Sequential Screening: A Bayesian Dynamic Programming Analysis

Peter I. Frazier (Cornell), Bruno Jedynak (JHU), Li Chen (JHU)

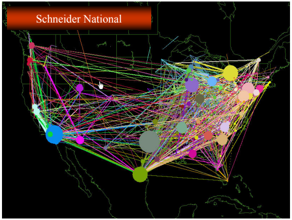
Operations Research & Information Engineering, Cornell University
Applied Mathematics, Johns Hopkins University

Tuesday December 11, 2012
Winter Simulation Conference
Berlin
Supported by AFOSR YIP FA9550-11-1-0083

The Screening Problem

- We have a simulator with a large number of input parameters, e.g., 100s or 1000s.
- We wish to understand how the simulator's output depends on its input, in some region of the input parameter space.
- Some of the input parameters have little or no effect on the simulator's output.
- **The Screening Problem**: Using as few simulations as possible, identify those parameters with an important effect on the output.
- **Why?** Once we finish sequential screening, we can do more fine-grained analyses focusing only on the important parameters. If there are only a few important parameters, this is faster than doing the analysis over all the parameters.

Screening is important when simulations take a long time to run, and there are many input parameters



Example: A stochastic simulator of operational decision-making at the logistics company Schneider National, developed in Simão et al. 2009.

- The simulator has more than 97 parameters.
- One full simulation takes 3 days.
- In the simulation calibration study (Frazier, Powell & Simão 2009), two parameters captured a large amount of the variation in the particular output of interest.
- This allowed us to solve a simulation optimization problem with 2 parameters instead of 97.

Being sequential is often more efficient; This talk analyzes a particular sequential screening method

- The naive approach to screening would be to iterate through the input parameters, testing them one at a time, to see which ones have an important effect on the output.
- For the Schneider National simulation, performing one simulation for each parameter would have required $97 \times 3 = 291$ days of simulation time.
- **Sequential is better**: By choosing the parameter values at which to perform simulations adaptively, in a sequential manner, we can find the important parameters much more quickly.
- In this talk, we use Bayesian statistics and dynamic programming to analyze one particular sequential screening algorithm, called **sequential bifurcation** [Bettonvil, 1990].

Overview of our contribution

- We use dynamic programming to find the Bayes-optimal group-splitting rule for sequential bifurcation, in an idealized setting.
- When factors have homogeneous prior probability of importance, the group-splitting rule proposed in [Bettonvil and Kleijnen, 1997] is optimal, in all problem instances tested.
- When factors have heterogeneous prior probability of importance, the Bayes-optimal group-splitting rule can be substantially better than previously proposed group-splitting rules.
- Our analysis makes three idealized assumptions:
  1. We assume simulations can be evaluated without noise, i.e., that our simulator is deterministic.
  2. We assume our simulator obeys a linear model with no interactions.
  3. We assume that each unimportant factor has a main effect that is exactly 0, rather than using a strictly positive cutoff between important/unimportant.

Our group splitting rule can still be used when the assumptions required for optimality are not met

- Our analysis makes the three idealized assumptions previously noted.
- We hypothesize that the insights from our analysis, and the group-splitting rule it provides, are still useful in problems that do not meet these assumptions:
  - When these assumptions are not met, one can use one of the many generalizations of SB designed for these situations.
  - These generalized SB procedures require a group splitting rule.
  - One can use the group-splitting rule that we derive in these generalized SB procedures. It will no longer be Bayes-optimal, but we hypothesize that it will be "pretty good", i.e., better than the status quo. Caveat: experiments to see whether this is true are future work.

A review of sequential bifurcation (SB)

- Here we present the version of the SB algorithm we will analyze, originally due to [Bettonvil, 1990, Bettonvil and Kleijnen, 1997].
- The version of SB that we present makes the idealized assumptions previously noted.
- A number of generalization of SB have been subsequently proposed, which relax these assumptions, e.g., [Cheng, 1997, Wan et al., 2010, Shi et al., 2012, Yaesoubi et al., 2010, Wan et al., 2006]

The simplest version of SB uses a factor model without interactions

- K is the number of parameters. We use the terms "factor" and "parameter" interchangably.
- We model the output of our simulator at $\vec{x} = \vec{c} + \vec{\Delta}$, where $\vec{c}$ is a fixed center point, and $\vec{\Delta}$ is "small", as

$$E[Y(\vec{c}+\vec{\Delta})] = E[Y(\vec{c})] + \sum_{k=1}^{K} \beta_k \Delta_k.$$

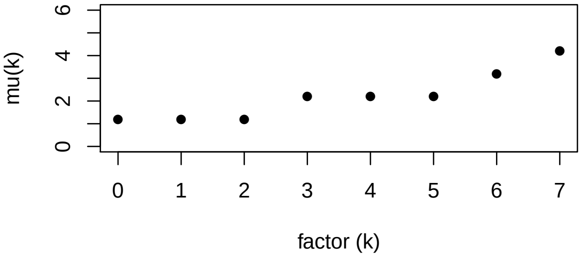
- A factor k is said to be **important** if $\beta_k \neq 0$. If $\beta_k = 0$ it is **unimportant**.
- We assume that we know the **sign** of each of the β_k values. Then, w.l.o.g, $\beta_k \geq 0$ (if not, replace x_k with $-x_k$).

$\mu(k)$ is monotone, with jumps at important factors

- For each factor k, choose two levels: low and high. In this talk, high is 1, and low is 0.
- Let $\vec{x}(k)$ correspond to setting the first k factors high, and the remaining low.

$$\vec{x}(k) = \vec{c} + [1, \ldots, 1, 0, \ldots, 0]$$

- Define $\mu(k) = E[Y(\vec{x}(k))] = E[Y(\vec{c})] + \sum_{i=1}^{k} \beta_i$
- Factor k is important iff $\mu(k) > \mu(k-1)$.

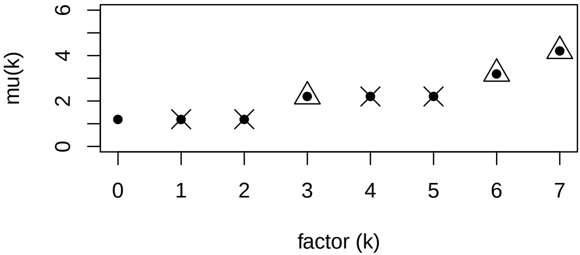


$\mu(k)$ is monotone, with jumps at important factors

- For each factor k, choose two levels: low and high. In this talk, high is 1, and low is 0.
- Let $\vec{x}(k)$ correspond to setting the first k factors high, and the remaining low.

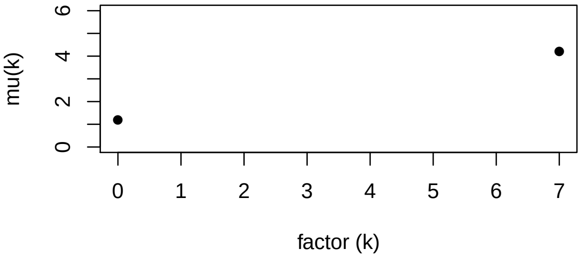
$$\vec{x}(k) = \vec{c} + [1, \ldots, 1, 0, \ldots, 0]$$

- Define $\mu(k) = E[Y(\vec{x}(k))] = E[Y(\vec{c})] + \sum_{i=1}^{k} \beta_i$
- Factor k is important iff $\mu(k) > \mu(k-1)$.



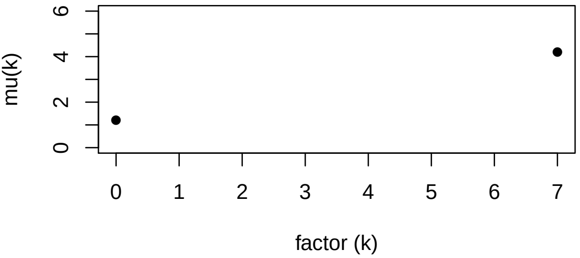
Example

- Recall $\mu(k) = E[Y(\vec{c})] + \sum_{i=1}^{k} \beta_i$.
- Step 1: Evaluate $\mu(0)$ and $\mu(K)$.
- If $\mu(0) = \mu(K)$, there are no important factors, and we are done.
- If $\mu(0) < \mu(K)$, there is at least one important factor in $1, \ldots, K$. Continue to Step 2.



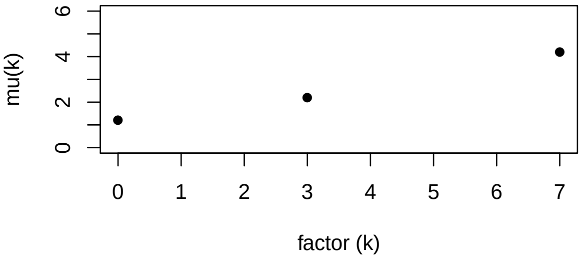
Example

- Step 2: The range $1, \ldots, K$ contains an important factor. Choose a factor k in this range and evaluate $\mu(k)$.
- We choose to evaluate at $k = 3$.



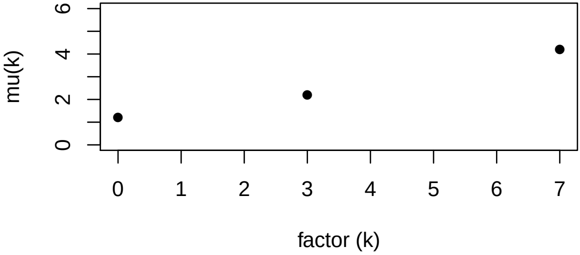
Example

- Step 2: The range $1, \ldots, K$ contains an important factor. Choose a factor k in this range and evaluate $\mu(k)$.
- We choose to evaluate at $k = 3$.



Example

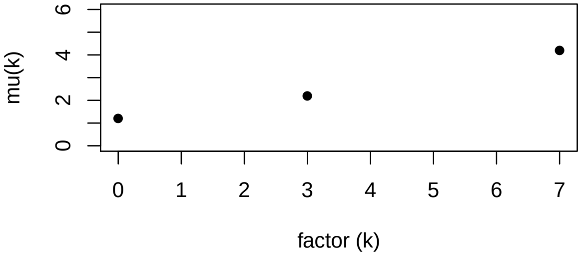
- Step 2: The range $1, \ldots, K$ contains an important factor. Choose a factor k in this range and evaluate $\mu(k)$.
- We choose to evaluate at $k = 3$.



- Since $\mu(0) < \mu(3)$, there is at least one important factor in $1, 2, 3$.
- Since $\mu(3) < \mu(7)$, there is at least one important factor in $4, 5, 6, 7$.

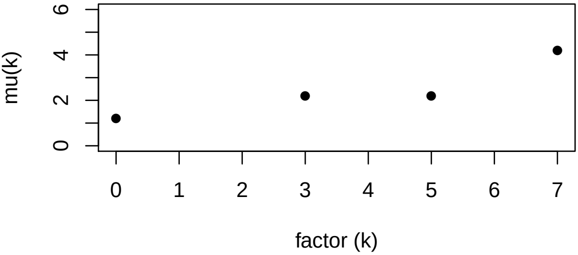
Example

- Step 3: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $4, 5, 6, 7$ and choose to evaluate at $k = 5$.



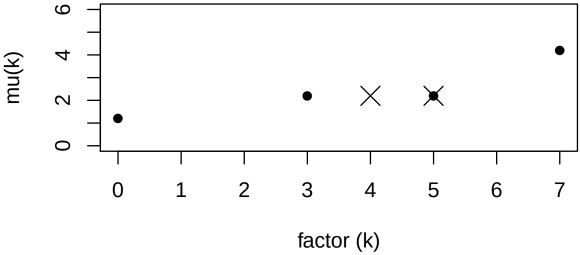
Example

- Step 3: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $4, 5, 6, 7$ and choose to evaluate at $k = 5$.



Example

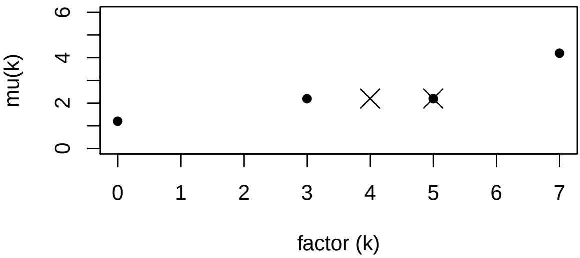
- Step 3: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $4, 5, 6, 7$ and choose to evaluate at $k = 5$.



- Since $\mu(3) = \mu(5)$, factors 4 and 5 are unimportant.

Example

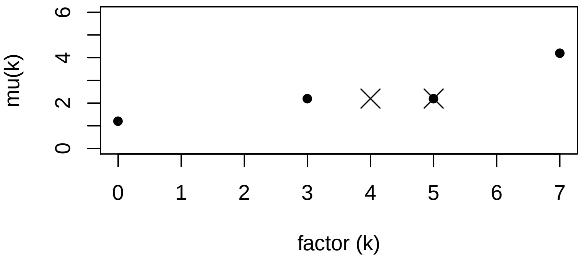
- Step 3: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $4, 5, 6, 7$ and choose to evaluate at $k = 5$.



- Since $\mu(3) = \mu(5)$, factors 4 and 5 are unimportant.
- Since $\mu(5) < \mu(7)$, there is at least one important factor in $6, 7$.

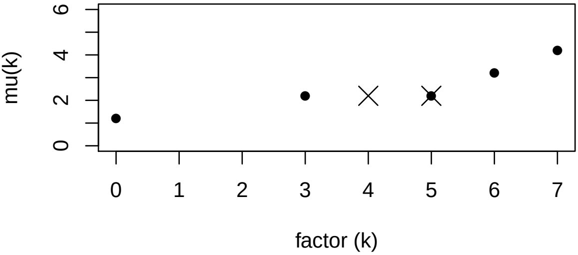
Example

- Step 4: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range 6,7 and choose to evaluate at $k = 6$.



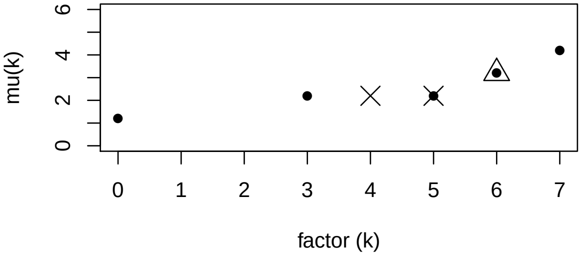
Example

- Step 4: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range 6,7 and choose to evaluate at $k = 6$.



Example

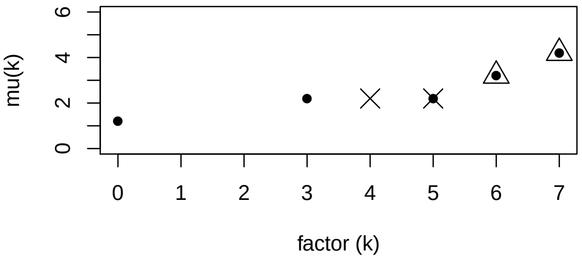
- Step 4: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $6,7$ and choose to evaluate at $k = 6$.



- Since $\mu(5) < \mu(6)$, factor 6 is important.

Example

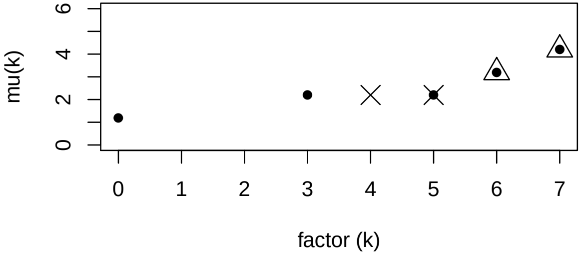
- Step 4: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range 6,7 and choose to evaluate at $k = 6$.



- Since $\mu(5) < \mu(6)$, factor 6 is important.
- Since $\mu(6) < \mu(7)$, factor 7 is important.

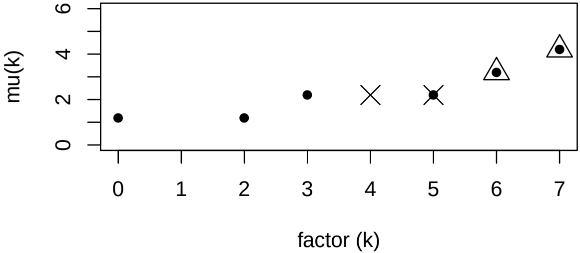
Example

- Step 5: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $1, 2, 3$ and choose to evaluate at $k = 2$.



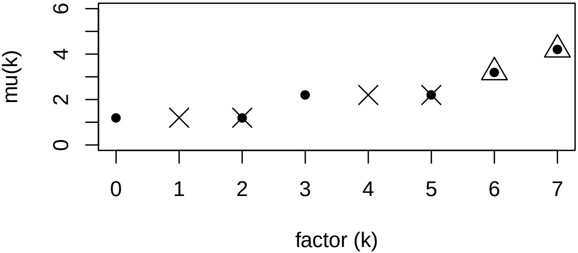
Example

- Step 5: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $1, 2, 3$ and choose to evaluate at $k = 2$.



Example

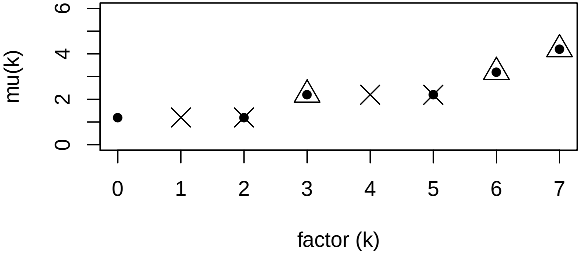
- Step 5: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $1, 2, 3$ and choose to evaluate at $k = 2$.



- Since $\mu(0) = \mu(2)$, factors 1 and 2 are unimportant.

Example

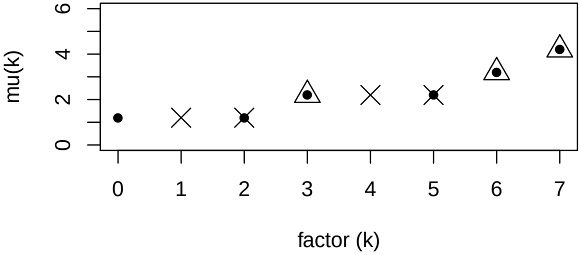
- Step 5: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $1,2,3$ and choose to evaluate at $k = 2$.



- Since $\mu(0) = \mu(2)$, factors 1 and 2 are unimportant.
- Since $\mu(2) < \mu(3)$, factor 3 is important.

Example

- Step 5: Choose a range known to contain an important factor, and select a factor in that range.
- We choose the range $1, 2, 3$ and choose to evaluate at $k = 2$.



- Since $\mu(0) = \mu(2)$, factors 1 and 2 are unimportant.
- Since $\mu(2) < \mu(3)$, factor 3 is important.
- We are done: 3,6,7 are important. 1,2,4,5 are unimportant.

We find an optimal group splitting rule

The focus of this talk is on the group-splitting rule.

- How should we split groups? Should we split in half (used by [Cheng, 1997, Wan et al., 2006])? Should we split so that one sub-group has a size that is a power of 2 (used by [Bettonvil and Kleijnen, 1997])? Should we use another rule?
- In this talk, we use dynamic programming to find a group splitting rule that minimizes the expected number of split points, under a Bayesian prior probability distribution on which factors are important.

We assume a Bayesian prior probability distribution

We construct our Bayesian prior probability distribution as follows:

- A factor k is said to be important if $\beta_k > 0$.
- We require $\mathbf{1}_{\{\beta_1>0\}}, \ldots, \mathbf{1}_{\{\beta_K>0\}}$ to be independent under our prior.
- Our prior is then completely specified by $P(\beta_1 > 0), \ldots, P(\beta_K > 0)$.
- We put factors into groups, where factors in group j have a common probability of importance, $P(\beta_k > 0) = p_j$.
- **Prior elicitation is convenient:** In practice, we imagine asking a practitioner to group factors into two groups, "likely to be important" and "unlikely to be important", and to estimate the probability of importance for each group.

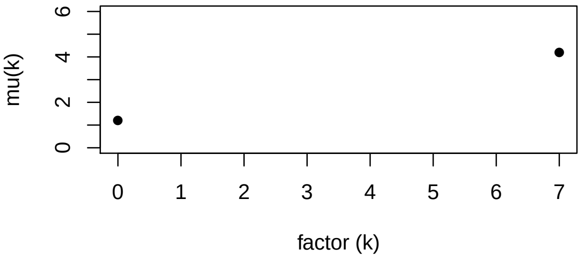
We use dynamic programming to find the optimal group splitting rule

- To illustrate, focus on the case of just one group, with probability of importance $p_1 = p$.
- The method we describe generalizes to an arbitrary number of groups, but the dimension of the DP that must be solved is equal to the number of groups.
- In practice, we think practitioners will be happy with 2 or 3 groups, for which computation is fast.

We use dynamic programming to find the optimal group splitting rule

- Let $V(n)$ be the expected number of additional evaluations of $\mu(\cdot)$, under the optimal group splitting rule, required to completely process a group of n factors known to contain at least one important factor, when the two endpoints have already been evaluated.
- The overall number of points to evaluate under the optimal policy is:

$$2 + P(\text{at least one important factor})V(K) = 2 + (1 - p^K)V(K)$$



Bellman's equation

- First, define $r(k,n)$ to be the probability that there is at least one important factor in A, given that there is at least one important factor in B, where $A \subseteq B$ are two arbitrary subsets of factors, with $|A| = k$ and $|B| = n$.

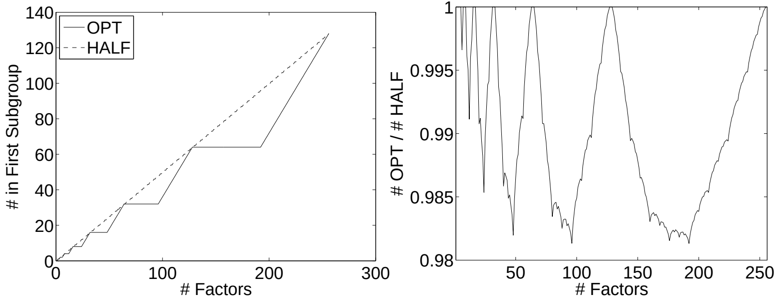
$$r(k,n) = \frac{1-(1-p)^k}{1-(1-p)^n}$$

- Bellman's equation is

$$V(n) = 1 + \min_{1 \le k \le n-1} r(k,n)V(k) + r(n-k,n)V(n-k).$$

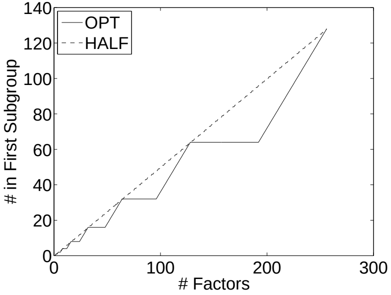
- The terminal condition is $V(1) = 0$.
- An optimal group splitting rule is to split so the first subgroup contains $k^*(n)$ elements, where $k^*(n)$ attains the minimum in Bellman's recursion.

Optimal policy provides only a small benefit with homogeneous factors



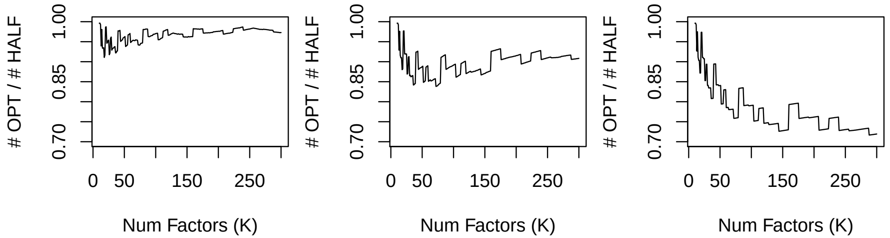
- The above uses a probability of importance of .10. The optimal policy is insensitive to this probability.
- OPT uses the optimal group-splitting rule.
- HALF splits groups in half, rounding to the nearest integer.
- The savings from OPT is small (at most 2% up to $K = 256$ factors).
- If K is a power of 2, splitting in half is optimal.

The optimal policy for homogeneous factors has a nice structure



- **Conjecture**: When factors have a homogeneous probability of importance, $k^*(n)$ is the largest integer power of 2 strictly less than n.
- This is the rule proposed by (Bettonvil & Kleijnen 1997).
- All numerical examples I have tried support it (up to 256 factors, with a variety of values for the probability of importance).

Optimal policy provides a larger benefit with heterogeneous factors



- Two groups of factors: "likely to be important" ($p_1 = 0.75$ and $K_1 = 10$) and "unlikely to be important" (p_2 and $K_2 = K - K_1$).
- Panels show different values of p_2: Left $p_2 = 0.05$, Middle $p_2 = 0.01$, Right $p_2 = 0.001$.
- When there is a big difference in probability of importance between the two groups, the optimal policy can be much more efficient.

Conclusion

- We used dynamic programming to find the optimal group-splitting rule,
- For homogeneous factors, one of the rules previously proposed in the literature appears to be optimal.
- For heterogeneous factors, the optimal policy is novel, and provides significant reductions for some problem settings.
    - We made some restrictive assumptions: deterministic simulator; no interactions; threshold at 0 for importance. Future work: investigate whether the group-splitting rules derived under these assumptions work well when these assumptions are not met, when used within SB methods that allow these assumptions to be relaxed.

Thank You!

References I

Bettonvil, B. (1990).
Detection of important factors by sequential bifurcation.
PhD thesis, Tilburg University.

Bettonvil, B. W. M. and Kleijnen, J. P. C. (1997).
Searching for important factors in simulation models with many factors: sequential bifurcation.
European Journal of Operational Research, 96(1):180–194.

Cheng, R. C. H. (1997).
Searching for important factors: sequential bifurcation under uncertainty.
In *Proceedings of the 29th conference on Winter simulation*, pages 275–280. IEEE Computer Society.

Shi, W., Kleijnen, J. P. C., and Liu, Z. (2012).
Factor Screening for Simulation with Multiple Responses : Sequential Bifurcation.
CentER Discussion Paper No. 2012-032, pages 1–35.

Wan, H., Ankenman, B. E., and Nelson, B. L. (2006).
Controlled Sequential Bifurcation: A New Factor-Screening Method for Discrete-Event Simulation.
Operations Research, 54(4):743–755.

Wan, H., Ankenman, B. E., and Nelson, B. L. (2010).
Improving the efficiency and efficacy of controlled sequential bifurcation for simulation factor screening.
INFORMS Journal on Computing, 22(3):482–492.

Yaesoubi, R., Roberts, S. D., and Klein, R. W. (2010).
A modification of Cheng's method: An alternative Factor Screening method for stochastic simulation models.
In *Winter Simulation Conference (WSC), Proceedings of the 2010*, pages 1034–1047. IEEE.

Summary of Sequential Bifurcation

- Evaluate $\mu(0)$ and $\mu(K)$.
- If $\mu(0) = \mu(K)$, we are done. Otherwise, the set $\{1, \ldots, K\}$ contains an important factor. Push it onto the stack.
- While the stack is non-empty, pop a set off the stack, call it $\{k_1, \ldots, k_2\}$.
  - Choose $k \in \{k_1, \ldots, k_2 - 1\}$. This is called **group-splitting**.
  - Evaluate $\mu(k)$.
  - If $\mu(k_1) < \mu(k)$, $\{k_1, \ldots, k\}$ contains an important factor. Push it on the stack.
  - If $\mu(k) < \mu(k_2)$, $\{k + 1, \ldots, k_2\}$ contains an important factor. Push it on the stack.

We can also solve the dynamic program for heterogeneous factors

- Suppose that factors are of two types.
- Factors of type 1 have probability of importance p_1.
- Factors of type 2 have probability of importance p_2, with $p_1 > p_2$.
- (Bettonvil & Kleijnen 1997) recommend sorting the factors in order of probability of importance. We do this.

We can also solve the dynamic program for heterogeneous factors

- Let $V(m,n)$ be the expected number of additional evaluations of $\mu(\cdot)$ required to completely process a group of factors with m factors of type 1 and $n-m$ factors of type 2 (under the optimal group splitting rule, when the two endpoints have already been evaluated).
- Terminal condition: $V(m,1)=0$ for $m=0$ and $m=1$.
- Bellman's recursion is on the next slide.

We can also solve the dynamic program for heterogeneous factors

$$V^*(m,n) = 1 + \min\{ \min_{m+1 \le u \le n-1} r(m,u)V^*(m,u) + r(0,n-u)V^*(0,n-u), \min_{1 \le u \le \min(m,n-1)} r(u,u)V^*(u,u) + r(m-u,n-u)V^*(m-u,n-u)\},$$

where

$$r(v,u) = \frac{1-(1-p_1)^v(1-p_2)^{u-v}}{1-(1-p_1)^m(1-q_2)^{n-m}} \tag{1}$$

is the conditional probability that a subgroup with v factors of type 1 and u factors overall contains an important factor. The dependence of $r(v,u)$ on m and n is supressed.