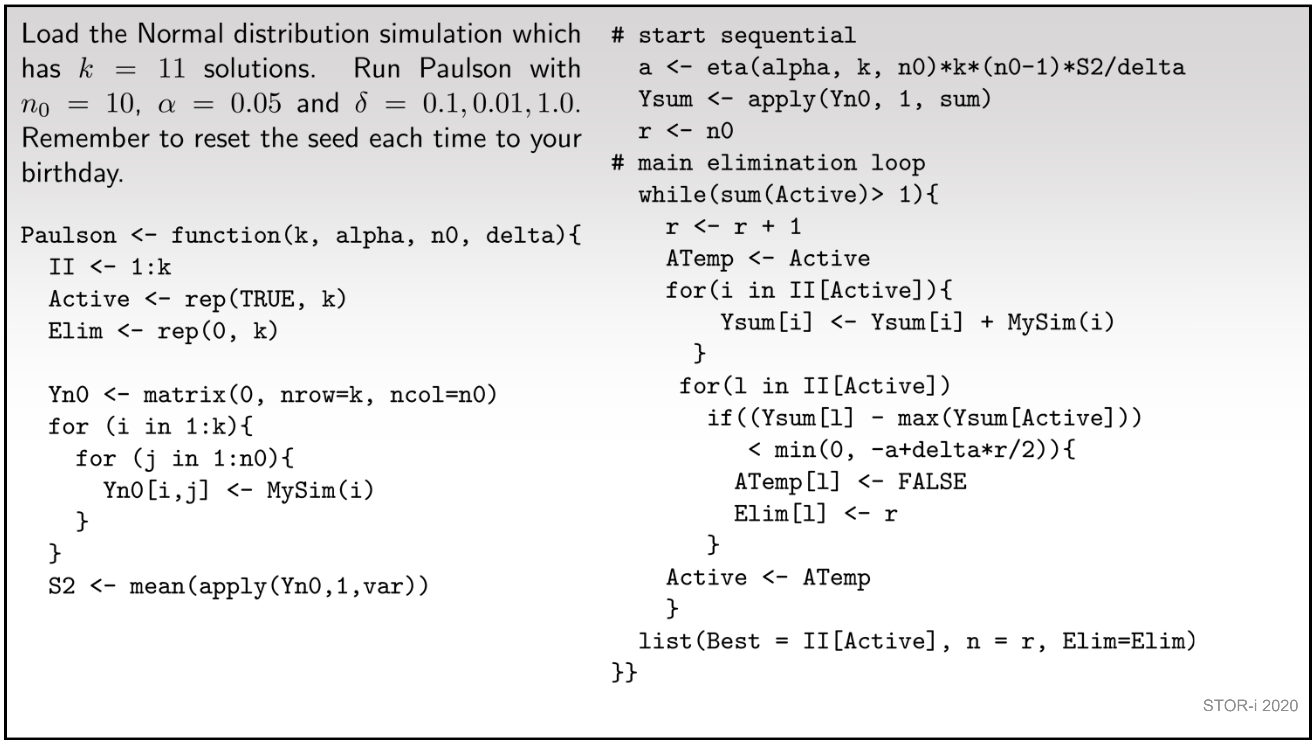
0. Set $\mathcal{S} = \{1, 2, \ldots, k\}$, choose $\lambda \in (0, \delta)$, set $a = \frac{\sigma^2}{\delta - \lambda} \ln\left(\frac{k-1}{\alpha}\right)$ and set $r = 0$.
1. Set $r = r + 1$. Simulate $Y_r(x),\ \forall x \in \mathcal{S}$.
2. Mark systems $\ell \in \mathcal{S}$ for elimination if

$$\min_{i \in \mathcal{S}} \left\{ \sum_{j=1}^{r} (Y_j(\ell) - Y_j(i)) \right\} < \min\{0, -a + \lambda r\}.$$



3. Remove all marked systems from $\mathcal{S}$.
4. If $|\mathcal{S}| = 1$ then stop and select system $\mathcal{S}$ as best; else go to Step 1.

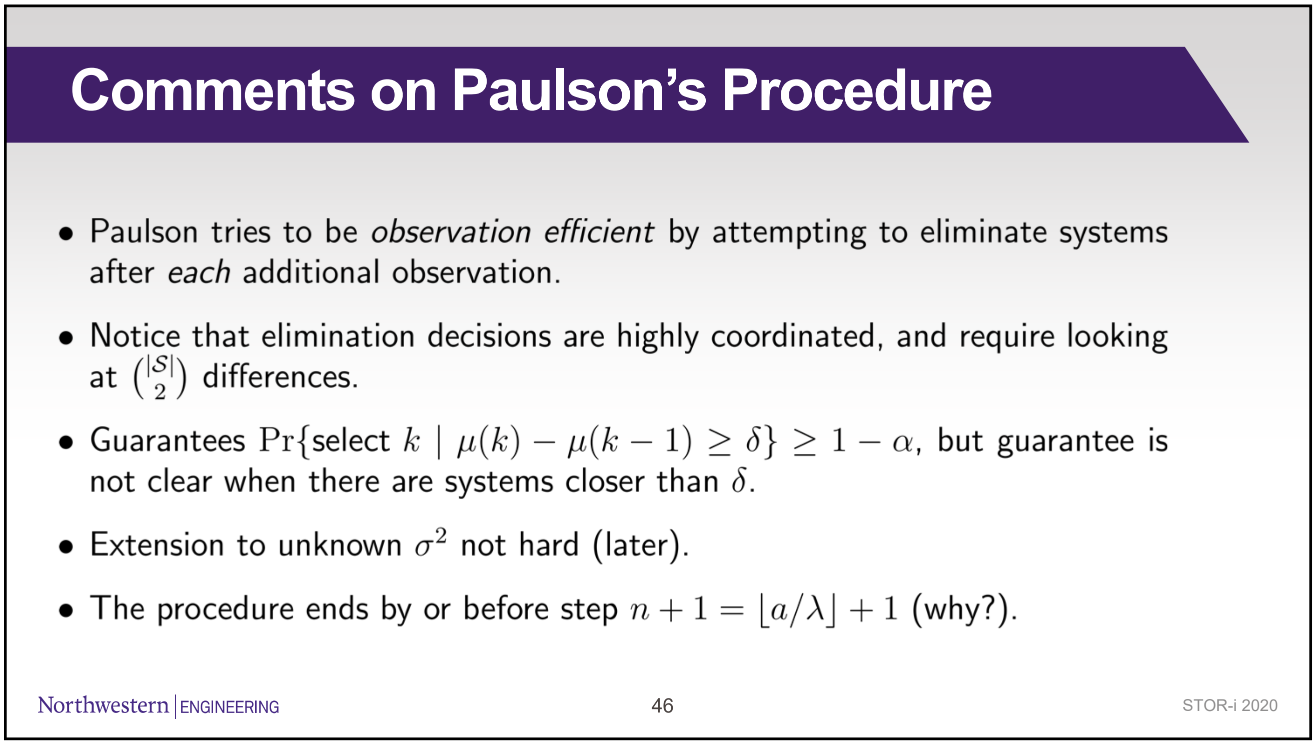
Northwestern | ENGINEERING 44 STOR-i 2020



Load the Normal distribution simulation which has $k = 11$ solutions. Run Paulson with $n_0 = 10$, $\alpha = 0.05$ and $\delta = 0.1, 0.01, 1.0$. Remember to reset the seed each time to your birthday.

```
Paulson <- function(k, alpha, n0, delta){
  II <- 1:k
  Active <- rep(TRUE, k)
  Elim <- rep(0, k)

  Yn0 <- matrix(0, nrow=k, ncol=n0)
  for (i in 1:k){
    for (j in 1:n0){
      Yn0[i,j] <- MySim(i)
    }
  }
  S2 <- mean(apply(Yn0,1,var))
# start sequential
  a <- eta(alpha, k, n0)*k*(n0-1)*S2/delta
  Ysum <- apply(Yn0, 1, sum)
  r <- n0
# main elimination loop
  while(sum(Active)> 1){
    r <- r + 1
    ATemp <- Active
    for(i in II[Active]){
        Ysum[i] <- Ysum[i] + MySim(i)
      }
     for(l in II[Active])
       if((Ysum[l] - max(Ysum[Active]))
          < min(0, -a+delta*r/2)){
         ATemp[l] <- FALSE
         Elim[l] <- r
       }
    Active <- ATemp
    }
  list(Best = II[Active], n = r, Elim=Elim)
}}
```

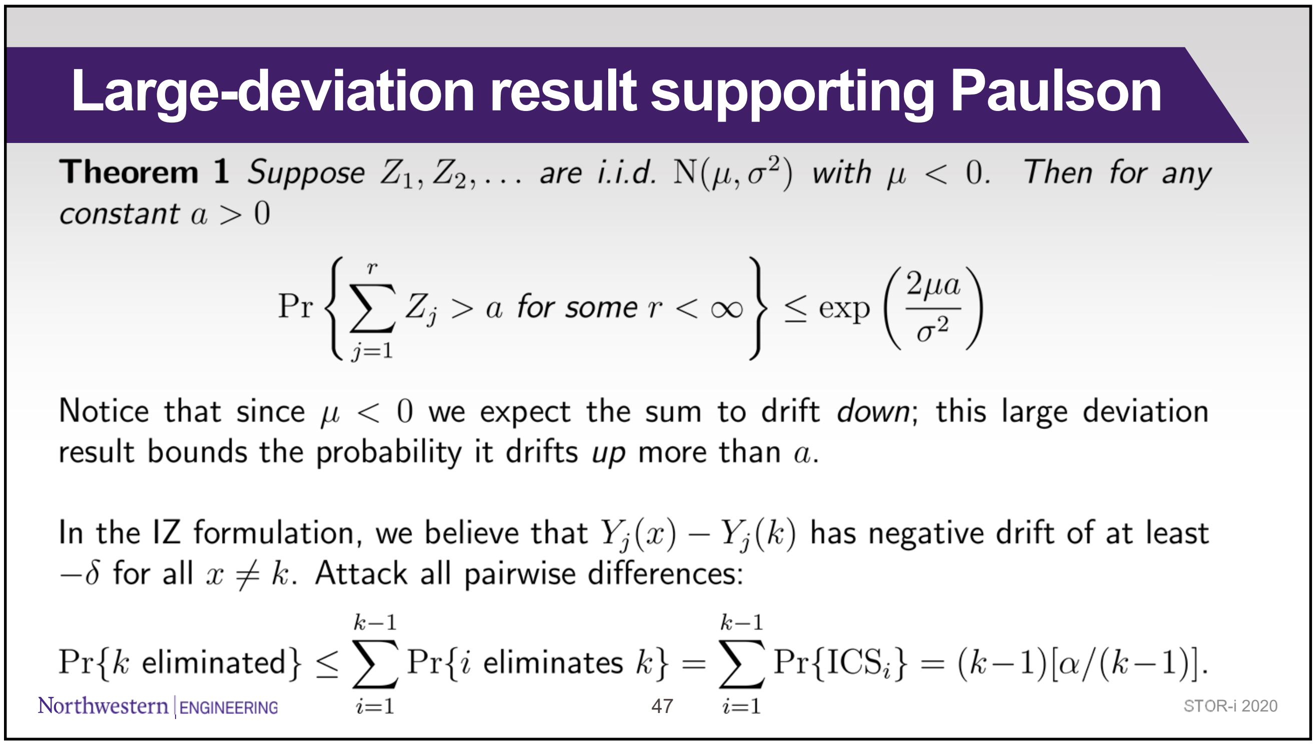
STOR-i 2020



Comments on Paulson’s Procedure

- Paulson tries to be *observation efficient* by attempting to eliminate systems after *each* additional observation.
- Notice that elimination decisions are highly coordinated, and require looking at $\binom{|\mathcal{S}|}{2}$ differences.
- Guarantees $\Pr\{\text{select } k \mid \mu(k) - \mu(k-1) \geq \delta\} \geq 1 - \alpha$, but guarantee is not clear when there are systems closer than δ.
- Extension to unknown σ^2 not hard (later).
- The procedure ends by or before step $n + 1 = \lfloor a/\lambda \rfloor + 1$ (why?).

Northwestern | ENGINEERING 46 STOR-i 2020



Large-deviation result supporting Paulson

Theorem 1 *Suppose* $Z_1, Z_2, \ldots$ *are i.i.d.* $\mathrm{N}(\mu, \sigma^2)$ *with* $\mu < 0$. *Then for any constant* $a > 0$

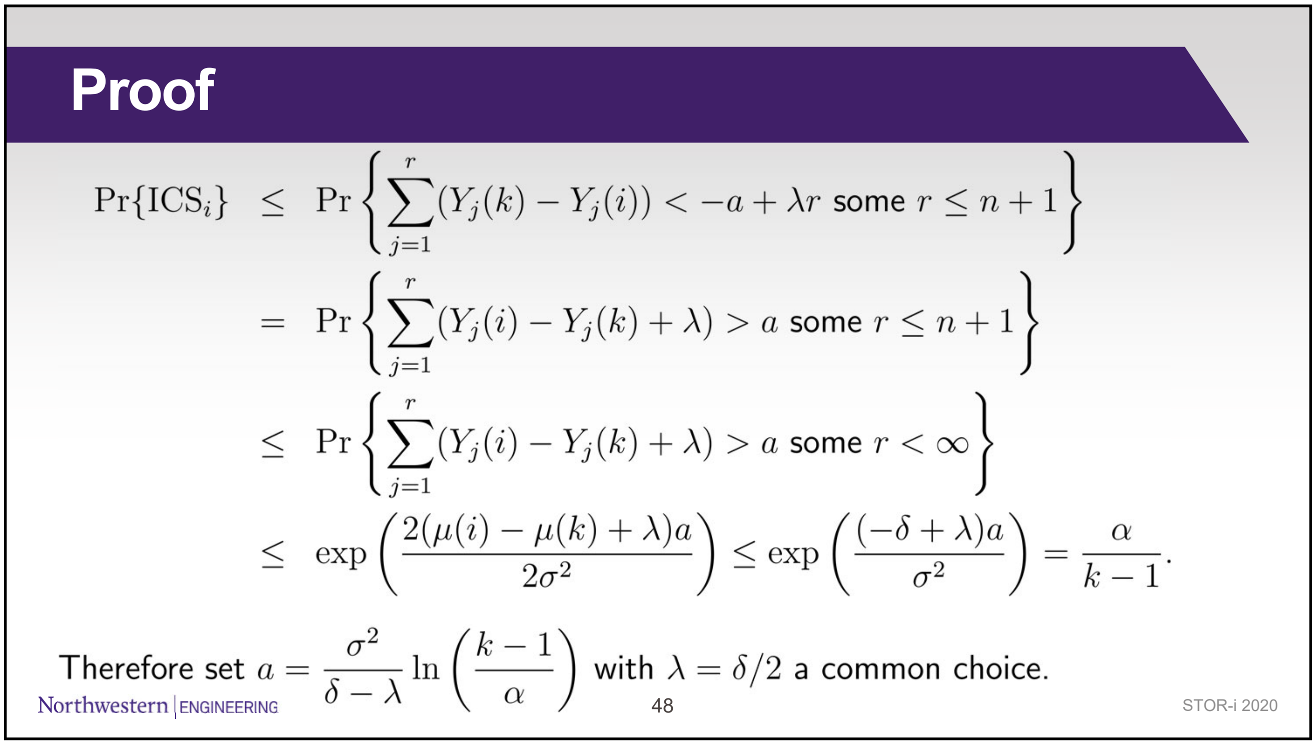
$$\Pr\left\{\sum_{j=1}^{r} Z_j > a \textit{ for some } r < \infty\right\} \leq \exp\left(\frac{2\mu a}{\sigma^2}\right)$$

Notice that since $\mu < 0$ we expect the sum to drift *down*; this large deviation result bounds the probability it drifts *up* more than a.

In the IZ formulation, we believe that $Y_j(x) - Y_j(k)$ has negative drift of at least $-\delta$ for all $x \neq k$. Attack all pairwise differences:

$$\Pr\{k \text{ eliminated}\} \leq \sum_{i=1}^{k-1} \Pr\{i \text{ eliminates } k\} = \sum_{i=1}^{k-1} \Pr\{\mathrm{ICS}_i\} = (k-1)[\alpha/(k-1)].$$

Northwestern | ENGINEERING 47 STOR-i 2020

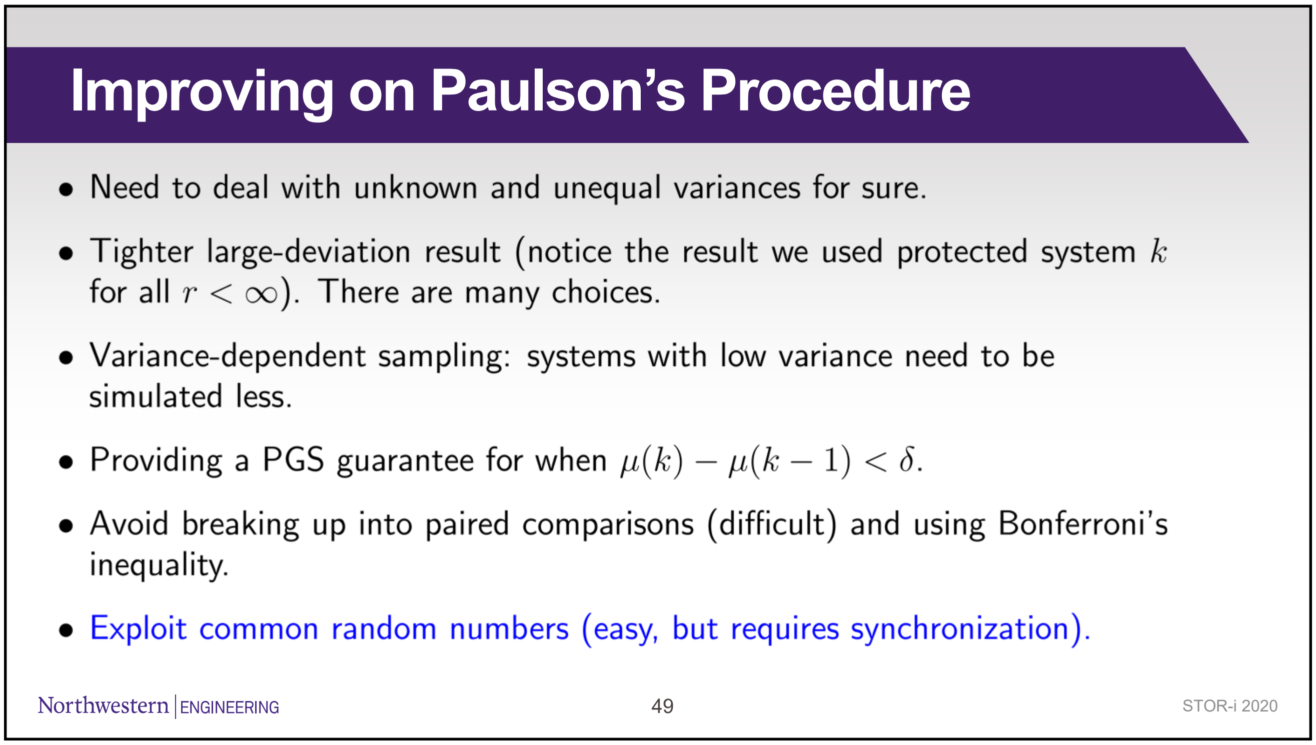


Proof

$$\begin{aligned}
\Pr\{\mathrm{ICS}_i\} &\leq \Pr\left\{\sum_{j=1}^{r}(Y_j(k)-Y_j(i)) < -a+\lambda r \text{ some } r \leq n+1\right\} \\
&= \Pr\left\{\sum_{j=1}^{r}(Y_j(i)-Y_j(k)+\lambda) > a \text{ some } r \leq n+1\right\} \\
&\leq \Pr\left\{\sum_{j=1}^{r}(Y_j(i)-Y_j(k)+\lambda) > a \text{ some } r < \infty\right\} \\
&\leq \exp\left(\frac{2(\mu(i)-\mu(k)+\lambda)a}{2\sigma^2}\right) \leq \exp\left(\frac{(-\delta+\lambda)a}{\sigma^2}\right) = \frac{\alpha}{k-1}.
\end{aligned}$$

Therefore set $a = \dfrac{\sigma^2}{\delta-\lambda}\ln\left(\dfrac{k-1}{\alpha}\right)$ with $\lambda = \delta/2$ a common choice.

Northwestern | ENGINEERING 48 STOR-i 2020



Improving on Paulson's Procedure

- Need to deal with unknown and unequal variances for sure.
- Tighter large-deviation result (notice the result we used protected system k for all $r < \infty$). There are many choices.
- Variance-dependent sampling: systems with low variance need to be simulated less.
- Providing a PGS guarantee for when $\mu(k) - \mu(k-1) < \delta$.
- Avoid breaking up into paired comparisons (difficult) and using Bonferroni's inequality.
- Exploit common random numbers (easy, but requires synchronization).

Northwestern | ENGINEERING 49 STOR-i 2020

Common random numbers

R&S procedures that employ pairwise comparisons can often be "sharpened" by using CRN:

$$\mathrm{Var}(Y(x) - Y(x')) = \mathrm{Var}(Y(x)) + \mathrm{Var}(Y(x')) - 2\,\mathrm{Cov}(Y(x), Y(x'))$$

CRN tends to make $\mathrm{Cov}(Y(x), Y(x')) > 0$, but usually requires equal sample sizes.

Intuition:
In the inventory problem, CRN implies each (s, S) policy sees *exactly the same sequence of demands.*

In the TTF problem excessively short component failure times times occur in the same sequence: $F^{-1}(U_r; \lambda_x) = -\ln(1 - U_r)/\lambda_x$, $r = 1, 2, \ldots$.

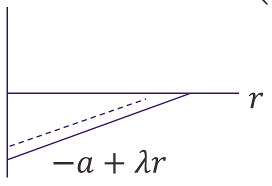
CRN effect

Impact on subset:

$$W(x, x') = \left(t(x)^2 \frac{S^2(x)}{n(x)} + t(x')^2 \frac{S^2(x')}{n(x')} \right)^{1/2} \text{ becomes } W(x, x') = \left(t^2 \frac{S^2(x, x')}{n} \right)^{1/2}$$

Impact on Paulson (equal, known variance σ^2 & correlation $\rho > 0$):

$$a = \frac{\sigma^2(1-\rho)}{\delta - \lambda} \ln\left(\frac{k-1}{\alpha}\right) \text{ rather than } a = \frac{\sigma^2}{\delta - \lambda} \ln\left(\frac{k-1}{\alpha}\right)$$



CRN and R

set.seed(12345) maps to a starting seed, but we have no idea if set.seed(1) and set.seed(2) are near or far apart in the random number sequence.

Simulation languages have "streams" that map to starting seeds that are *very* far apart; thus, we can assign a unique stream to each random process and replication.

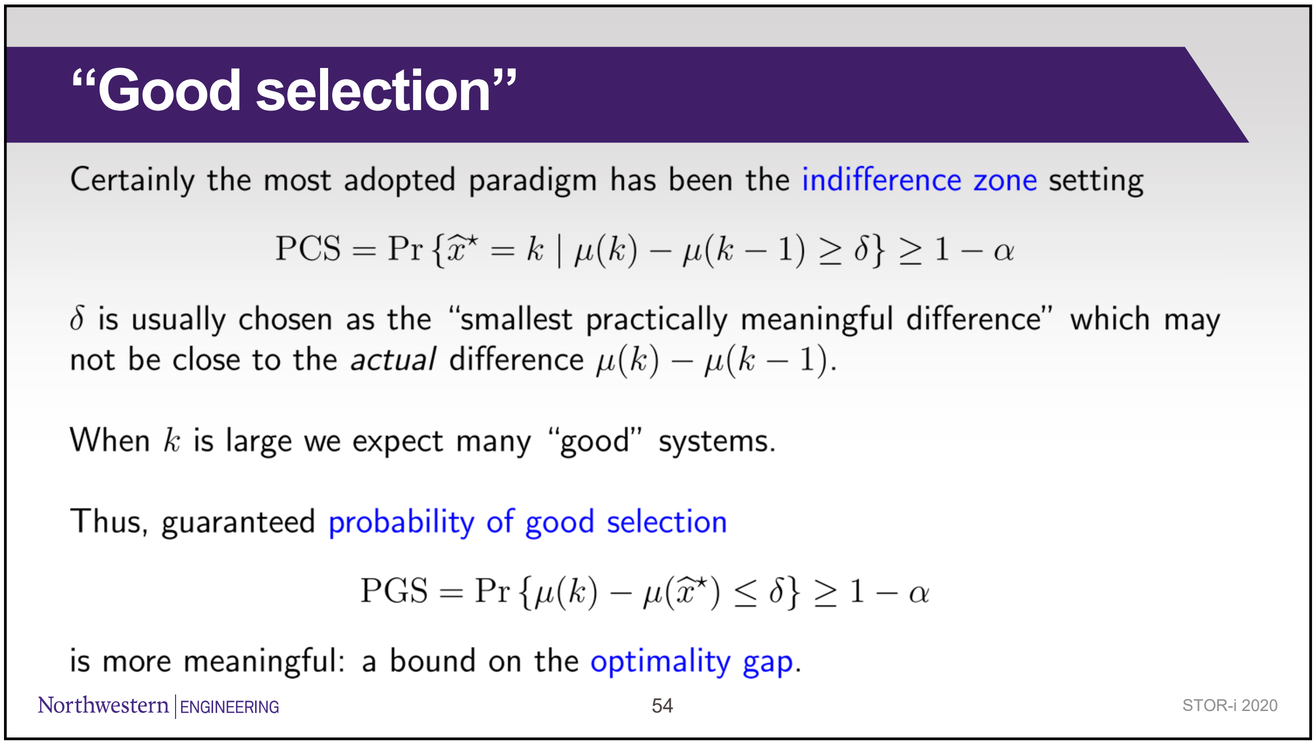
R is better at matrix operations than loops, so compute variance of difference as

```
S2 <- cov(Yall)/n                # var-cov matrix of sample means

S2[i,i] + S2[j,j] - 2*S2[i,j]  # var(Ybar[i] - Ybar[j])
```

Subset procedure with CRN

```
SubsetCRN <- function(k, alpha, n, seed){
  Yall <- NULL
  for (x in 1:k){
    Yall <- cbind(Yall, MySim(x, n, seed))}
  Ybar <- apply(Yall, 2, mean)
  S2 <- cov(Yall)/n
  tval <- qt(1-alpha/(k-1), df = n-1)
  Subset <- 1:k
  for (i in 1:k){
    for (j in 1:k){
      if (Ybar[i] < (Ybar[j]-tval*sqrt(S2[i,i] + S2[j,j] - 2*S2[i,j]))){
        Subset[i] <- 0
        break
      }
    }
  }
  list(Subset = Subset[Subset != 0], Ybar = Ybar, S2 = S2, corr=cor(Yall))
}
```

Go back and run SubsetCRN on the same M/M/1 problem, and compare the size of your subsets to your previous results. Use your birthday as your seed.



"Good selection"

Certainly the most adopted paradigm has been the indifference zone setting

$$\text{PCS} = \Pr\{\widehat{x}^\star = k \mid \mu(k) - \mu(k-1) \geq \delta\} \geq 1 - \alpha$$

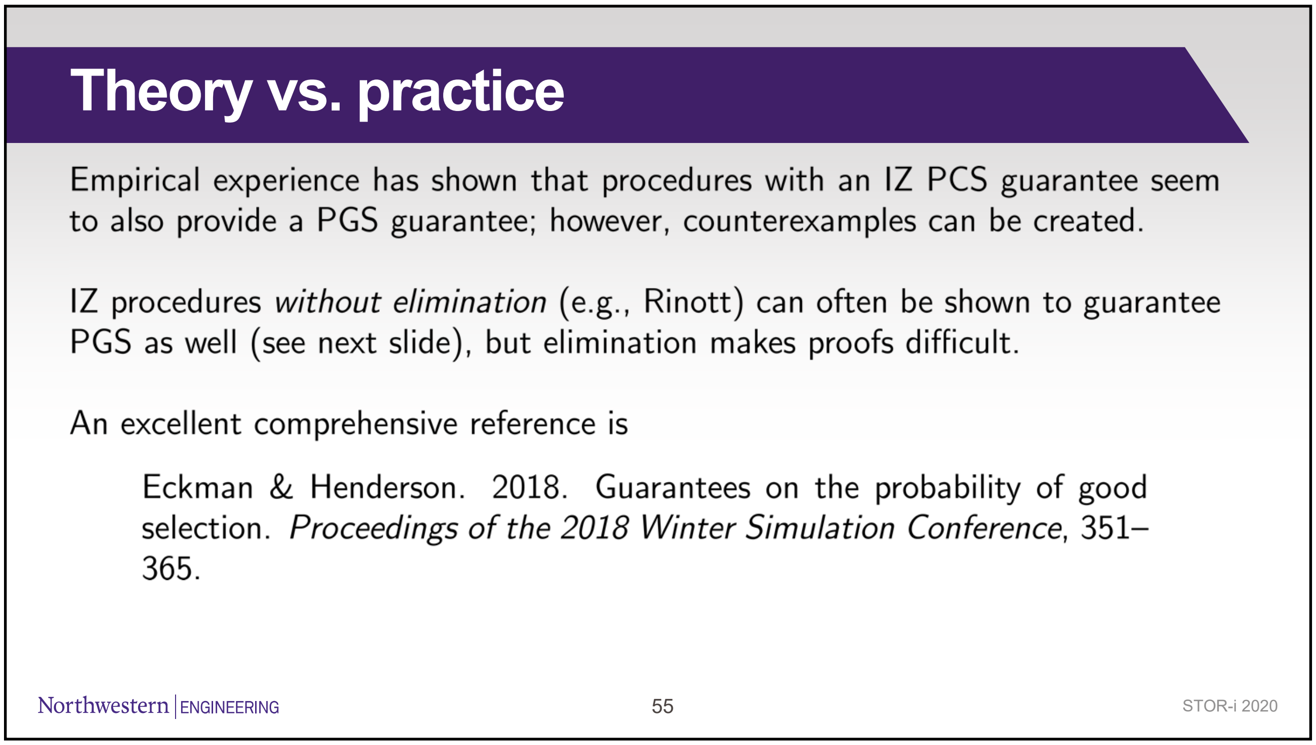
δ is usually chosen as the "smallest practically meaningful difference" which may not be close to the *actual* difference $\mu(k) - \mu(k-1)$.

When k is large we expect many "good" systems.

Thus, guaranteed probability of good selection

$$\text{PGS} = \Pr\{\mu(k) - \mu(\widehat{x}^\star) \leq \delta\} \geq 1 - \alpha$$

is more meaningful: a bound on the optimality gap.



Theory vs. practice

Empirical experience has shown that procedures with an IZ PCS guarantee seem to also provide a PGS guarantee; however, counterexamples can be created.

IZ procedures *without elimination* (e.g., Rinott) can often be shown to guarantee PGS as well (see next slide), but elimination makes proofs difficult.

An excellent comprehensive reference is

> Eckman & Henderson. 2018. Guarantees on the probability of good selection. *Proceedings of the 2018 Winter Simulation Conference*, 351–365.

Nelson & Matejcik (1995) condition

Theorem: *Suppose a R&S procedure creates estimators* $\widehat{\mu}(1), \widehat{\mu}(2), \ldots, \widehat{\mu}(k)$ *that guarantee* $\Pr\{\widehat{\mu}(k) > \widehat{\mu}(i),\ \forall i \neq k \mid \mu(k) - \mu(k-1) \geq \delta\} \geq 1 - \alpha$. *Then if*

$$\begin{pmatrix} \widehat{\mu}(k) \\ \widehat{\mu}(k-1) - \mu(k-1) + (\mu(k) - \delta) \\ \vdots \\ \widehat{\mu}(1) - \mu(1) + (\mu(k) - \delta) \end{pmatrix}$$

has the same distribution as estimators <u>*would have had*</u> *in the corresponding LFC problem, then the procedure also guarantees* $\mathrm{PGS} \geq 1 - \alpha$.

Normally distributed output procedures like Rinott that do not adapt to the sample means satisfy this.

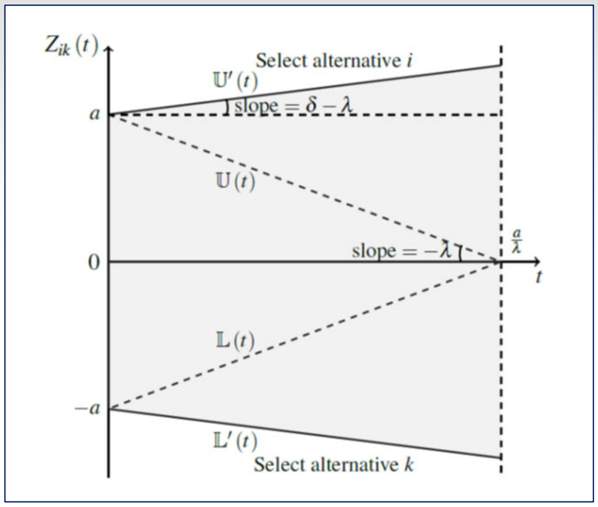
Zhong & Hong (2018) Paulson adjustment

Recall Paulson eliminates ℓ if for some i $\sum_{j=1}^{r}(Y_j(\ell) - Y_j(i)) < -a + \lambda r$.

Instead, Zhong & Hong use $\sum_{j=1}^{r}(Y_j(\ell) - Y_j(i) + \delta) < -a + \lambda r$.

Notice that when $\mu(k) - \mu(\ell) < \delta$, $\sum_{j=1}^{r}(Y_j(\ell) - Y_j(k) + \delta)$ has positive drift.

Thus, good systems should survive to the end, and we will pick the best looking one.



A Bayesian perspective on good solutions

A Bayesian "good selection" R&S procedure would stop when it has collected enough output so that there is a system $\widehat{x}^\star$ for which

$$\Pr\{M(\widehat{x}^\star) > M(x) - \delta,\ \forall x \neq \widehat{x}^\star \mid \mathcal{H}\} \geq 1 - \alpha$$

[Computable under some assumptions, but if not then can be approximated or bounded.]

Interpretation: *With probability at least $1-\alpha$ the* **random problem** *from your space of priors is one for which the* **fixed system** $\widehat{x}^\star$ *is good.*

This contrasts with the frequentist perspective: The **random solution** $\widehat{x}^\star$ chosen by the procedure has probability at least $1-\alpha$ of being good for this **fixed problem**.

Unknown variances

- As a general rule, neither known nor equal variances can be assumed in simulation R&S problems.
  - For procedures that break into pairwise differences and apply Bonferroni, the variance of each pairwise difference can be estimated separately.
- A useful result: If $Z_1, Z_2, \ldots, Z_{n_0}$ are i.i.d. $\mathrm{N}(\mu, \sigma^2)$ then $\bar{Z} \perp S^2$.
  - Thus, using a "first-stage" S^2 to calibrate the additional simulation needed does not introduce bias.
  - If done cleverly, we can derive the PCS *conditional* on S^2 and then uncondition.
  - Using estimated σ^2 increases E(sample size) relative to known variance.

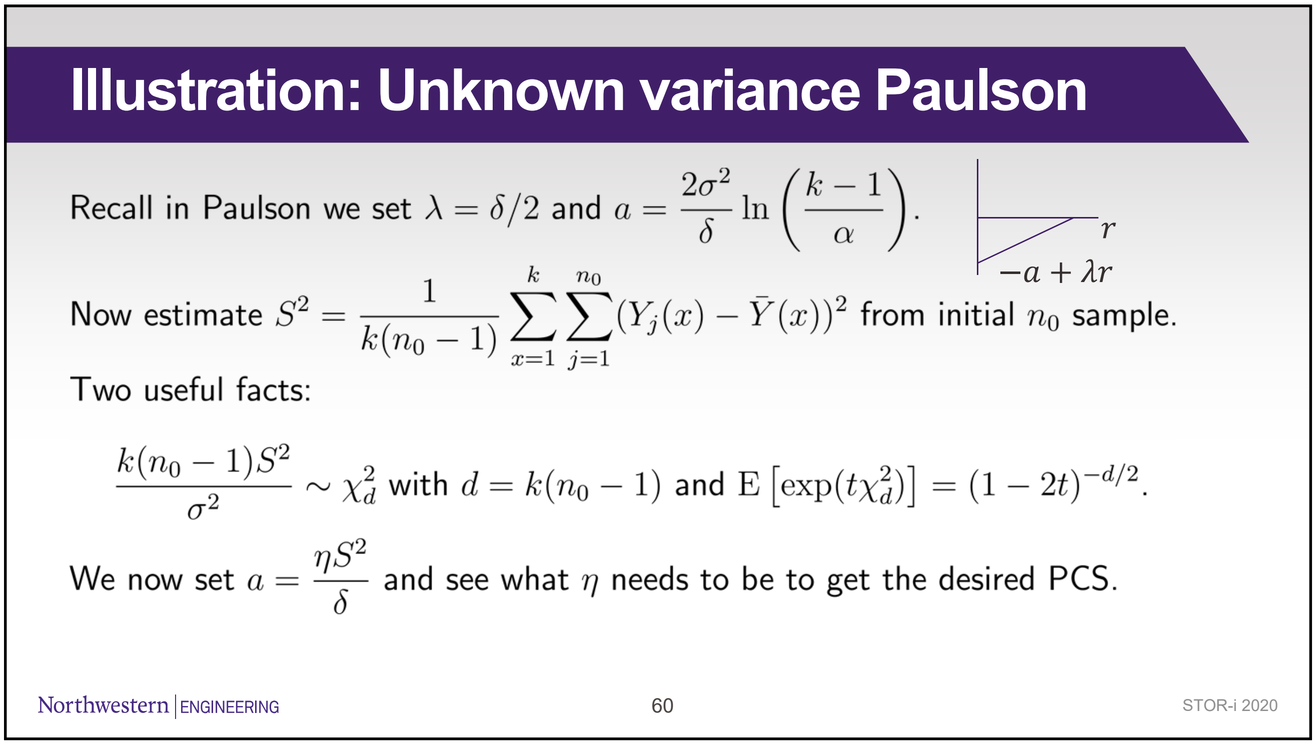
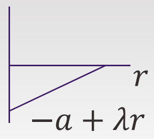


Illustration: Unknown variance Paulson

Recall in Paulson we set $\lambda = \delta/2$ and $a = \frac{2\sigma^2}{\delta} \ln\left(\frac{k-1}{\alpha}\right)$.



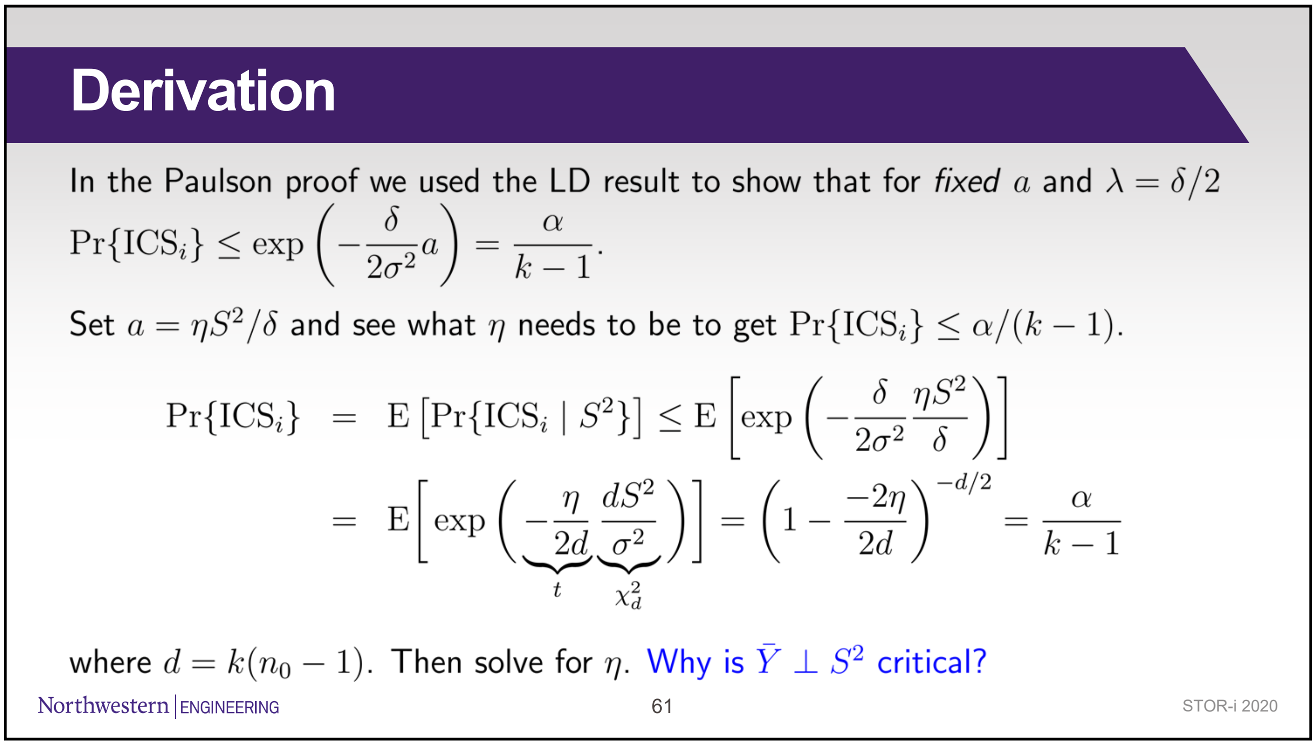
Now estimate $S^2 = \frac{1}{k(n_0 - 1)} \sum_{x=1}^{k} \sum_{j=1}^{n_0} (Y_j(x) - \bar{Y}(x))^2$ from initial n_0 sample.

Two useful facts:

$$\frac{k(n_0-1)S^2}{\sigma^2} \sim \chi^2_d \text{ with } d = k(n_0 - 1) \text{ and } \mathrm{E}\left[\exp(t\chi^2_d)\right] = (1-2t)^{-d/2}.$$

We now set $a = \frac{\eta S^2}{\delta}$ and see what η needs to be to get the desired PCS.

Northwestern | ENGINEERING 60 STOR-i 2020



Derivation

In the Paulson proof we used the LD result to show that for *fixed* a and $\lambda = \delta/2$

$$\Pr\{\mathrm{ICS}_i\} \le \exp\left(-\frac{\delta}{2\sigma^2}a\right) = \frac{\alpha}{k-1}.$$

Set $a = \eta S^2/\delta$ and see what η needs to be to get $\Pr\{\mathrm{ICS}_i\} \le \alpha/(k-1)$.

$$\begin{aligned}\Pr\{\mathrm{ICS}_i\} &= \mathrm{E}\left[\Pr\{\mathrm{ICS}_i \mid S^2\}\right] \le \mathrm{E}\left[\exp\left(-\frac{\delta}{2\sigma^2}\frac{\eta S^2}{\delta}\right)\right] \\ &= \mathrm{E}\left[\exp\left(-\underbrace{\frac{\eta}{2d}}_{t}\underbrace{\frac{dS^2}{\sigma^2}}_{\chi^2_d}\right)\right] = \left(1 - \frac{-2\eta}{2d}\right)^{-d/2} = \frac{\alpha}{k-1}\end{aligned}$$

where $d = k(n_0 - 1)$. Then solve for η. Why is $\bar{Y} \perp S^2$ critical?

Northwestern | ENGINEERING 61 STOR-i 2020

Beyond Paulson…

Paulson is great for illustrating concepts, but the limitation to equal variances and no common random numbers makes it rarely used in simulation.

There are many descendants, with one of the most statistically efficient and robust still being KN (Kim and N 2001).

- Uses a tighter Brownian motion LD result due to Fabian.
- Allows unequal variances and CRN.
- Has been shown to be asymptotically valid (discussed later) for non-normal output data.
- Has been implemented in commercial simulation languages, and in parallel.

Trying out KN

Apply Paulson and KN to the M/M/1 problem with $n_0 = 30$, $\delta = 1$ and $\alpha = 0.05$. Compare the chosen solution, elimination points, and final sample size.

```
result <- KN(100, 0.05, 30, 1)
```

Remember to set the seed to your birthday before running each experiment.

Note: Paulson is not technically valid for this problem because the variances are unequal.

Extra: Outside of this class, try both with $\delta = 0.1$ which is actually more reasonable for this problem. Leave Paulson to run over night!

A note on asymptotic analysis

Asymptotic analysis of R&S procedures is useful in at least three contexts:

1. Establishing that a procedure will work when core assumptions such as normality are violated (typically as $\delta \to 0$ in a way that makes the problem harder).
2. Comparing the efficiency of procedures that are difficult to evaluate in finite samples (typically as $1 - \alpha \to 1$ so that behavior becomes deterministic).
3. Comparing the efficiency of procedures with estimated variances relative to their known-variance counterparts (typically as $\delta \to 0$ drives $n_0 \to \infty$).

#1 helps explain why normal-theory procedures seem to work well more generally. #2 is often the only way (other than empirically) to compare procedures that eliminate systems.

Asymptotic PCS for IZ procedures

Show desired PCS is achieved in a *meaningful limit*, even if assumptions violated.

Pointless: If $\mu(k) - \mu(i)$ is **fixed**, then as we let $\delta \to 0$ we have $\mathrm{PCS} \to 1$ for any kind of data. Why? (remember $N_i \propto 1/\delta^2$ for many procedures)

Useful: Kim and N (2006) let $\mu(k) = \mu$ and $\mu(i) = \mu - \delta$ for $i \neq k$.

Notice that as $\delta \to 0$ the sample size goes to ∞ and the problem itself gets harder.

Is this a relevant setting? Yes. If $\delta \gg \mu(k) - \mu(i)$ then any solution is acceptable. If $\delta \ll \mu(k) - \mu(i)$ then we will simulate so much we will get it right. Thus $\mu(i) = \mu(k) - \delta$ is the critical regime.

Key tool for asymptotic PCS

Donsker's (Functional Central Limit) Theorem: If $Y_1, Y_2, \ldots$ are i.i.d. (μ, σ^2) with $\sigma^2 < \infty$ then as $N \to \infty$

$$\frac{\sum_{j=1}^{\lfloor Nt \rfloor} Y_j - Nt\mu}{\sigma\sqrt{N}} \xrightarrow{\mathcal{D}} \mathcal{B}(t),\ 0 \le t \le 1$$

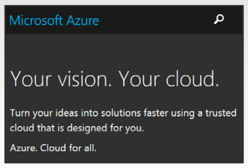
Note: The regular CLT is at $t = 1$.

Donsker's Theorem says that very general i.i.d. output processes, standardized the right way, look like Brownian motion as we get more and more data.

In IZ R&S procedures, $Y_j = (Y_j(x) - Y_j(x'))$, and letting $\delta \to 0$ drives the sample size to ∞ $(N \propto 1/\delta^2)$.

The future is now: Parallel R&S

- Simulation languages are being redesigned to run in the cloud.
  - Computer time is "rented."
  - Example: Simio can recruit up to 10,000 processors from Microsoft Azure; this greatly extends the R&S limit.

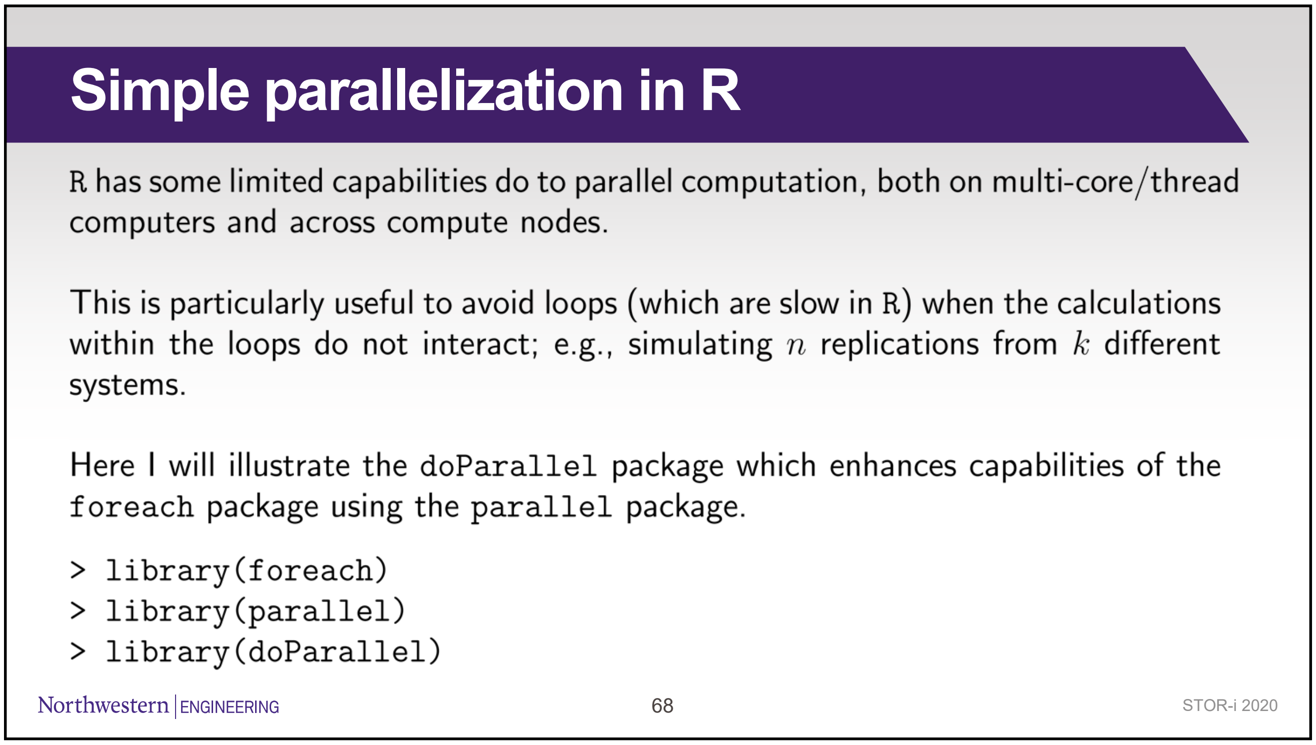

Microsoft Azure

Your vision. Your cloud.

Turn your ideas into solutions faster using a trusted cloud that is designed for you.

Azure. Cloud for all.

- Since we have to pay for the service, the focus changes from being observation efficient to being computationally efficient in wall clock time.
  - Ok to waste observations to avoid idling and get done faster.
- Heterogeneous processors, communication delays, processor failures, etc. disrupt the usual synchronization in R&S.



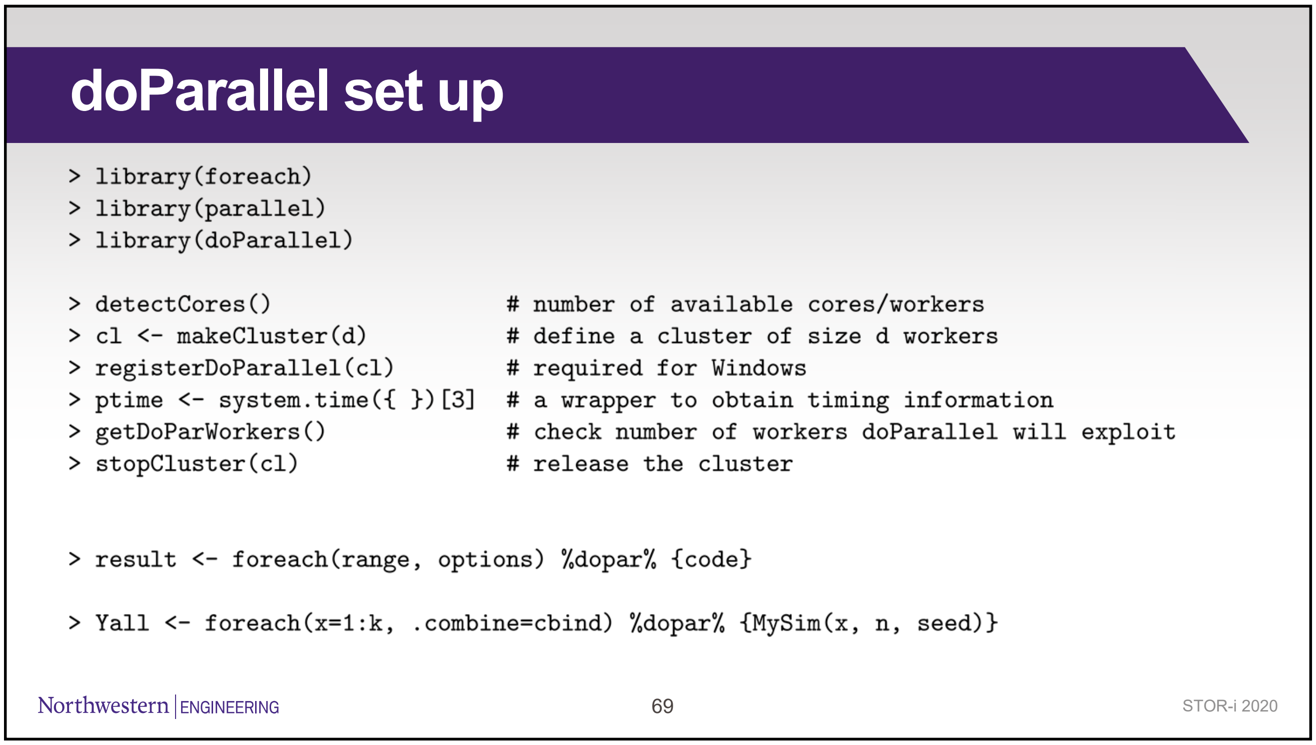
Simple parallelization in R

R has some limited capabilities do to parallel computation, both on multi-core/thread computers and across compute nodes.

This is particularly useful to avoid loops (which are slow in R) when the calculations within the loops do not interact; e.g., simulating n replications from k different systems.

Here I will illustrate the `doParallel` package which enhances capabilities of the `foreach` package using the `parallel` package.

```
> library(foreach)
> library(parallel)
> library(doParallel)
```



doParallel set up

```
> library(foreach)
> library(parallel)
> library(doParallel)

> detectCores()                  # number of available cores/workers
> cl <- makeCluster(d)           # define a cluster of size d workers
> registerDoParallel(cl)         # required for Windows
> ptime <- system.time({ })[3]   # a wrapper to obtain timing information
> getDoParWorkers()              # check number of workers doParallel will exploit
> stopCluster(cl)                # release the cluster


> result <- foreach(range, options) %dopar% {code}

> Yall <- foreach(x=1:k, .combine=cbind) %dopar% {MySim(x, n, seed)}
```

Timing comparison

Reload the TTF problem. Compare `SubsetCRN` to `SubsetParallel` using 4 cores and 100 replications, in terms of timing; if you use the same seed, they should get the same answer. The only difference between the two functions is

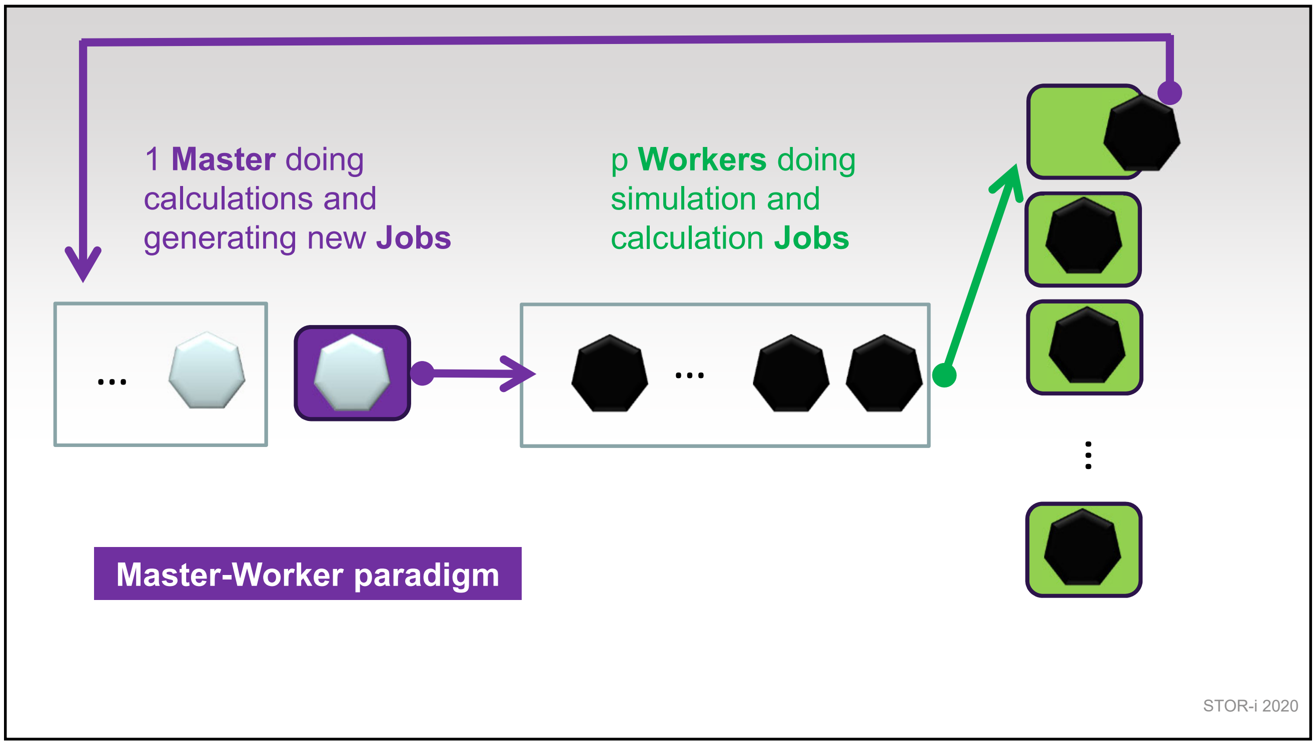
```
Yall <- foreach(x=1:k, .combine=cbind) %dopar% {MySim(x, n, seed)}
```

replaces

```
for (x in 1:k){Yall <- cbind(Yall, MySim(x, n))}
```

Example calling sequence:

```
ptime<-system.time({result<-SubsetParallel(4,0.05,100,12211956)})[3]
```


1 Master doing calculations and generating new Jobs
p Workers doing simulation and calculation Jobs
…
…
⋮

Master-Worker paradigm

What would be affected if we did a direct parallelization of Paulson with 1 master and p workers?

```
Paulson <- function(k, alpha, n0, delta){
  II <- 1:k                                  ┐
  Active <- rep(TRUE, k)                     ├ master
  Elim <- rep(0, k)                          ┘

  Yn0 <- matrix(0, nrow=k, ncol=n0)          ┐
  for (i in 1:k){                            │ p workers
    for (j in 1:n0){                         │ in parallel
      Yn0[i,j] <- MySim(i)                   │
    }                                        ┘
  }
  S2 <- mean(apply(Yn0,1,var))               ├ master
# start sequential                           ┐
  a <- eta(alpha, k, n0)*k*(n0-1)*S2/delta   │
  Ysum <- apply(Yn0, 1, sum)                 │ master
  r <- n0                                    │
# main elimination loop                      │
  while(sum(Active)> 1){                     ┘
    r <- r + 1                               ┐
    ATemp <- Active                          │ p workers
    for(i in II[Active]){                    │ in parallel
        Ysum[i] <- Ysum[i] + MySim(i)        │
    }                                        ┘
     for(l in II[Active])                    ┐
       if((Ysum[l] - max(Ysum[Active]))      │
          < min(0, -a+delta*r/2)){           │ master
         ATemp[l] <- FALSE                   │
         Elim[l] <- r                        │
       }                                     ┘
    Active <- ATemp
    }
  list(Best = II[Active], n = r, Elim=Elim)
}}
```

Thinking about R&S *computations*

Job j is the ordered list

$$J_j \equiv \{\underbrace{(\mathcal{Q}_j, \Delta_j, \mathcal{U}_j)}_{\text{simulate}}, \underbrace{(\mathcal{P}_j, \mathcal{C}_j)}_{\text{calculate}}\}.$$

Hunter & N, "Parallel ranking and selection," *Advances in Modeling and Simulation*, Springer, 2017.

- $\mathcal{Q}_j \subseteq \{1, 2, \ldots, k\}$ indices of systems to be simulated
- $\Delta_j = \{\Delta_{xj}\}$ how many replications to take from each system $x \in \mathcal{Q}_j$
- $\mathcal{U}_j$ (optional) the assigned block of random numbers
- $\mathcal{C}_j$ is a list of non-simulation calculations or operations to perform
- $\mathcal{P}_j$ is a list of jobs that must complete before the calculation $\mathcal{C}_j$

The nominal computational paradigm

For $\ell = 1, 2, \ldots$

1. *Simulation jobs*

$$\mathcal{J}_\ell = [\{(\text{system } 1, 1 \text{ rep}), (\emptyset)\}, \ldots, \{(\text{system } i, 1 \text{ rep}), (\emptyset)\}, \ldots]$$

2. *Comparison jobs*

$$J'_\ell = \{(\emptyset), (\text{all jobs in } \mathcal{J}_\ell, \mathcal{C}_\ell)\}$$

where $\mathcal{C}_\ell$ performs calculations on all (non-eliminated) systems.

The nominal model enforces many of the assumptions necessary for both small-sample and asymptotic analysis by "synchronized coupling."

Looking at parallel Paulson as Calculation and Simulation jobs.

```
Paulson <- function(k, alpha, n0, delta){
  II <- 1:k
  Active <- rep(TRUE, k)
  Elim <- rep(0, k)
```

Calculation job

```
  Yn0 <- matrix(0, nrow=k, ncol=n0)
  for (i in 1:k){
    for (j in 1:n0){
      Yn0[i,j] <- MySim(i)
    }
  }
```

Simulation jobs

```
  S2 <- mean(apply(Yn0,1,var))
```

Calculation job

```
# start sequential
  a <- eta(alpha, k, n0)*k*(n0-1)*S2/delta
  Ysum <- apply(Yn0, 1, sum)
  r <- n0
# main elimination loop
  while(sum(Active)> 1){
```

Calculation job

```
    r <- r + 1
    ATemp <- Active
    for(i in II[Active]){
        Ysum[i] <- Ysum[i] + MySim(i)
     }
```

Simulation jobs

```
     for(l in II[Active])
       if((Ysum[l] - max(Ysum[Active]))
          < min(0, -a+delta*r/2)){
         ATemp[l] <- FALSE
         Elim[l] <- r
       }
```

Calculation job

```
    Active <- ATemp
    }
  list(Best = II[Active], n = r, Elim=Elim)
}}
```

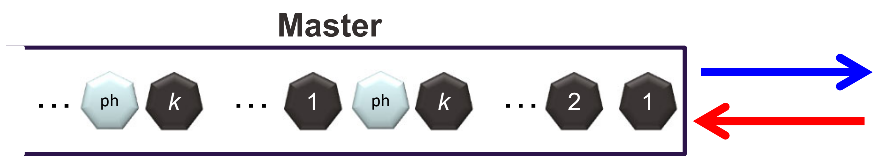
Why not just use the outputs as soon as you get them?

Recall there are $p+1$ processors: 1 master and p workers.

Input sequence: $X_j(x)$ is the jth requested observation from alternative x, with execution time $T_j(x)$.

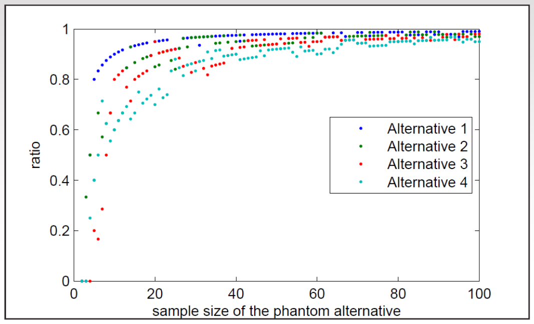
Output sequence: $Y_j(x)$ is the jth returned observation from alternative x.

Consider a round robin allocation of Simulation jobs.



New statistical issues

1. Random sample sizes $\rightarrow$



2. $Y_j(x),\ j=1,2,\ldots$ not i.i.d.

   Ex: $k=1,\ X_j=T_j\sim\text{Expon}(\mu)$ implies $\text{E}(Y_j)=\mu\left(1-\left(1-\frac{1}{p}\right)^j\right)$

3. Subtle dependence caused by elimination of systems.

New efficiency issues

R&S Procedure = jobs generated by the Master: $\mathcal{J} = \{J_j : 1 \le j \le M\}$.

- Let $0 < T_j < \infty$ be the wall-clock time job J_j finishes, so

$$T_e(\mathcal{J}) = \max_{j=1,2,\ldots,M} T_j$$

is the ending time of the procedure.

- $c(p,s)$ = cost to purchase p processors for s time units.
- $t(p,b)$ = maximum time we can purchase on p processors for budget \$$b$

$$t(p,b) = \max\{s : c(p,s) \le b\}.$$

Revised "efficient" objectives

Fixed precision requires statistical guarantees while being efficient:

$$\begin{aligned} \text{minimize}_{p,\mathcal{J}} \quad & \mathrm{E}[\beta_t \underbrace{T_e(\mathcal{J})}_{\text{time}} + \beta_c \underbrace{c(p, T_e(\mathcal{J}))}_{\text{cost}}] \\ \text{s.t.} \quad & \Pr\{\underbrace{G(\widehat{x}^\star, k)}_{\text{good event}}\} \ge 1-\alpha \end{aligned}$$

Fixed budget provides an efficiency guarantee within a budget:

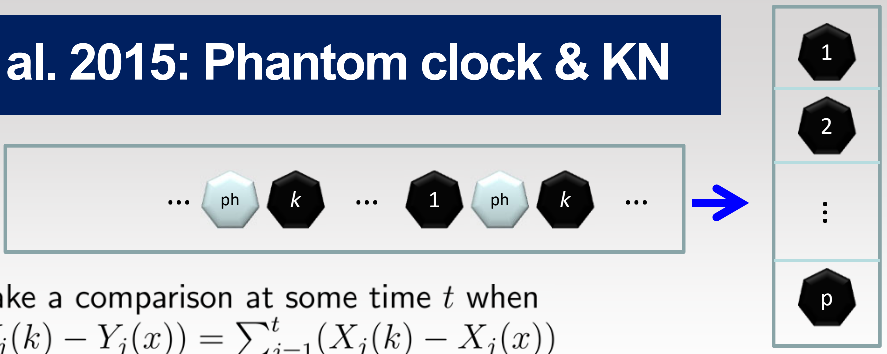
$$\begin{aligned} \text{minimize}_{p,\mathcal{J}} \quad & \mathrm{E}[\underbrace{\mathcal{L}(G^c(\widehat{x}^\star, k), \mathcal{J})}_{\text{loss of bad event}}] \\ \text{s.t.} \quad & \underbrace{t(p,b)}_{\text{processor time}} \le t^\star \end{aligned}$$

Existing patches

R&S Procedure	Load Balancing (Standard Assumptions)	Comparison Timing (Relaxed Assumptions)
Fixed-Precision	Simple Divide and Conquer (Chen 2005) Vector-Filling Procedure (Luo et al. 2015) Good Selection Procedure (Ni et al. 2017) Strategic Updating (Zhong et al. 2019)	Asymptotic Parallel Selection (Luo et al. 2015)
Fixed-Budget	Parallel OCBA (Luo et al. 2000) Asynchronous OCBA/KG (Kamiński & Szufel 2018)	

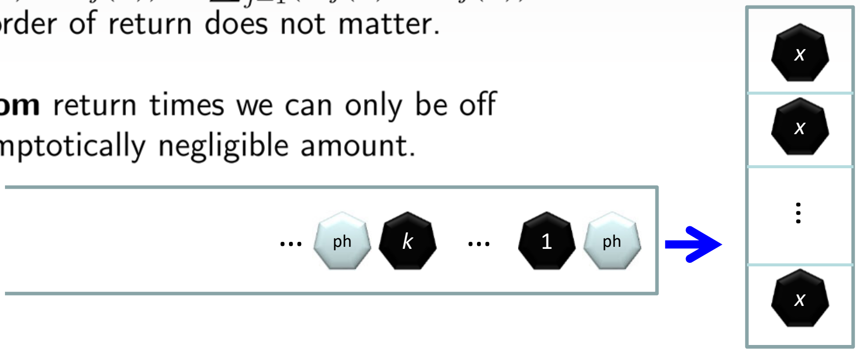
Northwestern | ENGINEERING 80 STOR-i 2020

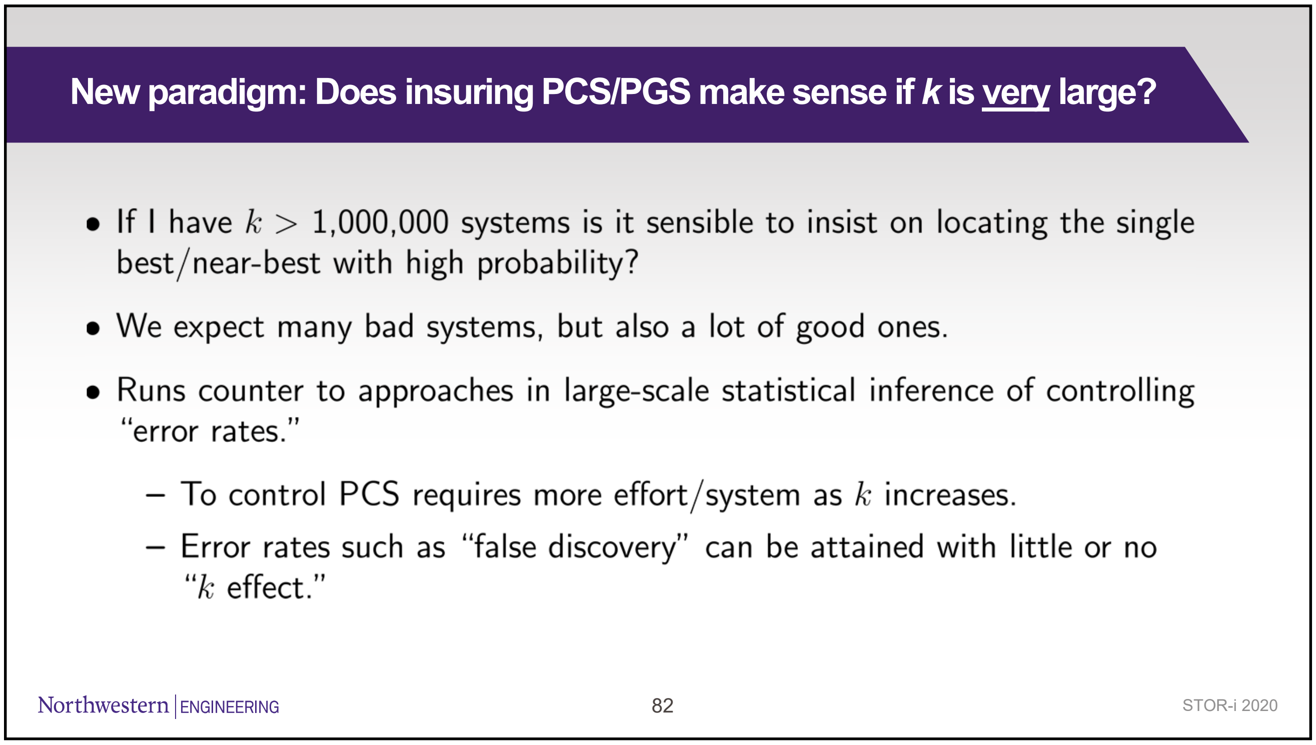
Luo et al. 2015: Phantom clock & KN



If we make a comparison at some time t when $\sum_{j=1}^{t}(Y_j(k) - Y_j(x)) = \sum_{j=1}^{t}(X_j(k) - X_j(x))$ then the order of return does not matter.

At **phantom** return times we can only be off by an asymptotically negligible amount.

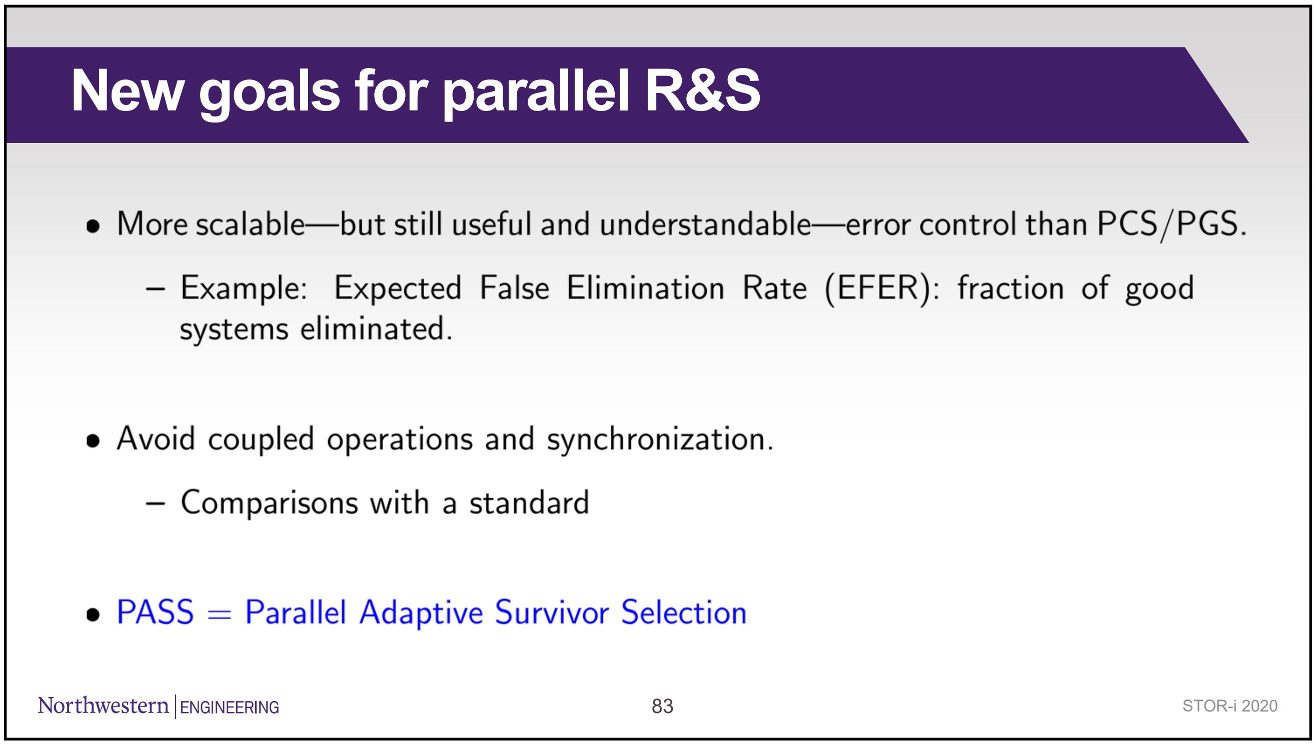




New paradigm: Does insuring PCS/PGS make sense if *k* is <u>very</u> large?

- If I have $k >$ 1,000,000 systems is it sensible to insist on locating the single best/near-best with high probability?
- We expect many bad systems, but also a lot of good ones.
- Runs counter to approaches in large-scale statistical inference of controlling "error rates."
  - To control PCS requires more effort/system as k increases.
  - Error rates such as "false discovery" can be attained with little or no "k effect."

Northwestern | ENGINEERING 82 STOR-i 2020



New goals for parallel R&S

- More scalable—but still useful and understandable—error control than PCS/PGS.
  - Example: Expected False Elimination Rate (EFER): fraction of good systems eliminated.
- Avoid coupled operations and synchronization.
  - Comparisons with a standard
- PASS = Parallel Adaptive Survivor Selection

Northwestern | ENGINEERING 83 STOR-i 2020

Building blocks

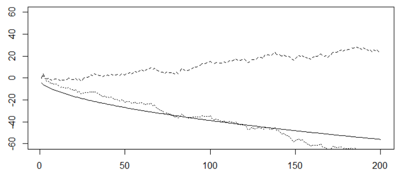
$Y_{x1}, Y_{x2}, \ldots$ i.i.d. with mean μ_x and $\mu_k \geq \mu_{k-1} \geq \cdots \geq \mu_1$

$S_x(n) = \sum_{j=1}^{n}(Y_{xj} - \mu^\star) = \sum_{j=1}^{n} Y_{xj} - n\mu^\star$

A non-decreasing function $g_x(\cdot)$ such that

$$\Pr\{S_x(n) \leq -g_x(n),\ \text{some } n < \infty\} \begin{cases} \leq \alpha & \mu_x \geq \mu^\star \\ = 1 & \mu_x < \mu^\star \end{cases}$$

$\mathcal{G} = \{x\colon\ \mu_x \geq \mu^\star\}$



Fan, Hong and N, "Indifference-zone-free Selection of the Best," *Operations Research* 64 (2016), 1499-1514.

Parallel Survivor Selection (PSS)

```
1. given a standard $\mu^\star$, an increment $\Delta n$ and a budget
2. let $\mathcal{W} = \{1, 2, \ldots, p\}$ be the set of available workers; $\mathcal{Q} = \{1, 2, \ldots, k\}$ the set of surviving
   systems; and $n_x = 0$ for all $x \in \mathcal{Q}$.
3. until the budget is consumed
   (a) while an available worker in $\mathcal{W}$, do in parallel:
       i. remove next system $x \in \mathcal{Q}$ and assign to available worker $w \in \mathcal{W}$
       ii. $j = 1$
       iii. while $j \leq \Delta n$
               simulate $Y_{x,n_x+j}$
               if $S_x(n_x + j) \leq -g_x(n_x + j)$ then eliminate system $x$ and break loop
               else $j = j + 1$
       iv. if $x$ not eliminated then return to $\mathcal{Q} = \mathcal{Q} \cup \{x\}$
       v. release worker $w$ to available workers $\mathcal{W}$
4. return $\mathcal{Q}$
```

Building block: Law of the iterated logarithm

The generic boundary function $g(\cdot)$ needs to unsure that driftless Brownian motion ($\mu_x = \mu^\star$) crosses with probability no more than the EFER α, while Brownian motion with negative drift ($\mu_x < \mu^\star$) crosses with probability 1.

Driftless Brownian motion grows to ∞ at rate $O(\sqrt{t \log\log(t)})$, while BM with negative drift goes to $-\infty$ at rate $O(t)$.

Thus $g(\cdot)$ needs to be between these two.

Example: $g(t) = \sqrt{[c + \log(t+1)](t+1)}$, tune c to get the desired EFER, and scale time by σ_x^2.

From PSS to PASS

- PSS requires no coupling & keeps the workers constantly busy.
  - Could be more efficient by making Δn depend on the system.
- The EFER is still controlled at $\leq \alpha$ and elimination still occurs with probability 1 if we replace $\mu^\star$ by $\mu(n) \leq \mu^\star$.
  - A system eliminated by a smaller standard would also have been eliminated by a larger standard.
  - A system protected from a larger standard would also be protected from a smaller one.
- This suggests we should try to **learn** a standard that achieves our objectives: Parallel **Adaptive** Survivor Selection.

Defining a "standard"

- Generically, we define the standard to be $\mu^\star = s(\mu_1, \mu_2, \ldots, \mu_k, \mu^+)$.
- Some examples of possibly interesting standards:
  - Protect the best: $\mu^\star = \mu_k$
  - Protect the top m: $\mu^\star = \mu_{k-m+1}$
  - Protect best & everything as good as μ^+: $\mu^\star = \min\{\mu^+, \mu_k\}$
- We want to **learn** the standard's value in a way that still avoids coupling and does not affect the EFER.

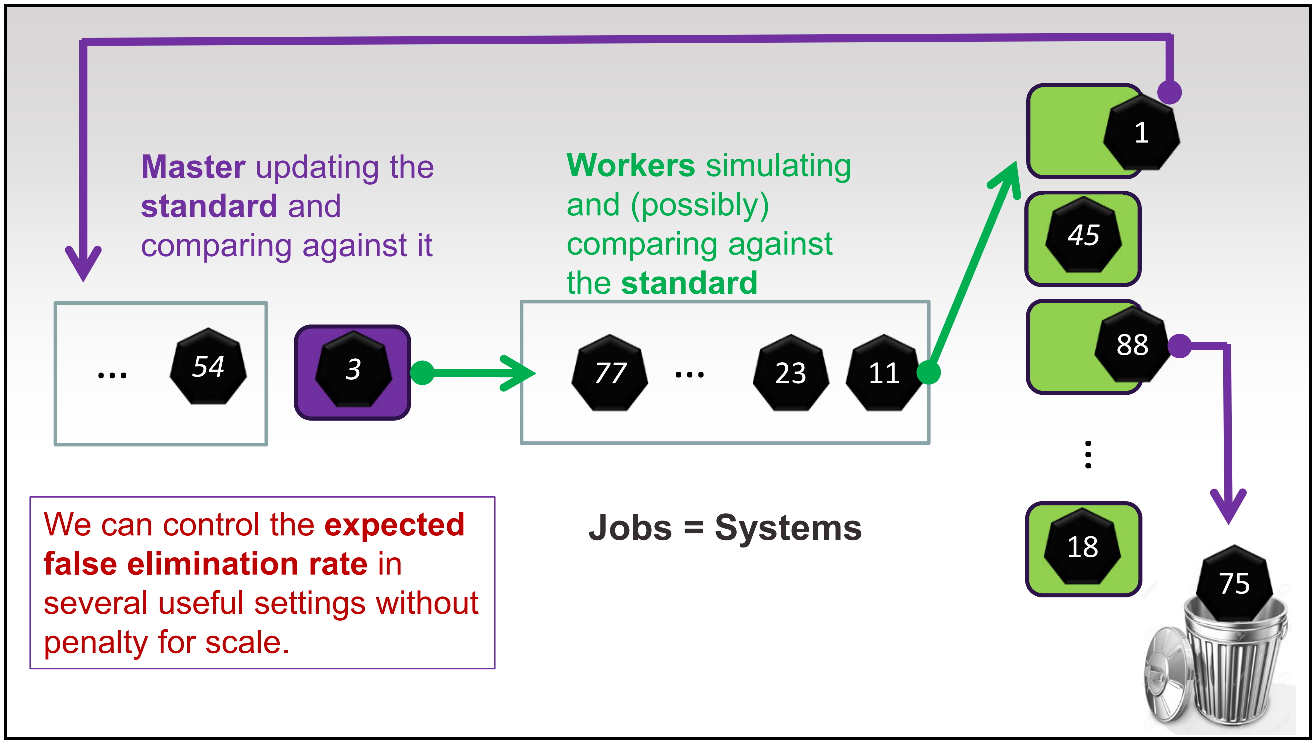
bi-PASS

- Consider the standard

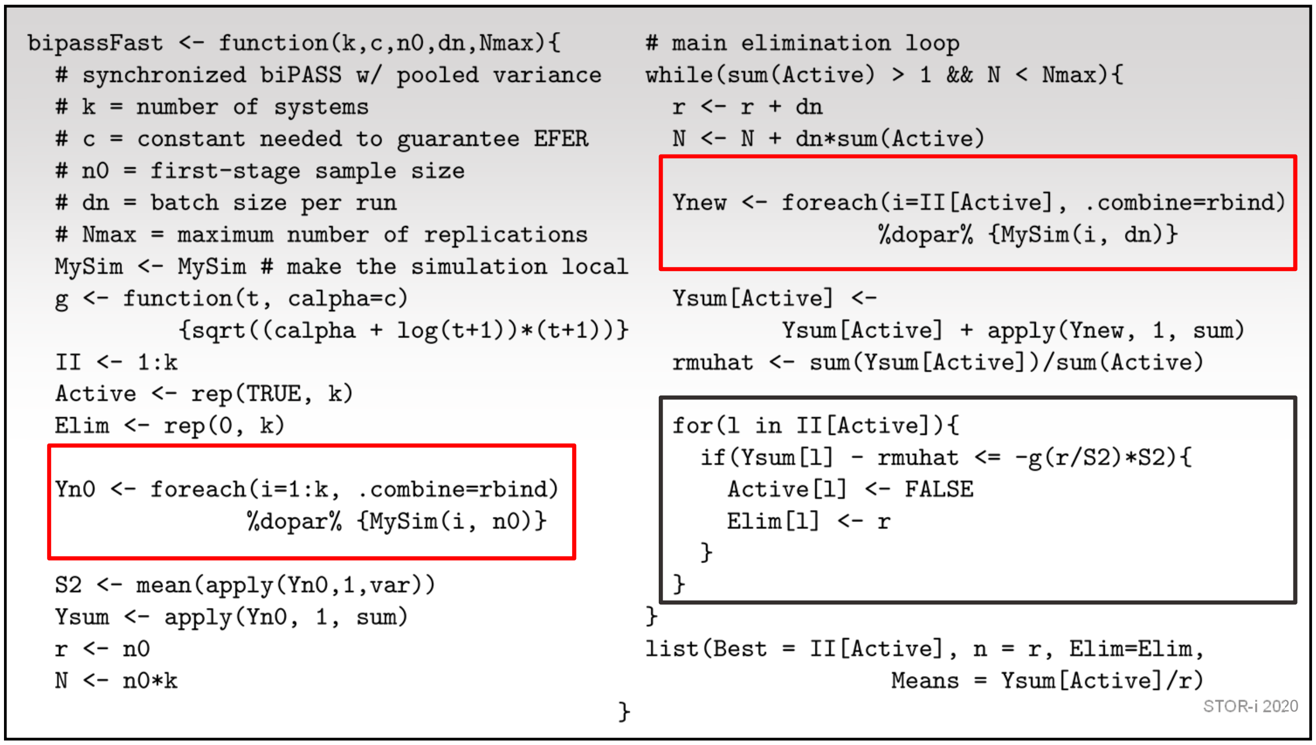
$$\bar{\mu} = \frac{1}{|\mathcal{Q}|} \sum_{x \in \mathcal{Q}} \bar{Y}_x(n_x)$$

- Essentially, the average of the current survivors.
  - Thus, the standard acts like a bisection search.
- Under some conditions we can show that the EFER is still $\leq \alpha$.

Pei, Hunter & N, "Parallel adaptive survivor selection," *WSC* 2018.


Master updating the **standard** and comparing against it
Workers simulating and (possibly) comparing against the **standard**
…
54
3
77
…
23
11
1
45
88
⋮
18
75
Jobs = Systems

We can control the **expected false elimination rate** in several useful settings without penalty for scale.



```
bipassFast <- function(k,c,n0,dn,Nmax){
  # synchronized biPASS w/ pooled variance
  # k = number of systems
  # c = constant needed to guarantee EFER
  # n0 = first-stage sample size
  # dn = batch size per run
  # Nmax = maximum number of replications
  MySim <- MySim # make the simulation local
  g <- function(t, calpha=c)
           {sqrt((calpha + log(t+1))*(t+1))}
  II <- 1:k
  Active <- rep(TRUE, k)
  Elim <- rep(0, k)

  Yn0 <- foreach(i=1:k, .combine=rbind)
              %dopar% {MySim(i, n0)}

  S2 <- mean(apply(Yn0,1,var))
  Ysum <- apply(Yn0, 1, sum)
  r <- n0
  N <- n0*k
  # main elimination loop
  while(sum(Active) > 1 && N < Nmax){
    r <- r + dn
    N <- N + dn*sum(Active)

    Ynew <- foreach(i=II[Active], .combine=rbind)
                %dopar% {MySim(i, dn)}

    Ysum[Active] <-
            Ysum[Active] + apply(Ynew, 1, sum)
    rmuhat <- sum(Ysum[Active])/sum(Active)

    for(l in II[Active]){
      if(Ysum[l] - rmuhat <= -g(r/S2)*S2){
        Active[l] <- FALSE
        Elim[l] <- r
      }
    }
  }
  list(Best = II[Active], n = r, Elim=Elim,
                  Means = Ysum[Active]/r)
}
```

STOR-i 2020

Illustration: Known μ_k vs. bi-PASS vs. subset

- (s, S) inventory problem
  - $p + 1 = 101$ processors
  - $k = 22{,}500$ solutions
  - $\alpha = 0.02$ EFER
  - Run as fixed budget; 7 systems left
- Baseline: take total consumed $\div\ k$ and apply subset selection
  - 181 in subset vs. 7 for bi-PASS

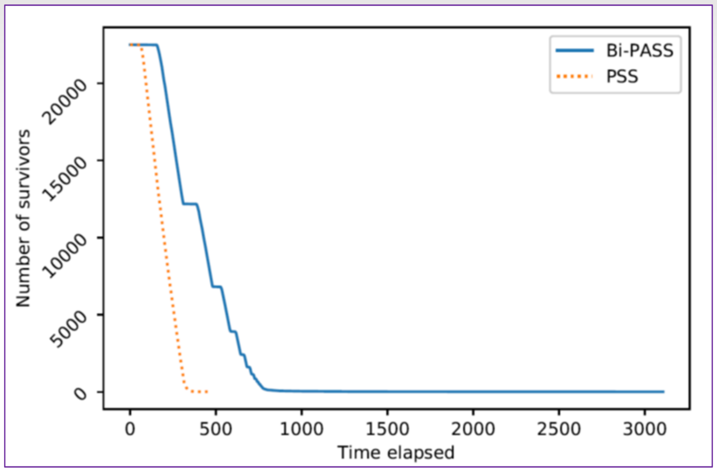


Illustration: GSP vs. bi-PASS

For the same problem setting, we ran 10 macroreplications of both procedures until the surviving systems had at least 1000 replications each.

- **GSP Results**

  95.3 survivors remaining

  1,265,439 total replications

  no false eliminations

  348.0 seconds on average

- **bi-PASS Results**

  37.7 survivors remaining

  575,326 total replications

  no false eliminations

  238.5 seconds on average

Experiment 1 with bi-PASS

Load the (s, S) inventory problem, which has $k = 1600$ solutions. With a total budget of 160,000 replications, compare `bipassSlow`, `bipassFast`, `SubsetCRN` and `SubsetParallel`.

For subset, allocate $n = 160{,}000/1600 = 100$ replications to each solution. For bi-PASS use $n_0 = 10$ $\Delta n = 20$ and $c = 5$, which gives an EFER of 0.05.

Remember to use your birthday for the seed.

```
ptimeSP <- system.time({resultSP <- SubsetParallel(1600, 0.05, 100, 12211956)})[3]

ptimeBF <- system.time({resultBF <- bipassFast(1600, 5, 10, 20, 160000)})[3]
```

Experiment 2 with bi-PASS

Load the $M/M/1$ queue problem, which has $k = 100$ solutions. With a total budget of 15,000 replications, compare the time to execute `bipassSlow` vs. `bipassFast`.

For bi-PASS use $n_0 = 10$, $\Delta n = 10$, and $c = 5$, which gives an EFER of 0.05.

Remember to use your birthday for the seed.

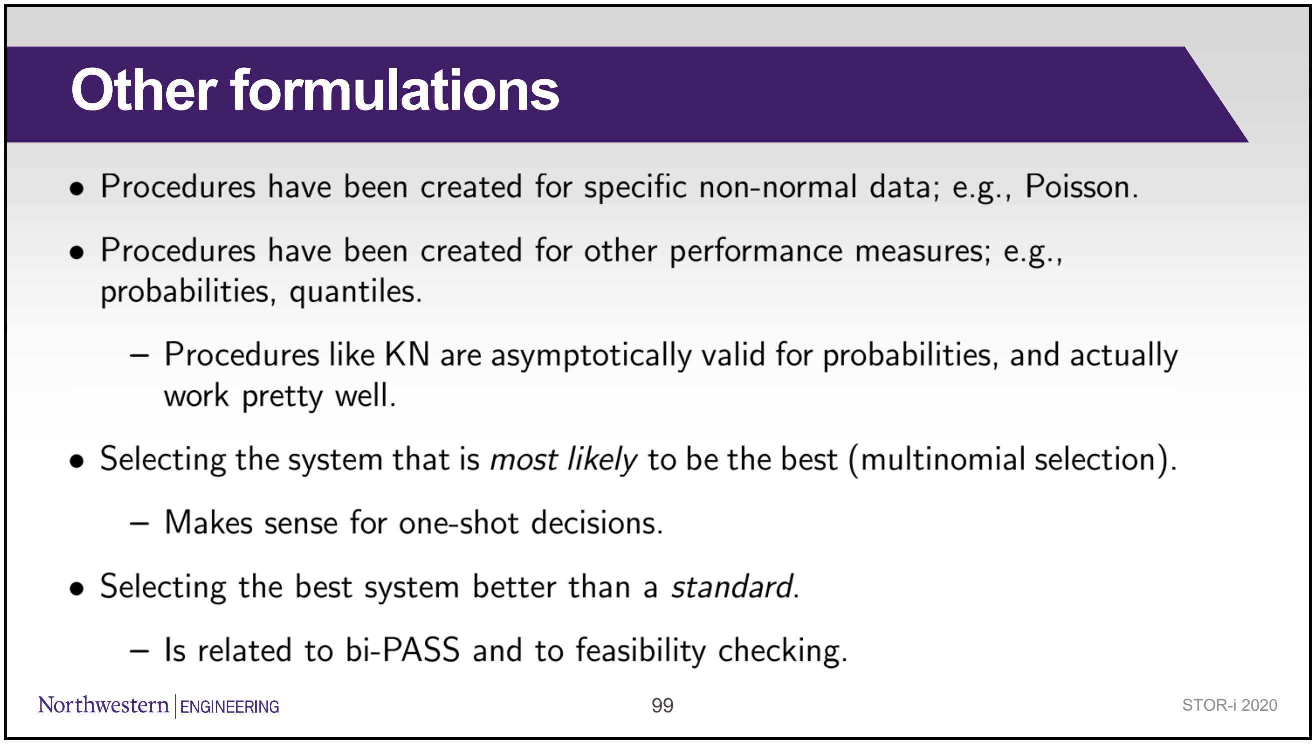
```
ptimeBS <- system.time({resultBS <- bipassSlow(100, 5, 10, 10, 15000)})

ptimeBF <- system.time({resultBF <- bipassFast(100, 5, 10, 10, 15000)})
```

CS issues really matter in parallel

- There is not one, unique parallel architecture, and customizations can matter.
- Message passing via MPI is conceptually easy, but unexpected behavior can occur, and passing messages does take time.
- Processors may be heterogeneous, and results can be lost.
- Memory may be shared or not.
- The overhead to load a simulation onto a processor can be substantial, so also need to consider fixed cost to set up as well as marginal time per replication.
- Management of pseudo-random numbers can be tricky, e.g., to use CRN.

Parallel R&S recap

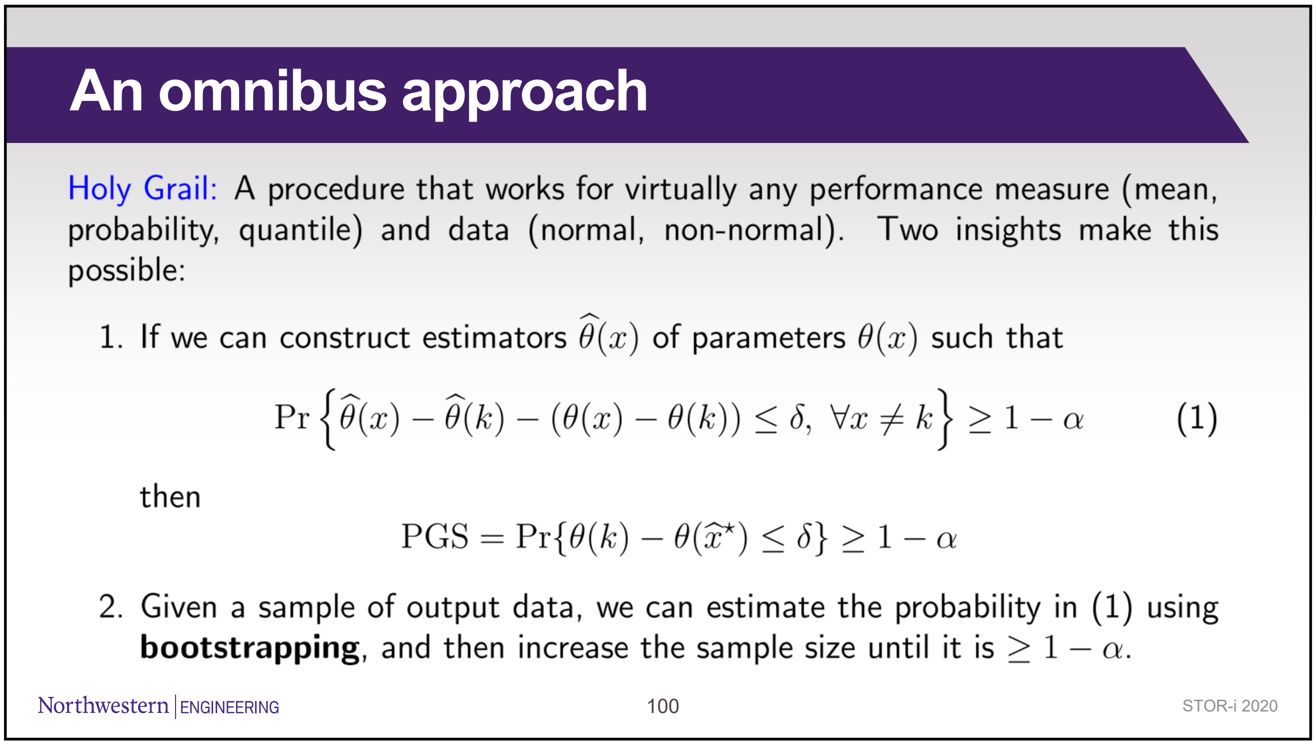
- If a simulation optimization problem can be treated as a R&S problem then it can be "solved."
  - All three errors can be controlled.
- High-performance, parallel computing extends the "R&S limit" but introduces new statistical and computational problems.
  - "Embarassingly parallel"
  - Violation of standard assumptions
  - cost $\neq$ number of observations
- The computer architecture issues can no longer be ignored.



Other formulations

- Procedures have been created for specific non-normal data; e.g., Poisson.
- Procedures have been created for other performance measures; e.g., probabilities, quantiles.
  - Procedures like KN are asymptotically valid for probabilities, and actually work pretty well.
- Selecting the system that is *most likely* to be the best (multinomial selection).
  - Makes sense for one-shot decisions.
- Selecting the best system better than a *standard*.
  - Is related to bi-PASS and to feasibility checking.

Northwestern | ENGINEERING 99 STOR-i 2020



An omnibus approach

Holy Grail: A procedure that works for virtually any performance measure (mean, probability, quantile) and data (normal, non-normal). Two insights make this possible:

1. If we can construct estimators $\widehat{\theta}(x)$ of parameters $\theta(x)$ such that
$$\Pr\left\{\widehat{\theta}(x) - \widehat{\theta}(k) - (\theta(x) - \theta(k)) \le \delta,\ \forall x \neq k\right\} \ge 1 - \alpha \qquad (1)$$
then
$$\text{PGS} = \Pr\{\theta(k) - \theta(\widehat{x}^\star) \le \delta\} \ge 1 - \alpha$$
2. Given a sample of output data, we can estimate the probability in (1) using **bootstrapping**, and then increase the sample size until it is $\ge 1 - \alpha$.

Northwestern | ENGINEERING 100 STOR-i 2020

Bootstrap PGS

Suppose we have N replications from each of the k systems, and let $\widehat{x}^\star = \mathrm{argmax}_x \widehat{\theta}(x)$, the sample best.

Then our bootstrap estimate of PGS is

$$\widehat{\mathrm{PGS}} = \frac{1}{B}\sum_{b=1}^{B}\prod_{x\neq\widehat{x}^\star}\mathcal{I}\left\{\widehat{\theta}^{(b)}(x)-\widehat{\theta}^{(b)}(\widehat{x}^\star)-\left[\widehat{\theta}(x)-\widehat{\theta}(\widehat{x}^\star)\right]\le\delta\right\}$$

where $\widehat{\theta}^{(b)}(x)$ come from bootstrap samples of size N. We increase N (generate more simulation output) until this estimate is $\geq 1-\alpha$.

Lee and N showed this approach to be asymptotically valid under very mild conditions on the data ($\delta \to 0$).

Best-mean illustration

Simulation output: $[Y_1(x),\ldots,Y_N(x)] \to \bar{Y}(x),\ x=1,2,\ldots,k$

$\widehat{x}^\star = \mathrm{argmax}_x \bar{Y}(x) \leftarrow$ current sample best with N replications

Bootstrap: We bootstrap the simulation outputs B times to get $\left[Y_1^{(b)}(x),\ldots,Y_N^{(b)}(x)\right] \to \bar{Y}^{(b)}(x),\ x=1,2,\ldots,k,\ b=1,2,\ldots,B$

$$\widehat{\mathrm{PGS}} = \frac{1}{B}\sum_{b=1}^{B}\prod_{x\neq\widehat{x}^\star}\mathcal{I}\left\{\bar{Y}^{(b)}(x)-\bar{Y}^{(b)}(\widehat{x}^\star)-\left[\bar{Y}(x)-\bar{Y}(\widehat{x}^\star)\right]\le\delta\right\}$$

Note: To incorporate CRN we bootstrap *vectors* of replications.

```
bootRS <- function(k,alpha,n0,delta,B,dn){
  # k = number of systems
  # n0 = first-stage sample size
  # 1-alpha = desired PCS
  # delta = indifference-zone parameter
  # B = number of bootstrap samples
  # dn = increment to increase n0
  PGS <- 0
  Yall <- NULL
  for (x in 1:k){
    Yall <- cbind(Yall, MySim(x, n0))}

  while(TRUE){
    bsum <- 0
    Ybar <- apply(Yall, 2, mean)
    xstar <- which.max(Ybar)
    for (i in 1:B){
      Ybarstar <- apply(apply(Yall, 2, sample, replace=TRUE), 2, mean)
      diffs <- Ybarstar[xstar] - (Ybar - Ybar[xstar])
      bsum <- bsum + prod(as.numeric(diffs <= delta))
    }
    PGS <- bsum/B
    print(c("N=", n0, "PGS =",PGS))
    if (PGS < 1 - alpha){
      Ytemp <- NULL
      for (x in 1:k){
        Ytemp <- cbind(Ytemp, MySim(x, dn))
      }
      Yall <- rbind(Yall, Ytemp)
      n0 <- n0 + dn
   }
    else{break}
  }
  list(Best = xstar, PGS=PGS, N = n0)
}
```

Testing bootstrap R&S

Reload the TTF example. Remember that the output data are highly non-normal.

Remembering to set the seed to your birthday, rerun `Rinott` with $k = 4$, $n_0 = 50$, $\alpha = 0.05$, and $\delta = 1000$. Note which system is selected, and the total number of observations generated.

Remembering to set the seed to your birthday, run `bootRS` with the same setting, plus $B = 200$ and $\Delta n = 100$.

```
resultBoot <- bootRS(4, 0.05, 50, 1000, 200, 100)
```

The total number of observations is 4 $\times$ ending sample size.

Multi-arm bandits

- There is a connection between R&S and multi-arm bandit (MAB) problems, but they are not the same.
  - Objectives of MAB and R&S often different (e.g., minimize regret).
  - MAB focus is online; R&S is always offline.
  - Different standards for "good performance."
  - Different assumptions about the data.

- R&S tends to be more willing to waste observations on inferior systems to reduce the **overall** number of observations.

Pointers

"Multi-armed bandit" is a slang name for a slot machine. Losing as little money as possible, you would like to find the machine with the highest payout.

- In Illinois the percentage payback ranged from 89–92.5% in 2017.
- Cooler name than "ranking & selection."

A good overview reference is

Jamieson & Nowak. 2014. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. *48th Annual Conference on Information Sciences and Systems*. IEEE.

More about the differences

"Online" means making decisions as we play, and it costs to play. "Offline" means doing our analysis, then implementing the choice, and rewards follow.

"Regret" depends on the rewards I accumulate as I play. PCS depends only on getting the best choice in the end, not how I get there.

MAB tends to evaluate algorithms via their probability *complexity*. R&S evaluates algorithms by their *finite-time effort*.

MAB tends to assume sub-Gaussian (even bounded) reward distributions; R&S often assumes normally distributed output.

MAB typically assumes finite budget; R&S often desires fixed precision.

Classical stochastic MAB

$x \in \{1, 2, \ldots, k\}$ arms to play, with reward distribution F_x having mean $\mu(x)$.

I_t the arm I choose to play on turn t, and $Y_t(I_t) \sim F_{I_t}$ is the reward I receive.

Regret	$R_n = \max_x \sum_{t=1}^{n} Y_t(x) - \sum_{t=1}^{n} Y_t(I_t)$
Expected regret	$r_n = \mathrm{E}(R_n)$
Pseudo-regret	$\bar{r}_n = \max_x \mathrm{E}\left[\sum_{t=1}^{n} Y_t(x) - \sum_{t=1}^{n} Y_t(I_t)\right]$

Loosely, the goal is to pick a policy for selecting I_t that minimizes (pseudo) regret.

Upper confidence bound policy

At the end of turn t, construct an UCB for each $\mu(x)$. On turn $t+1$ play the arm with the largest UCB. "Optimism in the face of uncertainty."

Clearly all forms of regret are non-decreasing in the number of turns n; MAB wants it to increase at the slowest possible rate. A building block:

$$\begin{aligned}\bar{r}_n &= n\mu(k) - \sum_{t=1}^{n} \mathrm{E}(\mu_{I_t}) \\ &= n\mu(k) - \sum_{x=1}^{k} \mu(x)\mathrm{E}(\#\text{ times played arm } x \text{ thru turn } n) \\ &= \sum_{x=1}^{k} (\mu(k) - \mu(x))\mathrm{E}(\#\text{ times played arm } x \text{ thru turn } n)\end{aligned}$$

MAB type of result

Try to upper bound the **rate** at which $\mathrm{E}(\#\text{ times played arm } x \text{ thru turn } n)$ increases as turns n increases for $x \neq k$.

This bounds the rate at which $\bar{r}_n$ increases.

Note that this bound on the rate of increase is neither an estimate of the pseudo-regret $\bar{r}_n$ nor a statistical guarantee.

It does say that as you play you accumulate regret no faster than the derived rate.

MAB policies are frequently quite simple to implement, which makes them attractive, and of course many problems require online solutions.

Project: Create a large-scale R&S procedure

Our (s, S) inventory problem has $k = 1600$ feasible solutions; pretty large.

Your job is to find the best by using the building blocks we already have to construct a new procedure: NSGS.

NSGS first applies subset selection to all k systems (n_0 reps, confidence level $1 - \alpha/2$), then passes the survivors to Rinott (n_0 reps, confidence level $1 - \alpha/2$, k systems, $\delta = 0.1$), using the data already obtained for subset.

Comment: It might seem that we could use the Rinott h for the reduced $k = |\widehat{\mathcal{S}}|$, the size of the surviving subset, but sadly this is not the case.

Some useful R

The Rinott phase will need to loop over just the survivors; here is one way to do that:

```
> result <- Subset(5, 10, 0.05)
> result$Subset
[1] 1 4
>
> for(i in result$Subset){
+ print(i)}
[1] 1
[1] 4
```

Key references

Hunter and N. 2017. Parallel Ranking & Selection, in *Advances in Modeling and Simulation: Seminal Research from 50 Years of Winter Simulation Conferences*, Springer.

Kim and N. 2016. Selecting the Best System, in *Handbooks in Operations Research and Management Science: Simulation*, Elsevier.

Ni, Ciocan, Henderson and Hunter. 2017. Efficient Ranking & Selection in Parallel Computing Environments, *Operations Research* **65**, 821-–836.

Luo, Hong, N and Wu. 2015. Sequential Procedures for Large-Scale Ranking-and-Selection Problems in Parallel Computing Environments, *Operations Research* **63**, 1177–1194.

Frazier. 2012. Tutorial: Optimization via Simulation with Bayesian Statistics and Dynamic Programming, *Proceedings of the 2012 Winter Simulation Conference*.

Pei, N and Hunter. 2018. Parallel Adaptive Survivor Selection, *Proceedings of the 2018 Winter Simulation Conference*.

Key references, continued

Chen & Ryzhov. 2019. Complete expected improvement converges to an optimal budget allocation. *Advances in Applied Probability*, **51**, 209–235.

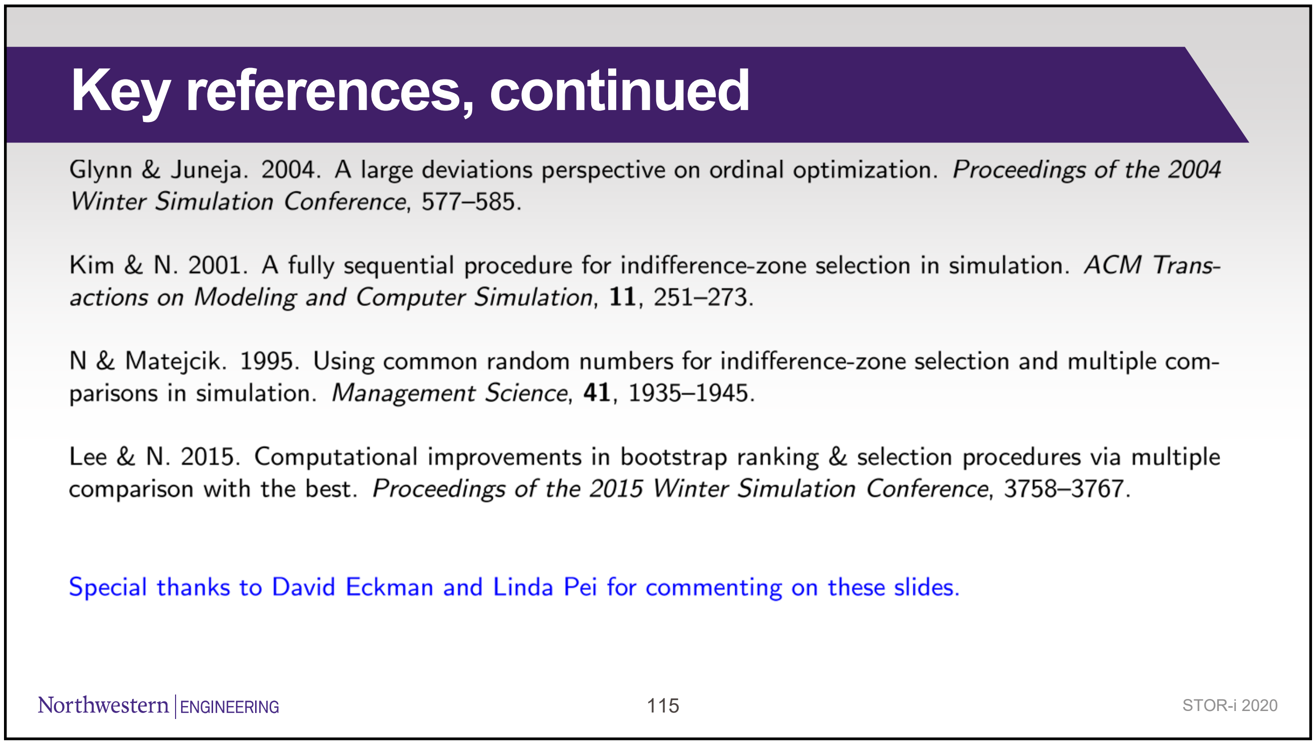
Fu. 2015. *Handbook of Simulation Optimization*. Springer.

Chen and Lee. 2011. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. World Scientific.

Shen, Hong and Zhang. 2017. Ranking and selection with covariates. *Proceedings of the 2017 Winter Simulation Conference*, 2137–2148.

Paulson. 1964. A sequential procedure for selecting the population with the largest mean from k normal populations. *The Annals of Mathematical Statistics*, **35**, 174–180.

Rinott. 1978. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics-Theory and methods*, **7**, 799–811.



Key references, continued

Glynn & Juneja. 2004. A large deviations perspective on ordinal optimization. *Proceedings of the 2004 Winter Simulation Conference*, 577–585.

Kim & N. 2001. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation*, **11**, 251–273.

N & Matejcik. 1995. Using common random numbers for indifference-zone selection and multiple comparisons in simulation. *Management Science*, **41**, 1935–1945.

Lee & N. 2015. Computational improvements in bootstrap ranking & selection procedures via multiple comparison with the best. *Proceedings of the 2015 Winter Simulation Conference*, 3758–3767.

Special thanks to David Eckman and Linda Pei for commenting on these slides.