

# A Brief Introduction to Optimization via Simulation

L. Jeff Hong

The Hong Kong University of Science and Technology

Barry L. Nelson

Northwestern University

# Outline

---

- Problem definition and classification
- Selection of the best
- Stochastic approximation and gradient estimation
- Random search algorithms
- Working with commercial solvers
- Conclusions

# Outline

---

- Problem definition and classification
- Selection of the best
- Stochastic approximation and gradient estimation
- Random search algorithms
- Working with commercial solvers
- Conclusions

# Problem Definition

---

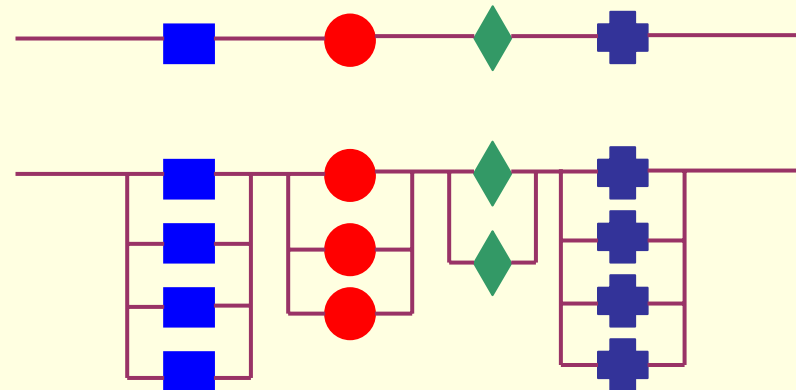
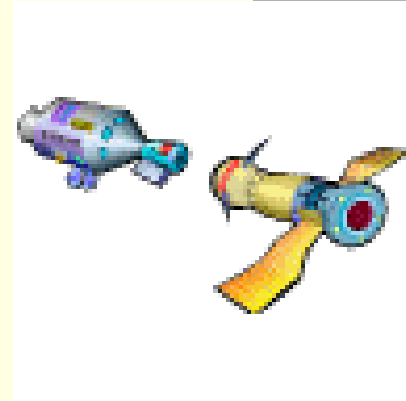
- Optimization via simulation (OvS) problems can be formulated as

$$\min \{g(\mathbf{x}) = E_{\mathbf{x}} [Y(\mathbf{x})]\}, \quad \mathbf{x} \in \Theta \subset \mathfrak{R}^d$$

- $\mathbf{x}$  is the vector of decision variables
- $g(\mathbf{x})$  is not directly observable, only  $Y(\mathbf{x})$  may be observed from running simulation experiments
- Little is known about the structure of the problem, e.g., convexity...
- We assume that  $\Theta$  is explicit

# Example: Highly reliable system

- A system works only if all subsystems work
- All subsystem components have their own time-to-failure and repair-time distributions
- **Decide how many and what redundant components to use**
- Goal is to **minimize steady-state system unavailability** given budget constraints
- **Few enough feasible alternatives that we can simulate them all**



# Example: Traffic signal sequencing

- Set the **length of the red, green and green-turn-arrow signals** along a network of road and intersections
- Goal is to **minimize mean aggregate driver delay**
- **Cycle lengths are naturally treated as continuous-valued decision variables**



# Example: Inventory management with dynamic customer substitution

- Single-period decision: **how many of each product variant** to stock?
- Goal is to **maximize expected profit**.
- Exogenous prices; consumer choice by MNL model, including no-purchase option
- Mahajan and Van Ryzin (2001)
- **Decision variables are naturally treated as integers (e.g., how many purple shirts)**



# Classification

---

- Based on the structure of the feasible region  $\Theta$ , we may divide OvS problems into three categories
  - **Section of the best**:  $\Theta$  has a small number of solutions (often less than 100). We may simulate all of them and select the best ← THIS IS OFTEN THE CASE
  - **Continuous OvS (COvS)**:  $\Theta$  is a (convex) subset of  $R^d$ , and  $x$  is a vector of continuous decision variables
  - **Discrete OvS (DOvS)**:  $\Theta$  is a subset of  $d$ -dimensional integers,  $x$  is a vector of integer-ordered decision variables
  - This classification is not exhaustive...



# Do these things fit?

---

- Find the strategy with the highest **probability** of delivering all orders on time
  - **Yes**, because a probability is the expected value of  $\{0, 1\}$  outputs
- Find the design that is **most likely to survive** the longest
  - **No**, because the performance of a design can only be judged relative to the competitors, not in isolation
- Maximize the **actual profit** that we will achieve next year
  - **No**, in fact this is impossible when there is uncertainty; we have to settle on a performance measure that can be averaged over the possible futures

# Outline

---

- Problem definition and classification
- **Selection of the best**
- Stochastic approximation and gradient estimation
- Random search algorithms
- Working with commercial solvers
- Conclusions

# Selection of the Best

Reminder:  $x$  is a selection of redundant components;  $\mu$  is long-run unavailability

## ■ Problem definition

- $\Theta = \{ x_1, x_2, \dots, x_k \}$
- Let  $\mu_i = g(x_i)$ ,  $Y_i = Y(x_i) \sim N(\mu_i, \sigma_i^2)$  with unknown  $\mu_i$  and  $\sigma_i^2$
- Suppose that  $\mu_1 \leq \mu_2 \leq \dots \leq \mu_{k-1} \leq \mu_k$
- The goal is to identify which solution is  $x_1$  by conducting simulation experiments
- The problem is to decide the sample sizes of all solutions so that the solution with the smallest sample mean is the best solution

# The difficulties

---

- Output randomness makes the decision difficult. We can only soften the goal to select the best solution with a high probability  $(1-\alpha)\times 100\%$ , say 95%
- The unknown difference between  $\mu_1$  and  $\mu_2$  can be arbitrarily small, making the decision very difficult, even just to achieve a given high probability
- Variances of the solutions may be unknown. They have to be estimated
- **Question:** When is the “normal” assumption reasonable?

# The Indifference-zone formulation

---

- Suppose that  $\mu_2 - \mu_1 \geq \delta$ , where  $\delta > 0$  is called an indifference-zone parameter. Basically, we only care about the difference between two solutions if it is more than  $\delta$ ; otherwise, they are indifferent.

- Example:  $\delta = 0.5\%$  in system availability

- The goal is to design procedures that assure

$$\Pr\{\text{select solution } \mathbf{x}_1 \mid \mu_1 \leq \mu_2 - \delta\} \geq 1 - \alpha$$

# Bechhofer's Procedure

- Assume all solutions have the same known variance  $\sigma^2$ .

**Step 1.** Determine the constant  $h$ , which satisfies  $\Pr\{Z_i \leq h, i = 1, 2, \dots, k-1\} = 1 - \alpha$  where  $(Z_1, Z_2, \dots, Z_{k-1})$  has a multivariate normal distribution with means 0, variances 1, and common pairwise correlations  $1/2$ . Let

$$n = \left\lceil \frac{2h^2\sigma^2}{\delta^2} \right\rceil.$$

**Step 2.** Take  $n$  observations from each solution and calculate  $\bar{Y}_i$  for all  $i = 1, 2, \dots, k$ , where  $\bar{Y}_i = \frac{1}{n} \sum_{j=1}^n Y_{ij}$  denotes the sample mean of  $Y(\mathbf{x}_i)$  calculated from the first  $n$  observations.

**Step 3.** Select the solution with the lowest sample mean  $\bar{Y}_i$ .

# Unknown and Unequal Variances

---

- Two-stage procedures are often used
  - *Stage I*
    - All solutions are allocated  $n_0$  observations to calculate their sample variances.
    - The sample variances are used to determine the sample size  $N_i$  for each  $x_i$
  - *Stage II*
    - $\max\{N_i - n_0, 0\}$  observations are taken for each solution
    - Calculate the sample means of all solutions based using all observations taken in Stage I and II
    - Select the solution with the smallest sample mean.

# When # of Solutions is Large

---

- Two-stage procedures are often conservative (i.e., allocating more observations than necessary)
  - Indifference-zone formulation
  - Bonferroni inequality
  - Especially when # of solutions is large
- NSGS Procedure (Nelson et al. 2001)
  - uses subset selection to screen out clearly inferior solutions after Stage I
  - much more efficient than two-stage procedures when # of solution is large



# Embedding Selection Procedure in Other Optimization Algorithms

---

- Selection-of-best procedures can also be embedded in other OvS algorithms (e.g., random search algorithms) to improve their efficiency and correctness
  - Clean-up at the end of optimization process (Boesel et al. 2003) ← **More later in the talk**
  - Neighborhood selection (Pichitlamken et al. 2006)
  - Guarantee an overall probability of correct selection at any time when solutions are generated sequentially (Hong and Nelson 2007)
  - Checking local optimality (Xu et al. 2010)

# Other Procedures

---

- In addition to two-stage procedures, there are also many sequential procedures
- Brownian motion approximation
  - Let
$$Z_{ij}(n) = \frac{n}{\sigma_{ij}^2} [\bar{Y}_i(n) - \bar{Y}_j(n)]$$
  - It can be approximated by a Brownian motion process with drift  $\mu_i - \mu_j$
  - Results on Brownian motion can be used to design sequential selection procedures, e.g., Paulson's procedure (Paulson 1964) and KN procedure (Kim and Nelson 2001)

# Other Procedures

---

- In addition frequentist “PCS” procedures, there are also many Bayesian procedures
  - The expected-value-of-information (EVI) procedures, e.g., Chick and Inoue (2001)
  - The optimal-computing-budget-allocation (OCBA) procedures, e.g., Chen et al. (2000)
  - Branke et al. (2007) compared frequentist’s and Bayesian procedures through comprehensive numerical studies. They conclude that
    - No procedure dominates all others
    - Bayesian procedures appear to be more efficient

# Outline

---

- Problem definition and classification
- Selection of the best
- **Stochastic approximation and gradient estimation**
- Random search algorithms
- Working with commercial solvers
- Conclusions

# Stochastic Root Finding

- Problem: Finding  $x$  such that  $E[H(x)] = 0$
- Robbins and Monro (1951) proposed the stochastic approximation algorithm

$$x_{n+1} = x_n - a_n H(x_n)$$

- They showed that  $x_n$  converges to a root if

$$a_n > 0, \quad \sum_{n=1}^{\infty} a_n = \infty, \quad \sum_{n=1}^{\infty} a_n^2 < \infty, \quad \text{e.g., } a_n = 1/n$$

# Continuous OvS

Reminder:  $x$  is a setting of traffic light timings;  $g(x)$  is mean aggregate delay

- Problem: minimize  $g(x) = E[Y(x)]$
- Assuming  $g(x)$  is continuously differentiable
- It is equivalent to find a root of

$$\nabla g(x) = 0$$

- If  $\nabla g(x) = E[H(x)]$ , then we may use Robbins-Monro SA algorithm to find a root

# More on Robbins-Monro SA

---

- If  $\nabla g(x) = E[H(x)]$  , then

$$x_{n+1} = x_n - a_n H(x_n)$$

- The algorithm may be viewed as a stochastic version of the steepest descent algorithm
- To apply Robbins-Monro SA, the key is to find an unbiased estimate of the gradient

# Infinitesimal Perturbation Analysis

---

- IPA (Ho and Cao 1983, Glasserman 1991) interchanges the order of differentiation and expectation

$$\nabla g(x) = \nabla E[Y(x)] = E[\nabla Y(x)]$$

- If  $Y$  is the system time of a queueing network and  $x$  is service rate, IPA can be applied
- If  $Y$  is discontinuous, e.g.,  $Y$  is an indicator function, then IPA cannot be applied



# The Likelihood Ratio Method

- The LR method differentiates its probability density (Reiman and Weiss 1989, Glynn 1990)
- Let  $f(y, x)$  denote the density of  $Y(x)$ . Then,

$$\begin{aligned}\nabla g(x) &= \nabla \mathbb{E}[Y(x)] = \nabla \int y f(y, x) dy = \int y \frac{\nabla_x f(y, x)}{f(y, x)} f(y, x) dy \\ &= \mathbb{E} \left[ Y \frac{\nabla_x f(Y, x)}{f(Y, x)} \right] = \mathbb{E} [Y \nabla_x \log f(Y, x)]\end{aligned}$$

Note that the decision variable  $x$  is a parameter of an input distribution; this is not always natural and may require some mathematical trickery

# Finite-Difference SA

---

- If  $Y(x)$  is a black box, finite-difference may be used to estimate the gradient (but with bias)
  - Run simulations at  $x$  and  $x + \Delta x$  then estimate derivative by  $[Y(x + \Delta x) - Y(x)] / \Delta x$
  - Need  $d+1$  simulations (forward difference) or  $2d$  simulations (central difference) if you have  $d$  decision variables

# Kiefer-Wolfowitz SA

- Kiefer-Wolfowitz SA algorithm (1952)

$$x_{n+1} = x_n - a_n \widehat{\nabla} g(\mathbf{x}_n)$$

where

$$\widehat{\nabla} g(\mathbf{x}_n) = \frac{1}{2c_n} \begin{pmatrix} Y(\mathbf{x}_n + c_n \mathbf{e}_1) - Y(\mathbf{x}_n - c_n \mathbf{e}_1) \\ Y(\mathbf{x}_n + c_n \mathbf{e}_2) - Y(\mathbf{x}_n - c_n \mathbf{e}_2) \\ \vdots \\ Y(\mathbf{x}_n + c_n \mathbf{e}_d) - Y(\mathbf{x}_n - c_n \mathbf{e}_d) \end{pmatrix}$$

- KW SA converges if  $c_n$  satisfies certain conditions

# Simultaneous Perturbation SA

- Kiefer-Wolfowitz needs  $2d$  simulation runs to estimate a gradient.
- Spall (1992) proposed the SPSA, which uses

$$\widehat{\nabla}g(\mathbf{x}_n) = \begin{pmatrix} B_1^{-1} \\ B_2^{-1} \\ \vdots \\ B_d^{-1} \end{pmatrix} \frac{Y(\mathbf{x}_n + c_n\mathbf{B}) - Y(\mathbf{x}_n - c_n\mathbf{B})}{2c_n}$$

where

$$\mathbf{B} = (B_1, \dots, B_d)' \text{ and } B_i = 1 \text{ or } -1 \text{ each w.p. } 1/2.$$

- SPSA only uses 2 simulation runs (but many replications of each in practice) to estimate a gradient no matter what  $d$  is.

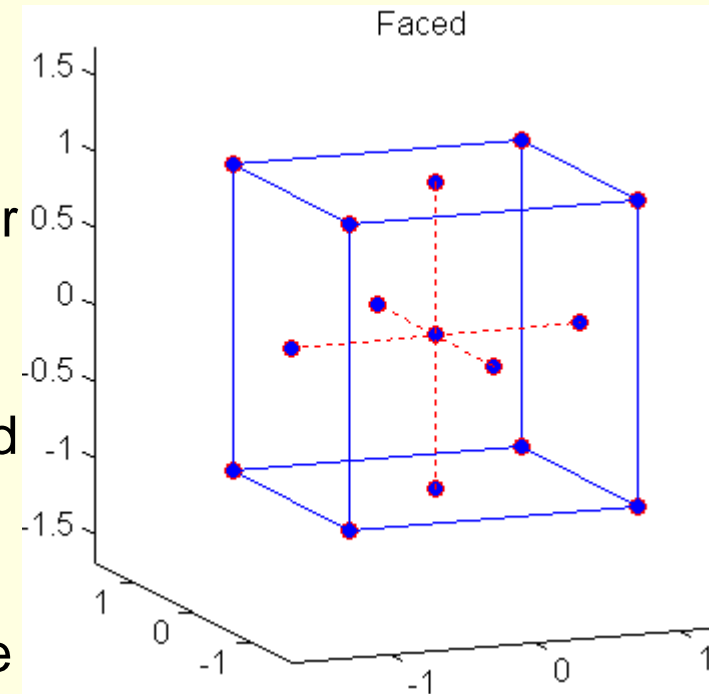
# Other COvS Algorithms

---

- There are also other convergent algorithms for COvS problems, including
  - Model reference adaptive search (MRAS, Hu et al. 2007) for global optimization
  - Grid search (e.g., Yakowitz et al. 2000) for global optimization
  - Stochastic trust region method (e.g., STRONG, Chang et al. 2007) for local optimization
- There are also many meta-model based algorithms (e.g., Barton and Meckesheimer 2006)

# Time out: Why not meta-models?

- Design of experiments and regression analysis are well known and supported by software; why not do that?
  - Ok, but rarely effective to fit a single global meta-model that is a low-order polynomial due to lack of fit  
→ need a sequential procedure
  - A lot of design points may be needed to support each meta-model when the dimension of  $x$  is large
  - Interpolation-based meta-models are just being developed for stochastic simulation



# Outline

---

- Problem definition and classification
- Selection of the best
- Stochastic approximation and gradient estimation
- **Random search algorithms**
- Working with commercial solvers
- Conclusions

# Discrete OvS

Reminder:  $x$  is the number of shirts of each type to order;  $g(x)$  is – expected profit

- DOvS problems:

$$\min g(\mathbf{x}) = E[Y(\mathbf{x})] \quad \text{s.t. } \mathbf{x} \in \Omega \cap Z^d$$

where  $\Omega$  is a convex, closed and bounded subset of  $R^d$  and  $Z^d$  is the set of  $d$ -dimensional integers

- Algorithms that relax integrality constraints, e.g., branch and bound, cannot be applied (e.g., it is not clear how to simulate an inventory with 12.3 shirts)
- Adaptive random search algorithms are often used



# Generic random search algorithm

---

1. **Randomly sample** some solutions from  $\Theta$  to get started; **simulate** them a little bit. Pick the sample best solution as your current optimal.
2. Randomly sample some additional solutions, perhaps favoring (but not exclusively) areas of  $\Theta$  where you have already seen some (apparently) good solutions.
3. Simulate the newly sampled solutions a bit more than solutions in previous iterations.
4. Pick the sample best of the new solutions as your current optimal.
5. If out of time, stop and report your current optimal; otherwise go to 2.

# Global Convergence

---

- There are many globally convergent random-search algorithms, e.g.,
  - Stochastic ruler method (Yan and Mukai 1992)
  - Simulated annealing (Alrefaei and Andradottir 1999)
  - Nested partitions (Shi and Olafsson 2000)
- As simulation effort goes to infinity...
  - All solutions are sampled
  - All solutions are simulated an infinite number of times
  - Different schemes are used to insure the two requirements

# Improving Finite-Time Performance

---

- Andradottir (1999) suggested using cumulative sample means to estimate the value of the solutions
  - Finite-time performance becomes much better
  - Almost-sure convergence becomes easier to prove (all solutions are simulated infinitely often)
  - Asymptotic normality may be established.

# Drawbacks of Global Convergence

---

- A good convergence result...
  - assures the correctness of the algorithm if it runs long enough
  - helps in determining when to stop the algorithm in a finite amount of time
- Global convergence...
  - achieves the former, but gives little information on the latter (because it requires all solutions to be sampled)
  - provides little information when the algorithm stops in a finite amount of time

# Local Convergence

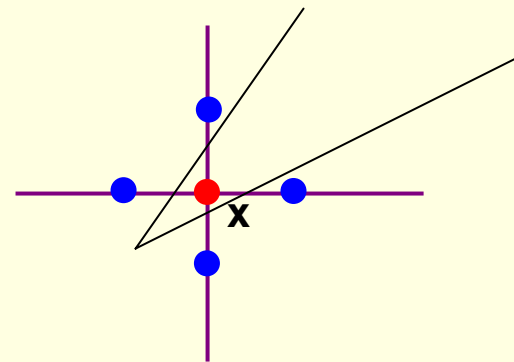
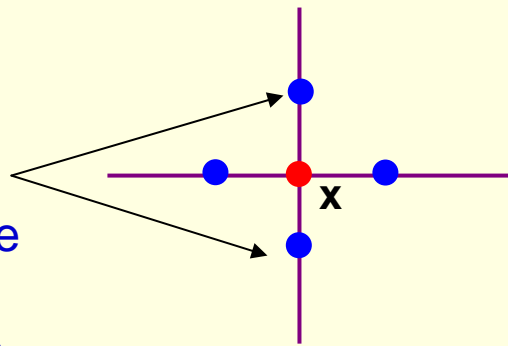
- Definition of the local neighborhood of  $x$ :

$$N(x) = \{ y : y \text{ in } \Theta \text{ and } \| y - x \| = 1 \}$$

- $x$  is a local optimal solution if

$$g(x) \leq g(y) \text{ for all } y \text{ in } N(x), \text{ or } N(x) = \Phi$$

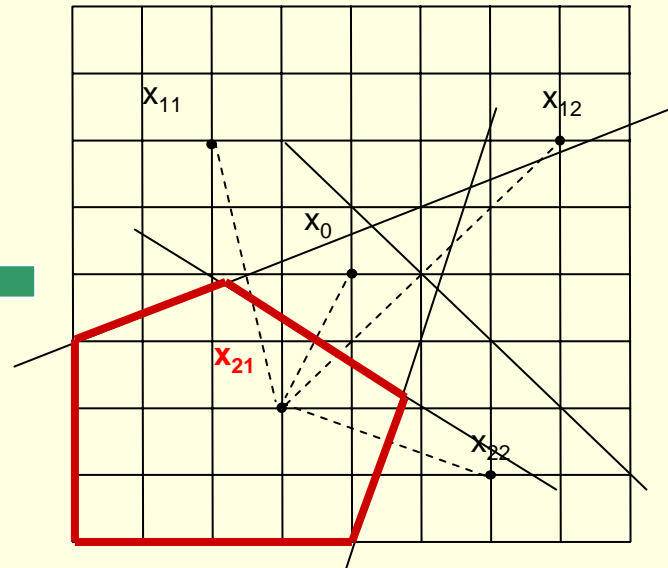
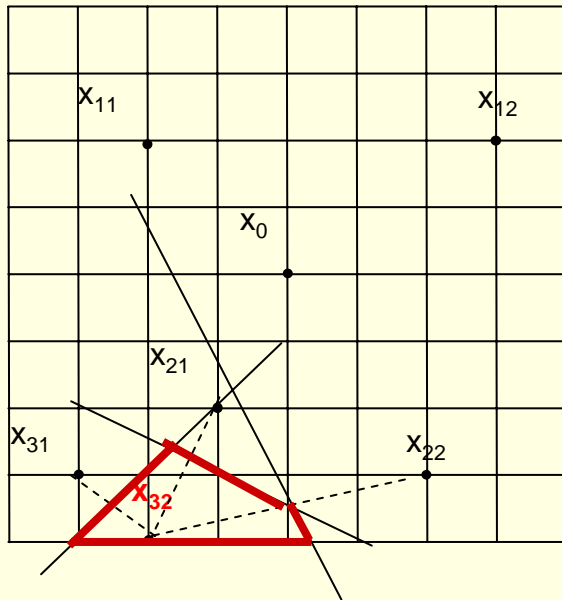
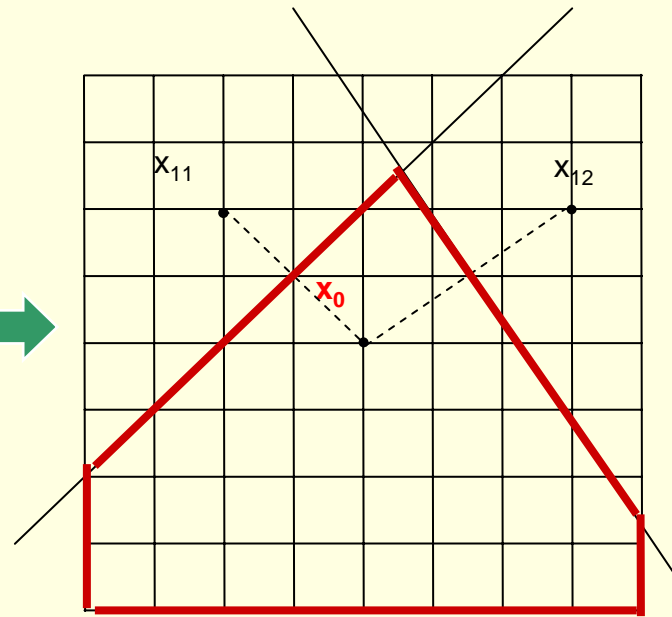
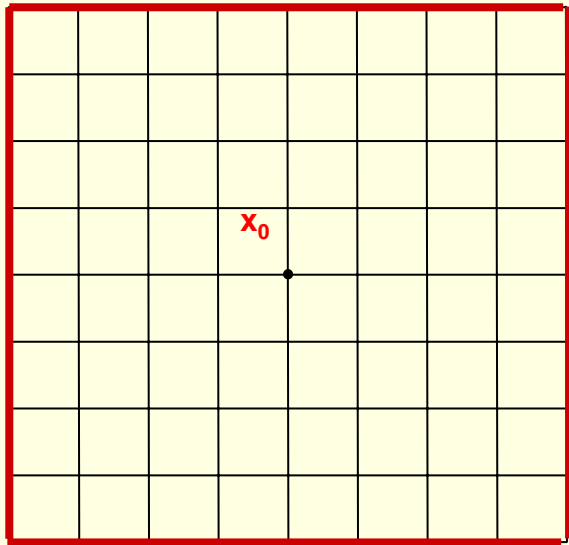
Example:  
Increase or  
decrease the  
number of  
purple shirts  
by 1



# COMPASS Algorithm

---

- **Convergent Optimization via Most Promising Area Stochastic Search** (Hong and Nelson 2006)
  1. Build the most promising area in each iteration around the **current sample best solution** based on geometry.
  2. Sample new solutions from the **most promising area** in each iteration.
  3. Simulate **all** sampled solutions a little bit more.
  4. Calculate the **cumulative sample mean** for each solution, and choose the solution with the best cumulative sample mean.



# Framework for LCRS Algorithms

---

- COMPASS is a specific instance of a general framework for locally convergent random search (LCRS) algorithms (Hong and Nelson 2007)
- The framework provides conditions on...
  - Sampling solutions: solutions in the neighborhood of the sample best must have a chance
  - Simulating solutions: the current best, its visited neighbors and all newly sampled solutions must continue to get more simulation
- Speed ups and smart heuristics can be embedded within the framework without spoiling convergence



# Properties of Local Convergence

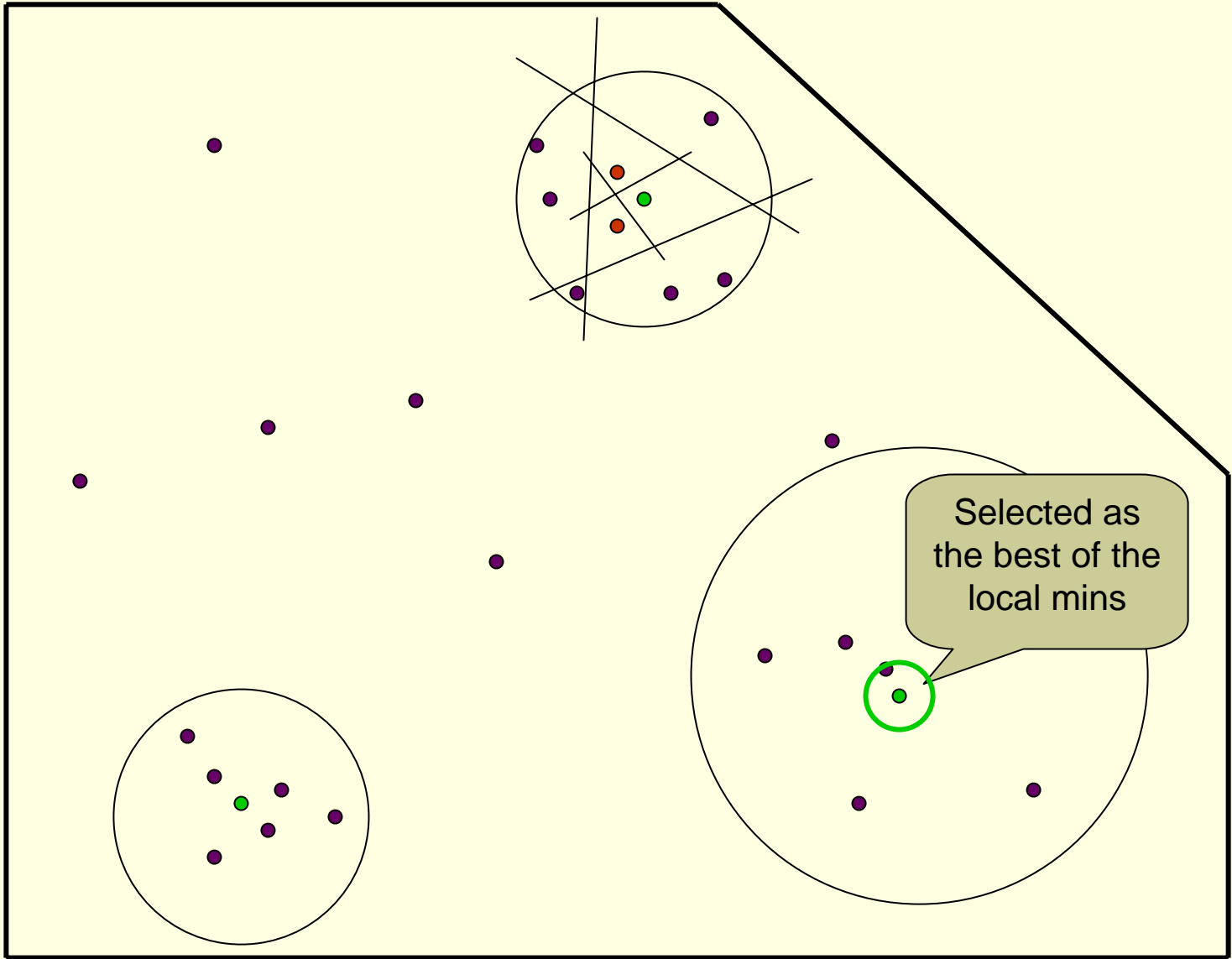
---

- LCRS algorithms often converge fast
- They can be used to design stopping criterion
  - When all solutions in the local neighborhood of a solution are visited and the solution appears to be better than its neighbors
  - Xu et al. (2010) designed a selection procedure to test the local optimality
- Of course: the algorithms may only find local optimal solutions that are much worse than global optimal solutions

# Industrial Strength COMPASS

---


- **Global Phase:** explore the feasible region with a globally convergent algorithm looking for promising subregions
  - Transition based on effort and quality rules
- **Local Phase:** take the promising regions as input to a locally convergent algorithm
  - Transition when locals found with high confidence
- **Clean-Up Phase:** Select & estimate the best
  - Sample more to guarantee PCS and  $\pm\delta$  error



Industrial Strength COMPASS - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.iscompass.net



# Industrial Strength COMPASS (ISC)

---

[Overview](#)

[Papers](#)

[Instructions](#)

[ISC Code](#)

[Contact Information](#)

## Overview

Industrial Strength COMPASS (ISC) is open source code for maximizing or minimizing the expected value of a single performance measure generated by a stochastic simulation with respect to integer ordered decision variables subject to linear-integer constraints. ISC is a particular implementation of the COMPASS framework developed by Hong and Nelson (see the link to papers) for locally convergent, discrete optimization-via-simulation (DOvS) algorithms. ISC was developed by Jie Xu.

The ISC software is distributed "as is," without warranties of any kind, either express or implied. Please report problems via the contact link.

The software is copyrighted by L. Jeff Hong, Barry L. Nelson and Jie Xu 2007. The authors grant permission for unlimited personal use of this software without fee. However, no derivative works based on the software may be prepared, including embedding any portion of it in another software product, without permission of the copyright holders.

Last update 4/27/2008

---

Done

start | Inbox - M... | My eBooks | WinEdt/... | Yap 2.4.1... | Microsoft ... | Industrial... | EN | 9:13 AM

[www.iscompass.net](http://www.iscompass.net)

# Outline

---

- Problem definition and classification
- Selection of the best
- Stochastic approximation and gradient estimation
- Random search algorithms
- Working with commercial solvers
- Conclusions

# Status of Commercial OvS Solvers

---

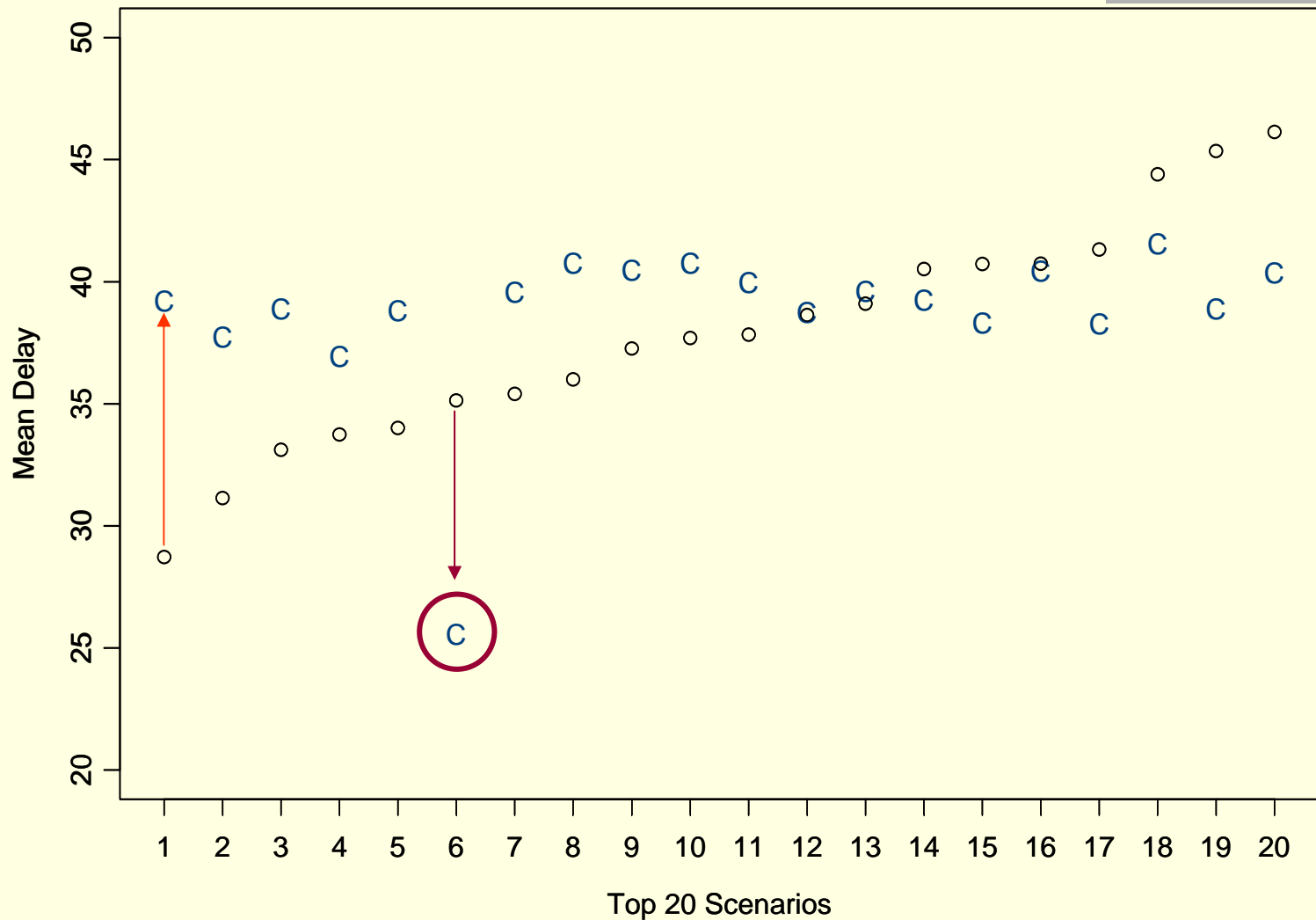
- Many simulation products have integrated OvS software
  - OptQuest is in Arena, Flexsim, SIMUL8, etc.
  - ProModel uses SimRunner
  - AutoMod uses AutoStat
- Robust heuristics are commonly used
  - OptQuest uses scatter search, neural network, tabu search
  - SimRunner and AutoStat both use evolutionary, genetic algorithms
- Easy to use on real, complex simulations
- No statistical guarantees on OvS problems

# Suggestions on Using OvS Solvers

---

- Controlling sampling variability
  - Simulation experiments are random
  - Use a preliminary experiment to decide an appropriate sample size for each solution
- Restarting the optimization
  - Heuristic algorithms may find different solutions on different runs because they have no provable convergence
  - Run the algorithms multiple times from different starting solutions and using different random number streams
- Statistical clean up
  - Perform a second set of experiments on top solutions
  - Better selects the best solution and estimates its value

# Why “clean up” is so important





# Outline

---

- Problem definition and classification
- Selection of the best
- Stochastic approximation and gradient estimation
- Random search algorithms
- Working with commercial solvers
- **Conclusions**

# Conclusions

---

- A lot of work has been done in the research community with a focus on...
  - Convergence properties
  - Statistical guarantees
  - Designing simple algorithms
- Commercial solvers mainly use heuristics having
  - Robust performance
  - No statistical or convergence guarantees
- ISC is an early attempt to bridge that gap