MS-E2170 Simulation
Spring 2022

Simulation based optimization

Outline

- Types of simulation based optimization
- Methods for continuous problems
  - Stochastic approximation
  - Sequential response surface method
- Methods for discrete problems
  - Finite feasible set
    - Ranking and selection
    - Computing budget allocation
  - Countably infinite feasible set
    - Metaheuristics

Optimization and simulation

- Experimentation with a set of designs produced by analyst or subject matter expert
- Experimental design and optimization
- Search algorithms that generate new designs for evaluation based on simulation response
  - Typically adopted from deterministic optimization

Types of simulation optimization

- Parametric problem
  - Static
  - Values of decision variables are assumed fixed during the operation of the system
- Control problem
  - Dynamic
  - Solution is a *policy* that controls the system according to its state
  - Approximate dynamic programming, simulation-based dynamic programming, neuro-dynamic programming, reinforcement learning

(Parametric) optimization

- General form of the problem

$$\min_{\theta \in \Theta} J(\theta) = E[L(\theta, \omega)]$$

$\theta = (\theta_{1,} \ldots, \theta_N) \in \Theta$	Vector of input parameters
ω	Sample path, i.e., a particular stream of random numbers
L	Response of the simulation model

Characteristics

- J can not be calculated directly
- Estimation through simulation
  - Time-consuming for complex models
  - Compare to function evaluation of deterministic optimization
  - In any case, some variability will remain

Stochastic approximation

- Counterpart of deterministic gradient-based techniques
- Local information on objective function is used to define most promising search direction
- Estimation of gradient on the basis of simulation responses

Stochastic approximation

- General form of the algorithm

$$\theta^{n+1} = \Pi_{\Theta}[\theta^n - a_n \hat{\nabla} J(\theta^n)]$$

θ^n	Current solution
Π_{Θ}	Projection to fesible set
$\hat{\nabla} J(\theta^n)$	Gradient estimate
a_n	Step size

Gradient estimation methods

- Finite difference
- Involves N+1 simulation model evaluations

$$\hat{\nabla} J(\theta) = \begin{bmatrix} (\hat{\nabla} J(\theta))_1 \\ \vdots \\ (\hat{\nabla} J(\theta))_N \end{bmatrix} = \begin{bmatrix} \left[\hat{J}(\theta_1 + c_1, \theta_2, \ldots, \theta_N) - \hat{J}(\theta_1, \theta_2, \ldots, \theta_N)\right] / c_1 \\ \vdots \\ \left[\hat{J}(\theta_1, \theta_2, \ldots, \theta_N + c_N) - \hat{J}(\theta_1, \theta_2, \ldots, \theta_N)\right] / c_N \end{bmatrix}$$

Gradient estimation methods

- Simultaneous perturbations
- 2 model evaluations

$$(\hat{\nabla} J(\theta))_i = \frac{\hat{J}(\theta+\Delta) - \hat{J}(\theta-\Delta)}{2\Delta_i},$$

where $\Delta = (\Delta_1, \ldots, \Delta_N)$ is a vector of IID random perturbations

Gradient estimation methods

- Infinitesimal perturbation analysis (IPA)
- Input parameters are perturbed, i.e., changed by an infinitesimal amount
- Resulting perturbation in L is propagated through model logic

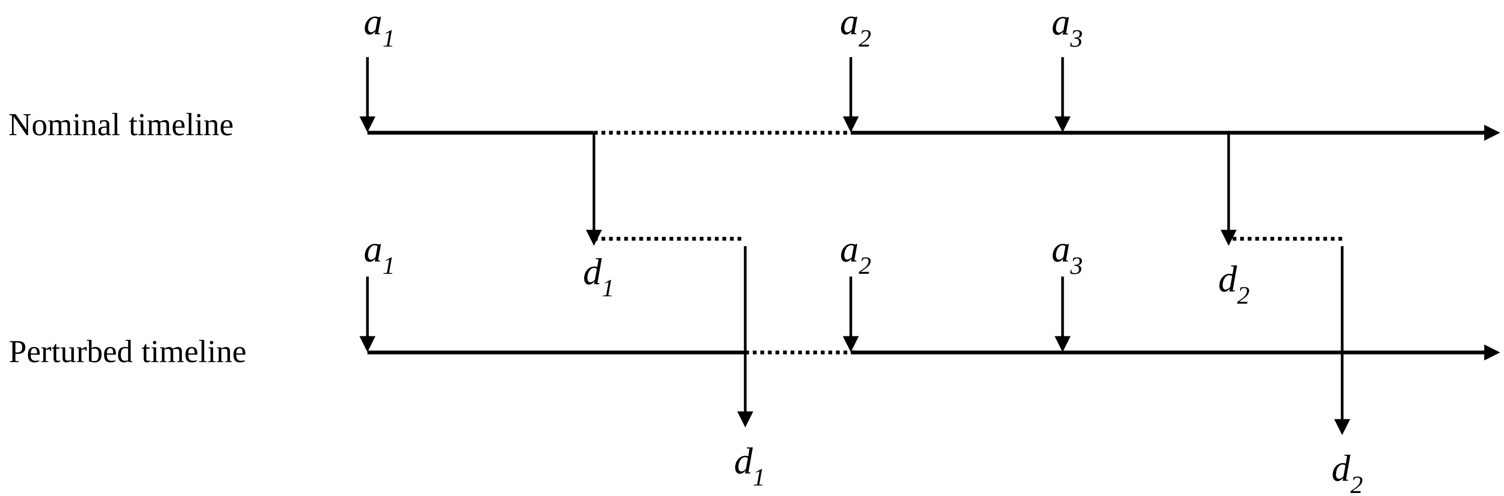
→ Only 1 simulation model evaluation is required to calculate a gradient estimate

Perturbation propagation

- Whenever a perturbed random variate is generated in the simulation, we record its consequence, e.g., "the value of its derivative"
- Simulation is performed using the original input parameter values
- Sequence of simulation events should not change

A queueing model – perturbed service time

- Arrival instants $(a_1,a_2,a_3,\ldots)$
- Departures $(d_1,d_2,d_3,\ldots)$



Perturbation for a single random variable

- An exponential random variable x with mean μ_n
- We observe x in the simulation
- Perturbation in x when the distribution mean is perturbed to μ_p

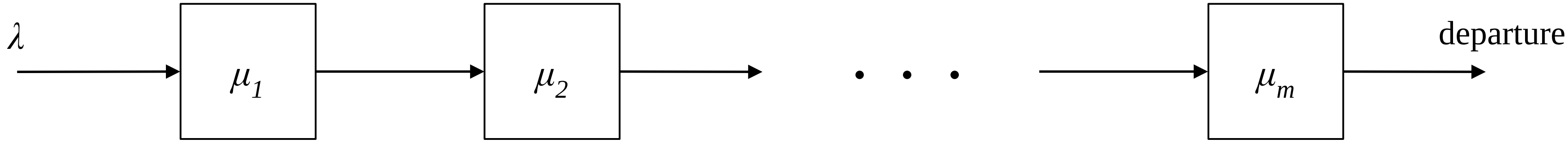
$$\Delta x = -\mu_p \ln(U) + \mu_n \ln(U)$$

$$\rightarrow \Delta x = \Delta \mu \ln(U), \text{ where } \Delta \mu = \mu_p - \mu_n$$

$$\text{since } \ln(U) = \frac{x}{\mu_n}, \quad \rightarrow \quad \Delta x = \Delta \mu \frac{x}{\mu_n} \quad \rightarrow \quad \frac{dx}{d\mu} = \frac{x}{\mu_n}$$

Propagation for a queueing network

- A series of G/G/1 queues
  - G=general, arbitrary distribution for interarrival & service times
  - 1 = one server



Propagation for a queueing network

- Departure times of items

$$d_i^j = s_i^j + \begin{bmatrix} max\left\{ d_{i-1}^j, d_i^{j-1} \right\} & \text{for } j>1 \\ max\left\{ d_{i-1}^j, a_i \right\} & \text{for } j=1 \end{bmatrix}$$

a_i arrival time of item i

d_i^j departure time of item i from server j

s_i^j processing time of item i at server j

Propagation for a queueing network

- Effect of processing time perturbation

$$\delta d_i^j = \delta s_i^j + \begin{cases} \delta d_{i-1}^j & \text{if } d_i^{j-1} < d_{i-1}^j \text{ (server j is busy for item i)} \\ \delta d_i^{j-1} & \text{otherwise (server j is idle for item i)} \end{cases}$$

Implementation for makespan of *n* customers

0.) Initialize $\nabla^j = 0$.

At the end of each customer service at server j:

If service time of the server is perturbed, go to step 1. Else, go to step 3.

1.) Calculate δs_i

2.) $\nabla^j = \nabla^j + \delta s_i$

3.) If machine $j+1$ is idle, set $\nabla^{j+1} = \nabla^j$

At the end of the simulation run:

4.) $\dfrac{\delta \text{Makespan}}{\delta \mu_s^j} = \nabla^m$

(Makespan = the departure time of the last customer from the last server)

Response surface methods

- Based on construction of a *metamodel*
- An approximation of the simulation model's input-output -behaviour
- Typical approaches are regression or neural networks
- Optimize the metamodel

Response surface methods

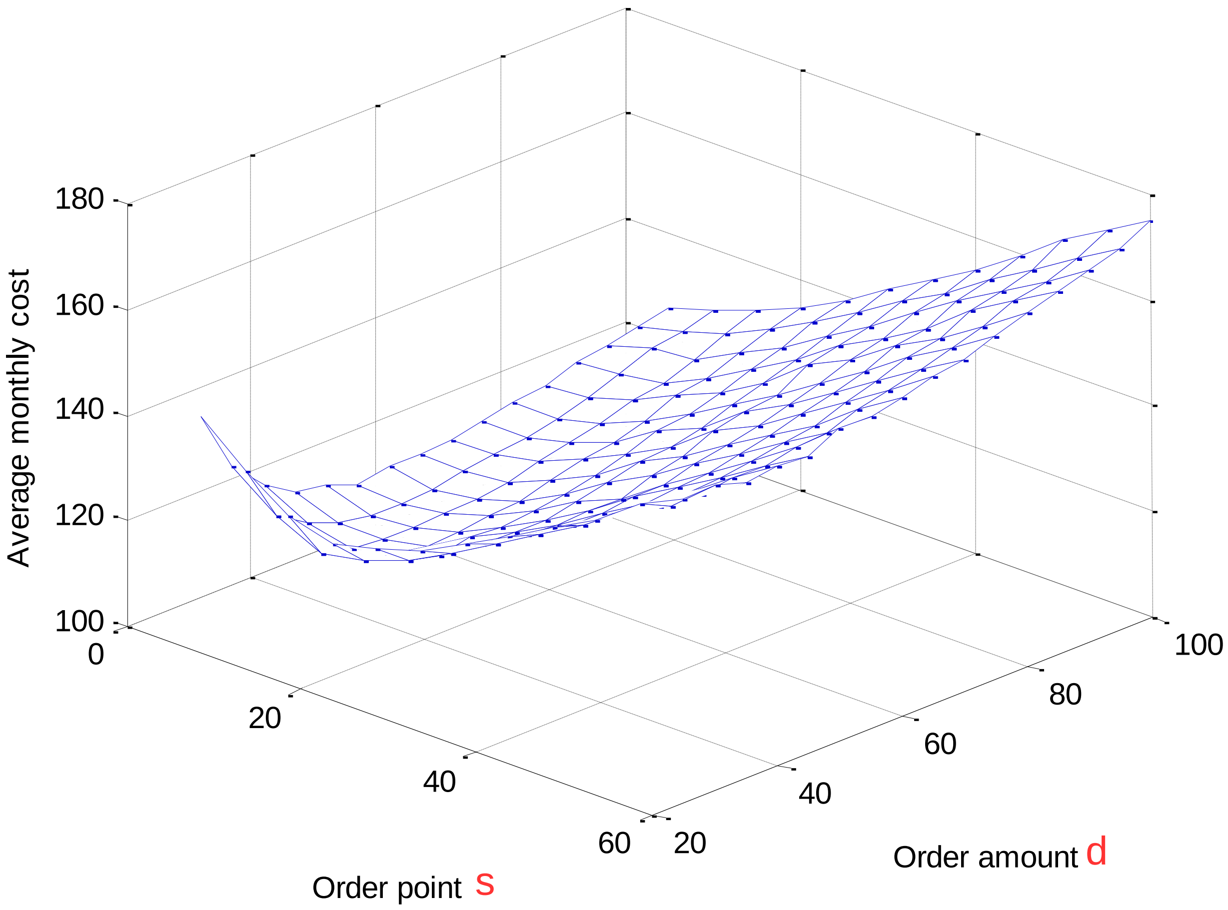
- A metamodel clearly results in crude approximations for long ranges of input parameters
- Metamodels are generally fitted to short ranges of parameters
- Gradient estimation

Example

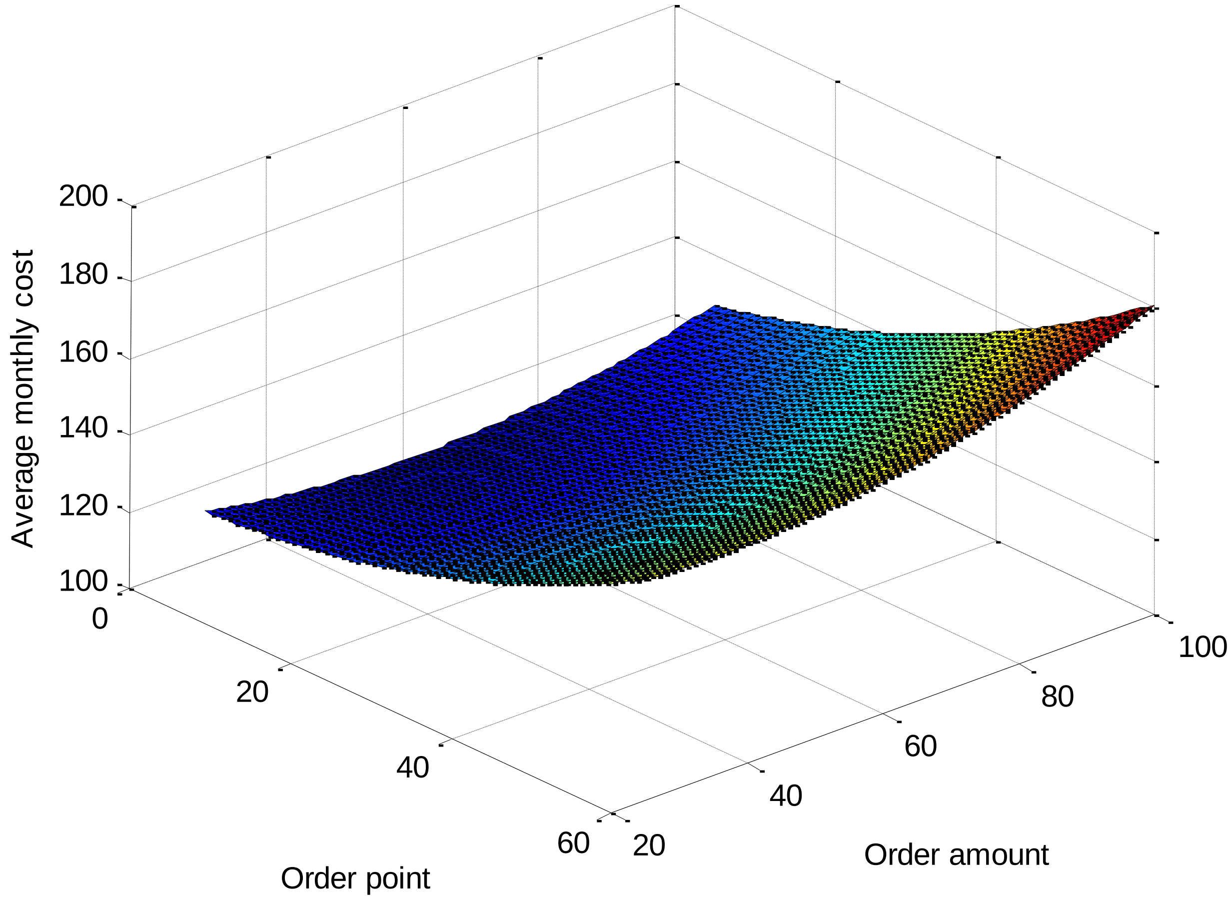
- Inventory system
  - Company uses stationary (s,d) -policy:

    If $inventory<s$, order amount d
  - Demand Disc(1/6,1;1/3,2;1/3,3;1/6,4) every Expo(10) months
  - Order cost C_F+C_VX, where X the ordered amount
  - Order delay Unif(0.5,1) months
  - Unsatisfied demand is backlogged
  - Holding and shortage costs, c_H and c_s per item per month

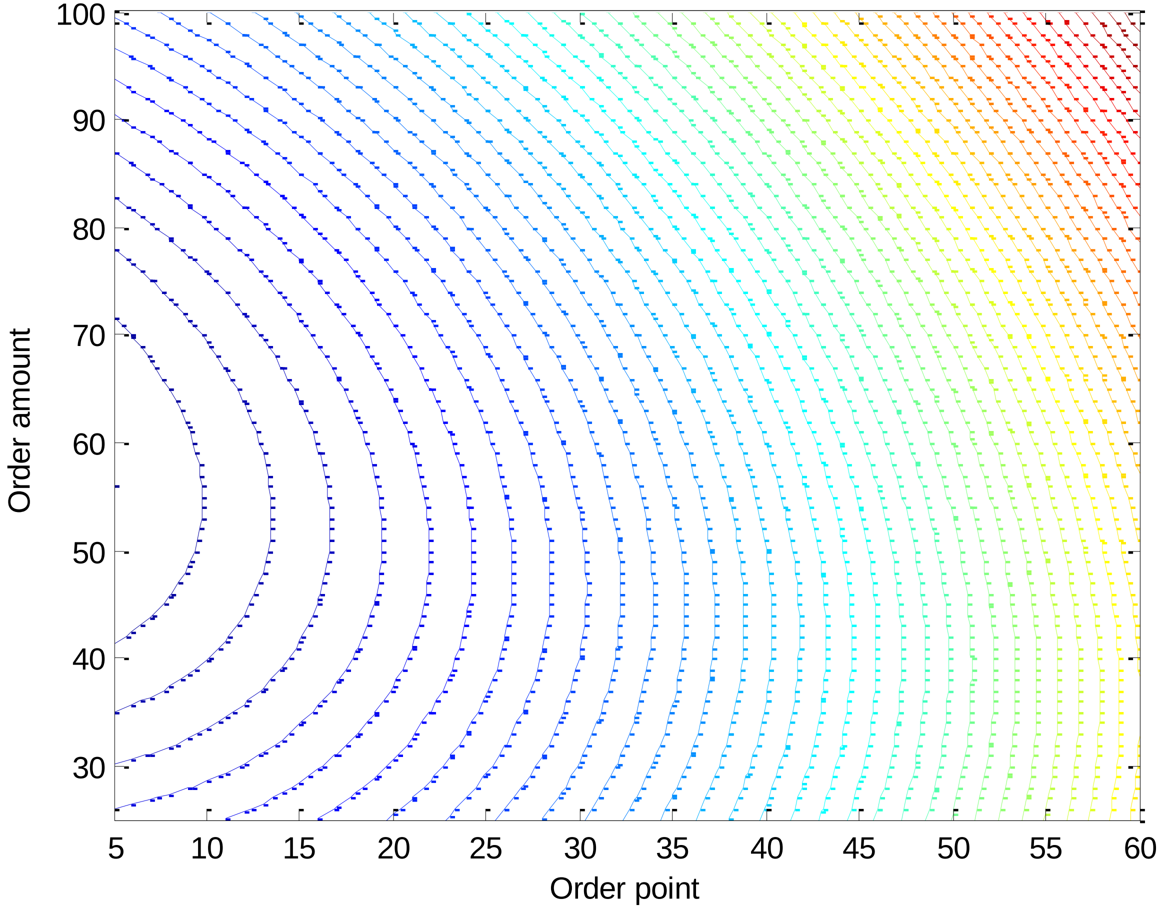
Response surface methods



Response surface methods

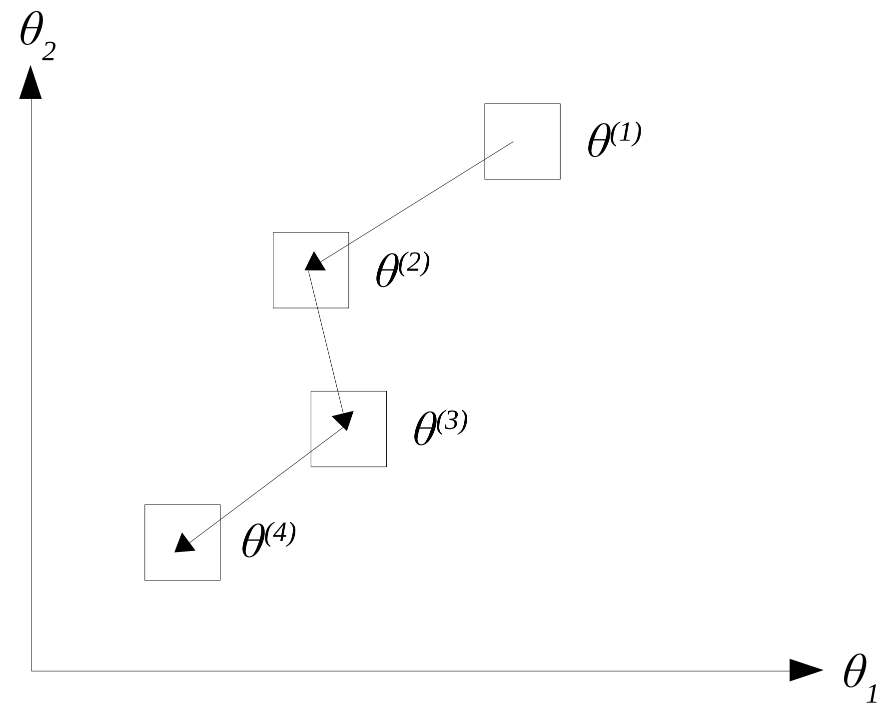


Response surface methods



Sequential response surface method

- Local metamodels are used for gradient estimation



Ranking and selection (R&S)

- Finding, among k systems the one with smallest $\mu_i = \mathrm{E}(X_{ij})$
  - X_{ij} is the response of interest from i^{th} system and j^{th} replication
  - k is relatively small, from 20 to a few thousand
  - X_{ij} are assumed independent across systems and replications

Indifference-zone

- We are typically indifferent between systems that are separated by less than some δ
- δ is called the indifference-zone parameter
- In other words, if $|\mu_i - \mu_j| < \delta$ we are willing to choose either i or j as the best system
- Indifference-zone procedures form the primary class of R&S -methods

Probability of correct selection

- If an R&S-method actually chooses the best system it is said to have made the correct selection (CS)

$$P(CS) = P(\hat{\mu}_{(j)} < \hat{\mu}_{(i)}, \forall\, i \neq j, \mu_{(i)} - \mu_{(j)} > \delta)$$

The R&S problem

- Guarantee given level of P(CS)
  - Allocating replications among systems

$$\min N_1 + N_2 + \ldots + N_k$$

$$\text{s.t.} \quad P(CS) \geq 1 - \alpha$$

$$N_i \in N, \; i = 1, \ldots, k$$

Levels of goals

- Select out of k systems
  - The best system
  - A subset of size m containing the best system
  - m best systems
- Different level goals can be handled by using a two-stage procedure that remains largely the same

The procedure – first stage

- Initially make n_0 replications for each system

- Calculate first-stage sample mean and variance estimates

$$\bar{X}_i^{(1)}(n_0) = \frac{1}{n_0} \sum_{j=1}^{n_0} X_{ij}, \quad S_i^2(n_0) = \frac{1}{n_0 - 1} \sum_{j=1}^{n_0} (X_{ij} - \bar{X}_i^{(1)}(n_0))$$

- The total sample size required for system i is

$$N_i = max\left\{ n_0 + 1, \left\lceil \frac{h^2 S_i^2(n_0)}{\delta^2} \right\rceil \right\}$$

The procedure – second stage

- Make $N_i - n_0$ additional replications and calculate second-stage means

$$\bar{X}_i^{(2)}(N_i - n_0) = \frac{1}{N_i - n_0} \sum_{j=n_0+1}^{N_i} X_{ij}$$

- Further define weigths

$$W_{i1} = \frac{n_0}{N_i}\left[1 + \sqrt{1 - \frac{N_i}{n_0}\left(1 - \frac{(N_i - n_0)\delta^2}{h^2 S_i^2(n_0)}\right)}\right], \quad W_{i2} = 1 - W_{i1}$$

- Make final selection according to the weighted sample means:

$$\tilde{X}_i(N_i) = W_{i1}\bar{X}_i^{(1)}(n_0) + W_{i2}\bar{X}_i^{(2)}(N_i - n_0).$$

The procedure

- h is a tabled constant depending on
  - Number of systems, first stage replications and confidence level
  - Different level goals
- Properties
  - Variances of X_{ij} need not be equal or known
  - Strictly X_{ij}'s should be normal, but the procedure is relatively robust against departures from normality

Optimal computing budget allocation (OCBA)

- Fixed number of replications divided among systems
- Attempts to maximize probability of correct selection (CS)

$$\max_{N_1, N_2, \dots, N_k} \quad P(CS)$$

$$\text{s.t.} \quad N_1 + N_2 + \cdots + N_k = T$$

$$N_i \in N, \, i = 1, \dots, k$$

Approximate P(CS)

- Let b the observed best design
- Using *Bonferroni* inequality

$$
\begin{aligned}
P(CS) &= P\left\{ \bigcap_{i=1,\dots,k;\, i\neq b} \left(\bar{J}_b - \bar{J}_i < 0\right) \right\} \\
&\geq 1 - \sum_{i=1,\dots,k;\, i\neq k} \left[1 - P\left\{ \bar{J}_b - \bar{J}_i < 0 \right\} \right] \\
&= 1 - \sum_{i=1,\dots,k;\, i\neq k} P\left\{ \bar{J}_b > \bar{J}_i \right\} := ACPS
\end{aligned}
$$

where $\bar{J}_i$ are the estimated performances.

An asymptotic allocation rule

- Theorem:

Given T simulation samples are allocated to k competing systems whose performance is depicted with r.v.'s with means $J(\theta_1),\ldots,J(\theta_k)$ and finite variances $\sigma_1^2,\ldots,\sigma_k^2$, as $T\rightarrow\infty$, ACPS is asymptotically maximized when

$$(1)\quad \frac{N_i}{N_j}=\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}},\quad i,j\in\{1,2,\ldots,k\},\ \text{and } i\neq j\neq b,$$

$$(2)\quad N_b=\sigma_b\sqrt{\sum_{i=1,\ldots,k;\, i\neq b}\frac{N_i^2}{\sigma_b^2}}$$

where $\delta_{b,i}=\bar{J}_b-\bar{J}_i$ and $\bar{J}_b\leq min_i\bar{J}_i$

The OCBA algorithm

(0) Perform n_0 replications for all systems: $l=0$; $N_1^l = N_2^l = \cdots = N_k^l = n_0$

(1) If $\sum_{i=1}^{k} N_i^l \geq T$, stop.

(2) Increase number of replications by Δ and compute the new budget allocation $N_1^{l+1}, N_2^{l+1}, \ldots, N_k^{l+1}$ using the above theorem.

(3) Perform additional $max(0, N_i^{l+1} - N_i^l)$ replications for $i = 1, \ldots, k$.

$l = l+1$. Go to step 1.

Metaheuristics

- Simulated annealing
  - Allows movements to non-improving solutions
- Tabu search
  - Maintains a list of forbidden moves
- Genetic algorithms
  - Crossover and mutation of promising solutions to create more fit descendants
  - Also applicable for discrete and qualitative variables
  - Ability to cope with large and complex search spaces

Simulated Annealing

- A metaheuristic to approximate global optimization in a large search space
- Name comes from metallurgy
  - Annealing refers cooling metal gradually after high heating
  - The molecules seek their optimal arrangements where the potential energy is minimized
- In simulated annealing, decision variables are randomly perturbed from current solution
  - If perturbed solution is better, it is accepted as current solution
  - If perturbed solution is worse, it is accepted at a probability which decreases by simulated temperature T
  - T is slowly decreased during the simulation

Simulated annealing in a simulation-optimization problem

(0) Choose temperature T and initial solution θ^0. Set $n=0$

Simulate the response $L(\theta^n, \omega)$

(1) Select randomly a neighbour $\tilde{\theta}^n$ of θ^n

(2) Simulate the response $L(\tilde{\theta}^n, \omega)$

(3) If $L(\tilde{\theta}^n, \omega) \leq L(\theta^n, \omega)$, then set $\theta^{n+1} = \tilde{\theta}^n$ and return to 1.

Else generate $U \sim U(0,1)$.

If $U \leq e^{-(L(\tilde{\theta}^n, \omega) - L(\theta^n, \omega))/T}$, then set $\theta^{n+1} = \tilde{\theta}^n$.

Else set $\theta^{n+1} = \theta^n$ and return to 1.

References

- Averill M. Law & W. David Kelton (2000)
  Simulation modeling and analysis
- Banks J. (ed.) (1998)
  Handbook of simulation: principles, methodology, advances, applications and practice.