

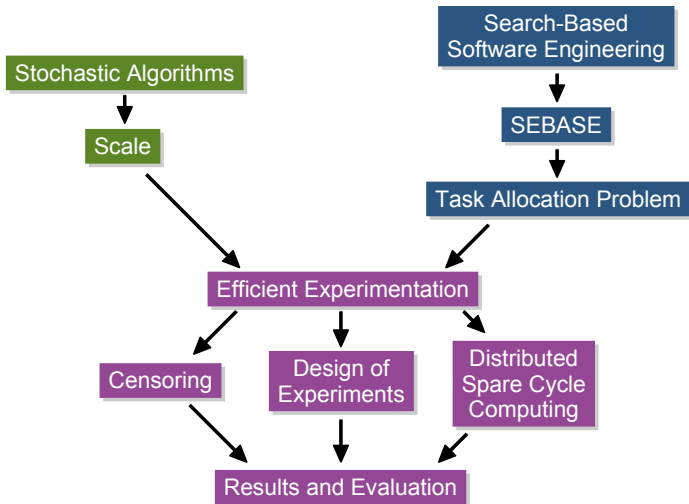
# An Efficient Approach to Large-Scale Experimentation on Stochastic Algorithms

Simon Poulding

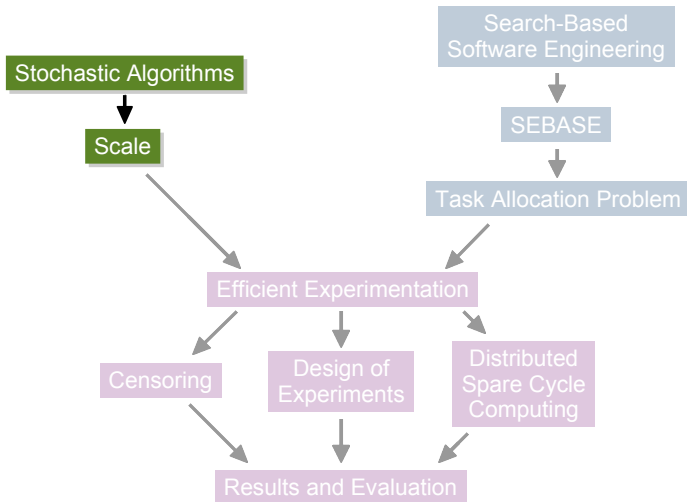
Non-Standard Computation Group  
Computer Science  
University of York

19 October 2007

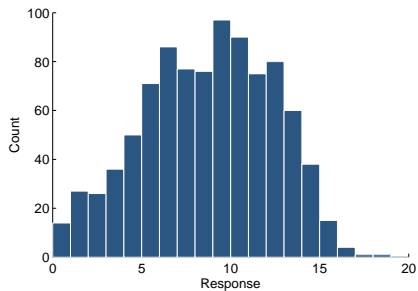
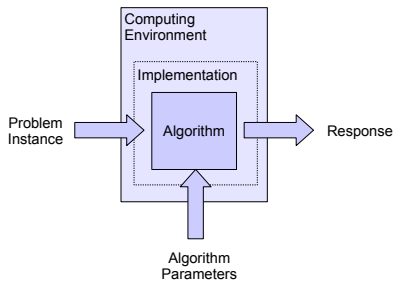
# Overview



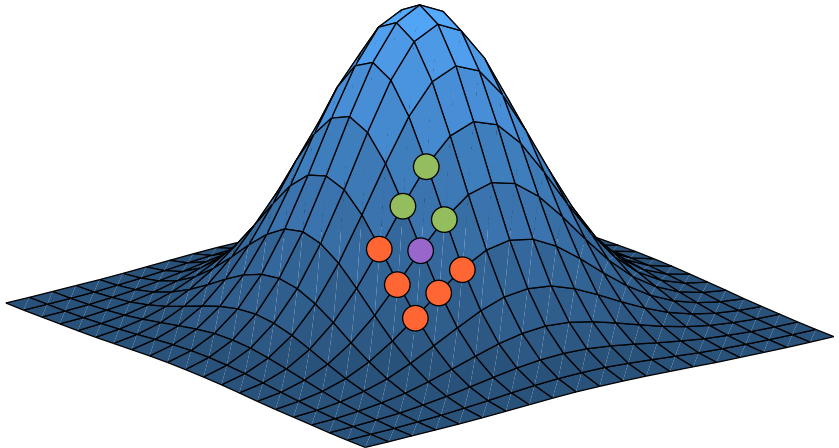
# Large-Scale Stochastic Algorithms



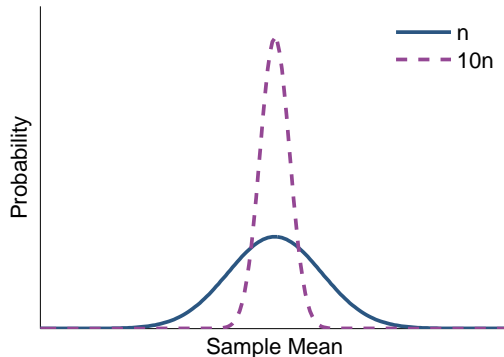
# Response Variance



# Simulated Annealing



# Handling Variance - Sample Size



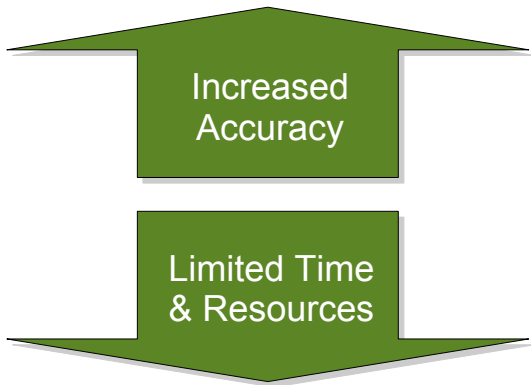
## Sample Size

For an increase in sample size by a factor  $k$ , the variance of the sample mean decreases by a factor of  $1/k$ .

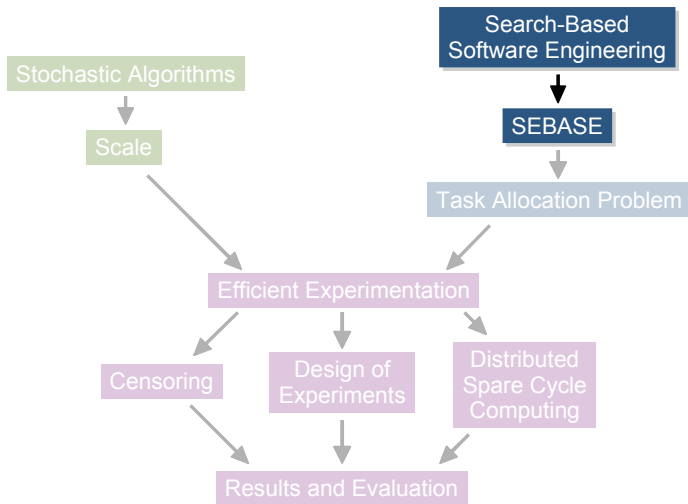
As the **scale** of the problem gets larger:

- algorithm takes longer to find a solution
- more computing resources are required

# Number of Experiments



# Search-Based Software Engineering and SEBASE

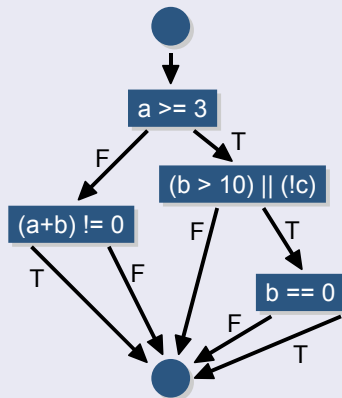


## Principle

If a software engineering problem can be expressed using a fitness function, then search algorithms may be used to find solutions with the desired fitness.

# SBSE Example - Branch Coverage

## Program



## Test Data Set

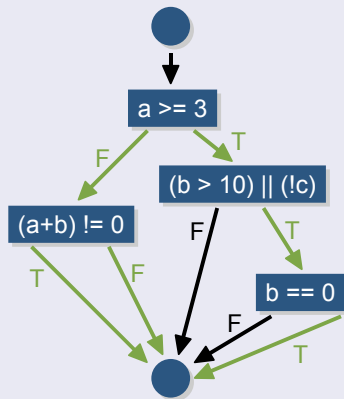
a	b	c
2	1	true
3	0	false
1	-1	false

## Fitness Metric

Proportion of branches (decisions) exercised by data set

# SBSE Example - Branch Coverage

## Program



## Test Data Set

a	b	c
2	1	true
3	0	false
1	-1	false

## Fitness Metric

Proportion of branches (decisions) exercised by data set

0.75

## Applications

Requirements Prioritisation  
Architectural Design  
Program Creation  
Test Data Generation  
Program Transformation  
Project Management  
Release Definition  
Protocol Derivation

## Algorithms

**deterministic** calculus  
operational research

**stochastic** Genetic Algorithms  
Genetic Programming  
Simulated Annealing  
Grammatical Evolution  
EDAs  
hill climbing

**hybrid** e.g. GAs + greedy



Software Engineering By Automated SEarch

[www.sebase.org](http://www.sebase.org)

UNIVERSITY OF  
BIRMINGHAM

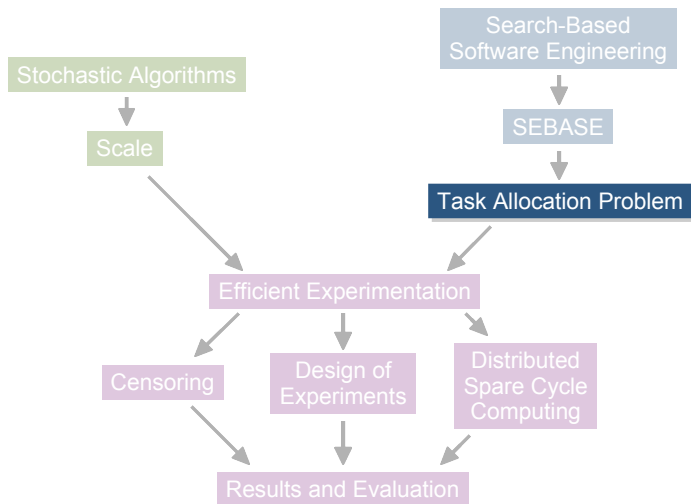
KING'S  
*College*  
LONDON

THE UNIVERSITY *of* York

# SEBASE Objectives

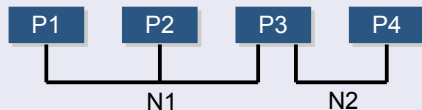
- New application areas
- Novel algorithms
- Cross-cutting themes
  - scale
  - robustness
  - problem characterisation
  - insight
  - understanding

# Task Allocation Problem

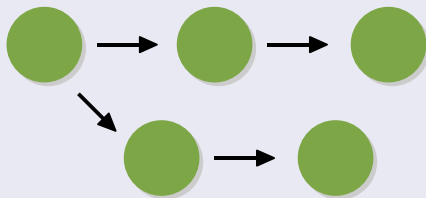


# Task Allocation Problem

## Hardware

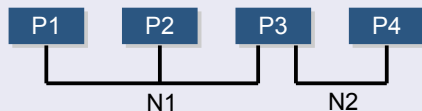


## Tasks and Messages

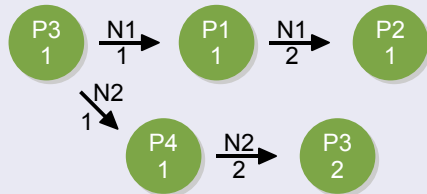


# Task Allocation Problem

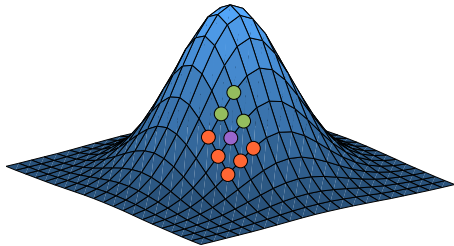
## Hardware



## Tasks and Messages



# Solution Using Simulated Annealing (Paul Emberson)



## Neighbourhood

Change to allocation (processor/network) or priority of a task or message.

## Cost Function

$$c = \sum_{i=1}^9 w_i g_i$$

$g_i$  cost function components  
 $w_i$  weightings ( $\sum w_i = 1$ )

## Question

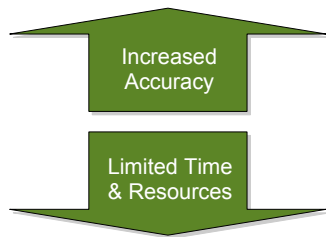
Which weighting vector,  $\mathbf{w} = (w_1, w_2, \dots, w_9)$ , enables the algorithm to solve the task allocation problem in the minimum number of cost function evaluations?

(Considered on average over a set of randomly generated problem instances with specific properties.)

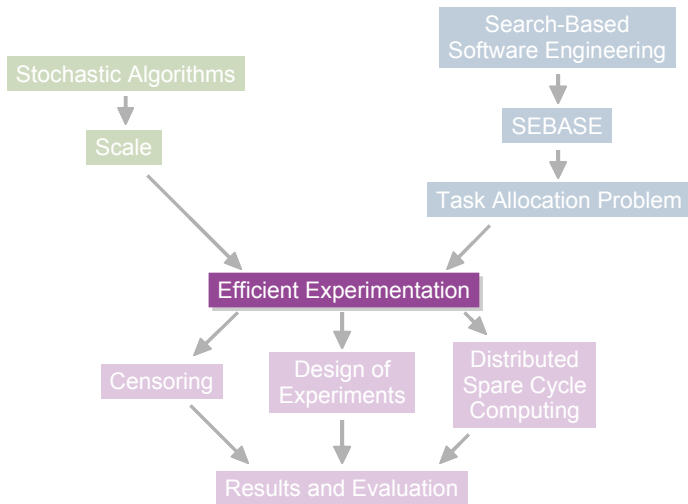
# Initial Experimentation

Response showed a great deal of variability

Individual algorithm **trials** could take more than 2 days to find a solution

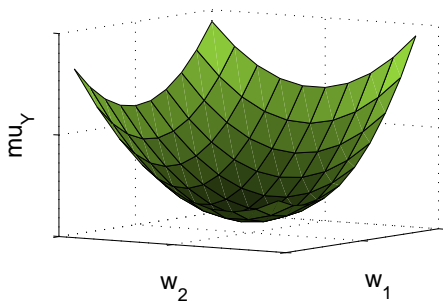


# Efficient Experimentation



# Strategy

- 1 Find approximation for the function:  $\mu_Y = f(w_1, w_2, \dots, w_9)$



- 2 Find  $\mathbf{w}$  that minimises  $f(\cdot)$  using **efficient** OR methods

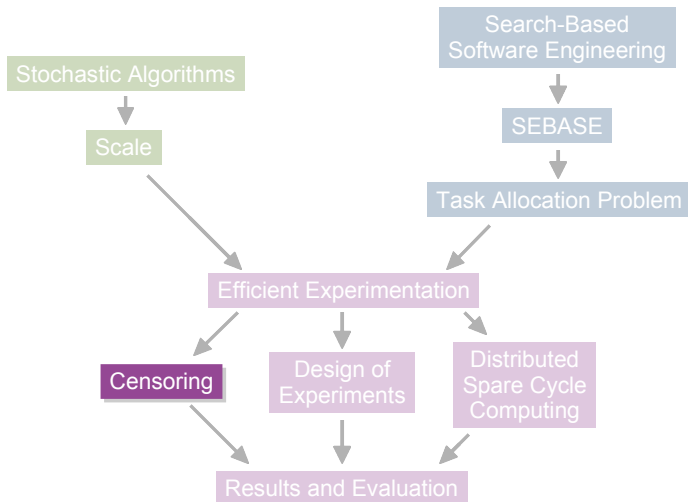
# Techniques For Model Fitting

**censoring** Reduces computing resources;  
maintains accuracy

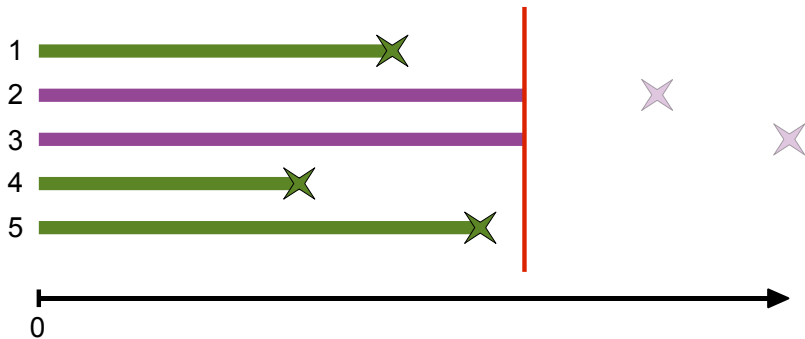
**design of experiments** Efficient experimental design  
reduces number of trials

**distributed spare cycle computing** Large pool of computing  
resources at little cost

# Censoring



# Censoring

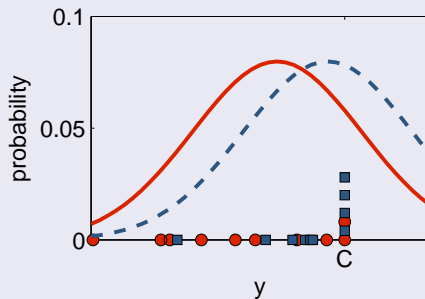


# Tobit Model

## Linear Model

$$Y = \beta_0 + \sum_{i=1}^9 \beta_i w_i + \varepsilon$$

## Maximum Likelihood Estimation

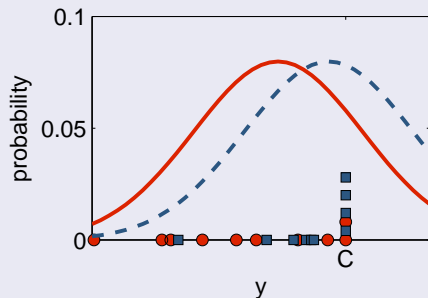


# Tobit Model

## Linear Model

$$\log(Y) = \beta_0 + \sum_{i=1}^9 \beta_i w_i + \varepsilon$$

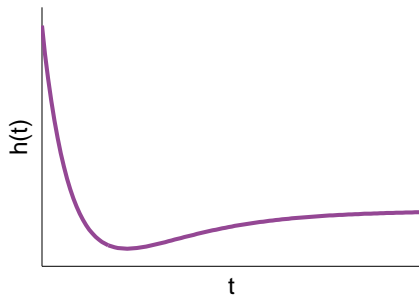
## Maximum Likelihood Estimation



# Cox's Proportional Hazards Model

## Hazard Function

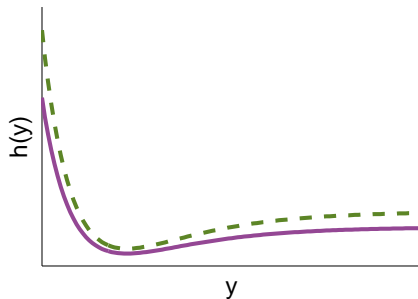
$h(t)$  = probability of event occurring at time  $t$  given it has not already occurred



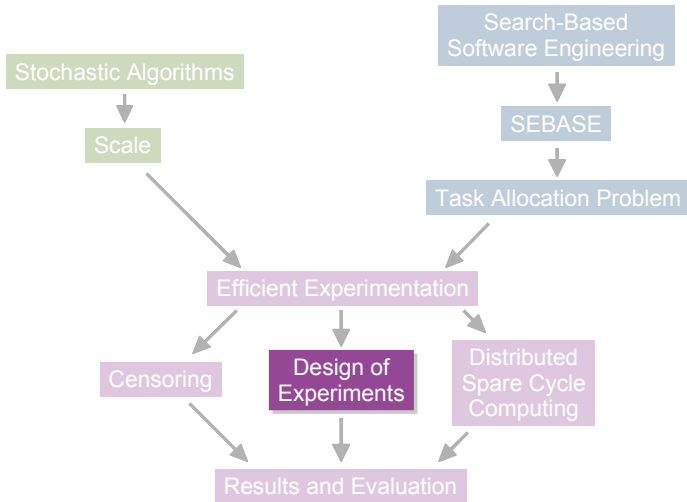
# Cox's Proportional Hazards Model

## Proportional Hazards Model

$$h(y; \mathbf{w}) = \exp \left\{ \beta_0 + \sum_{i=1}^9 \beta_i w_i \right\} h_0(y)$$



# Design of Experiments



# Design of Experiments

- ① Define detailed model
- ② Decide which trials to run

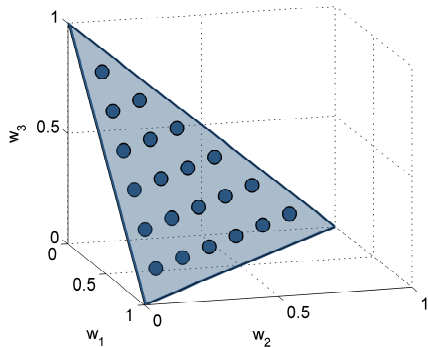
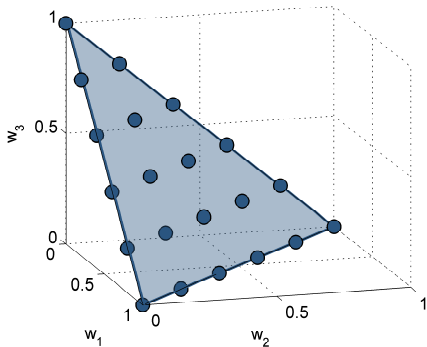
## Weighting Constraint

$$\sum_{i=1}^9 w_i = 1$$

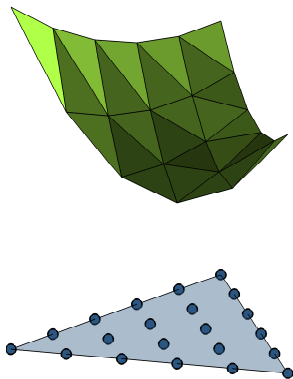
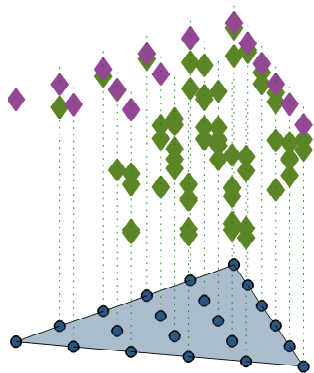
## Mixture Model Canonical Form (Scheffé)

$$\sum_{i=1}^9 \beta_i w_i + \sum_{i=1}^9 \sum_{j=i+1}^9 \beta_{ij} w_i w_j$$

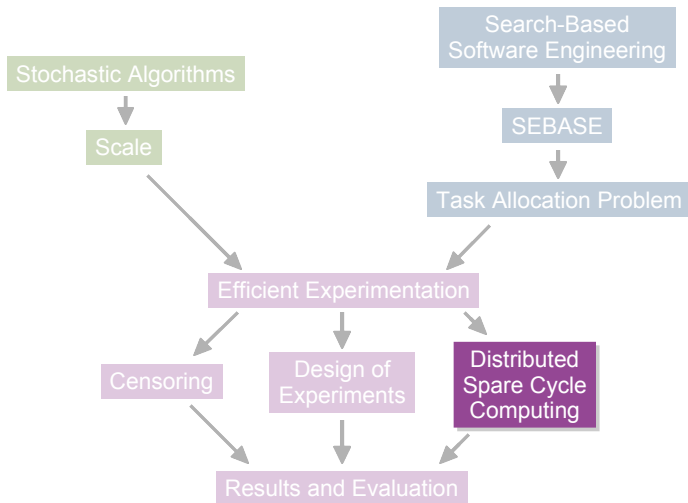
# Simplex Lattice Design

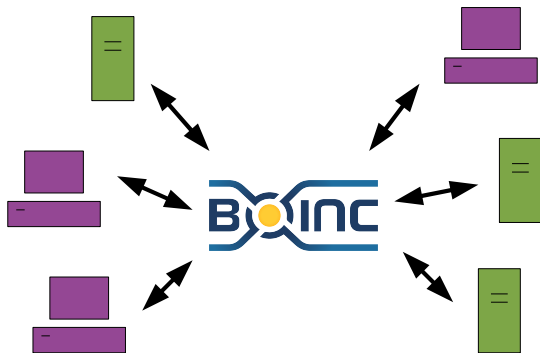


# Experiments

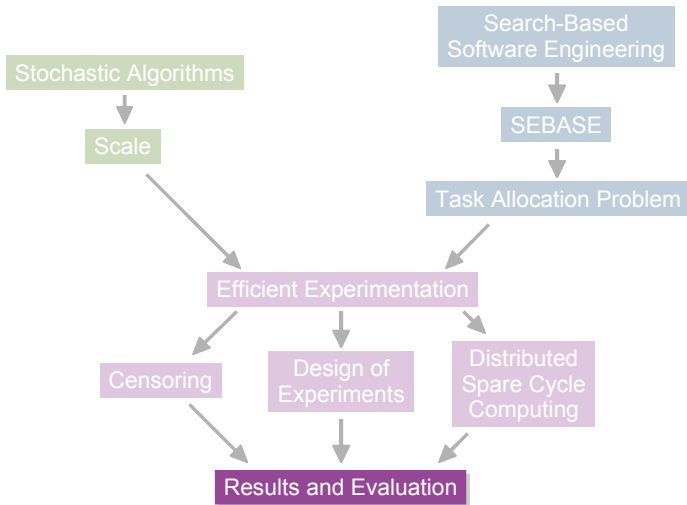


# Distributed Spare Cycle Computing

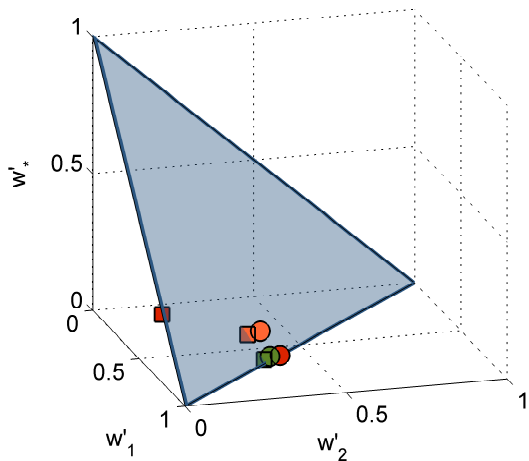




# Results and Evaluation



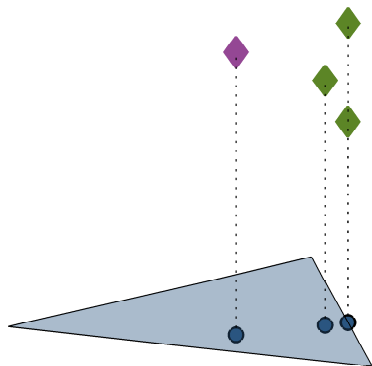
# Optimal Weightings



- Tobit
- Cox PH

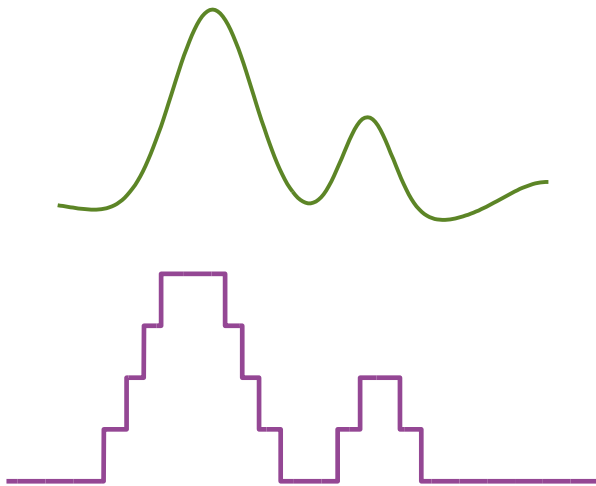
$$w_* = \frac{1}{7} \sum_{i=3}^9 w_i$$

# Verification

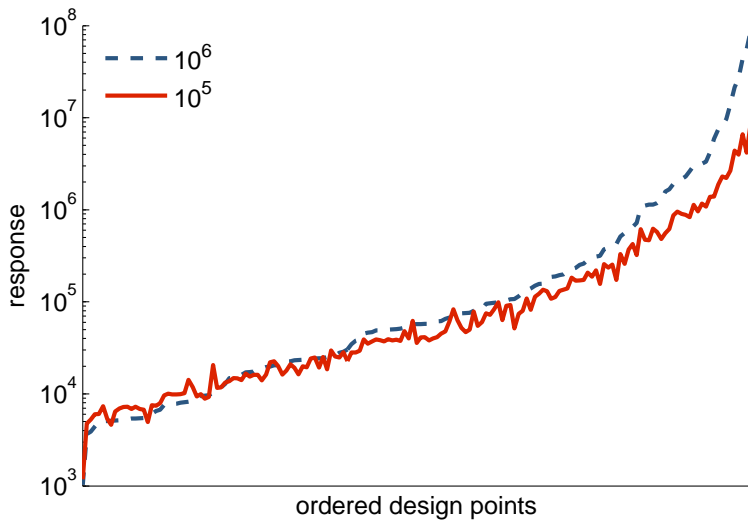


Optimal weightings resulted in 33% better algorithm performance over 'trial-and-error' best weightings

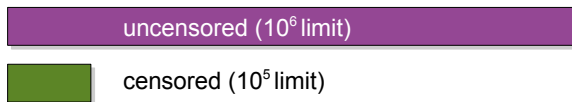
# Explanation



# Censoring Accuracy

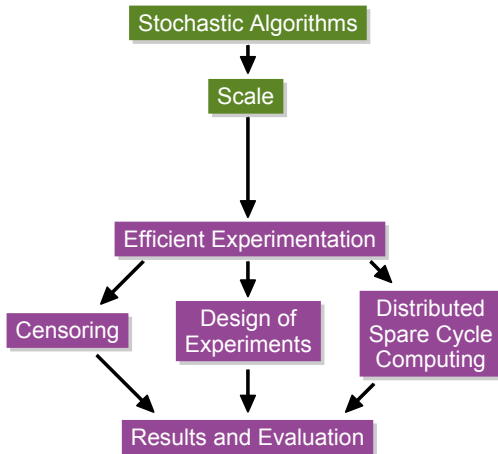


# Censoring Efficacy



Censoring reduced experimental processing requirements by over 85%

# Conclusion



# Thanks

Many thanks to all our colleagues who donated spare cycles on their computers.



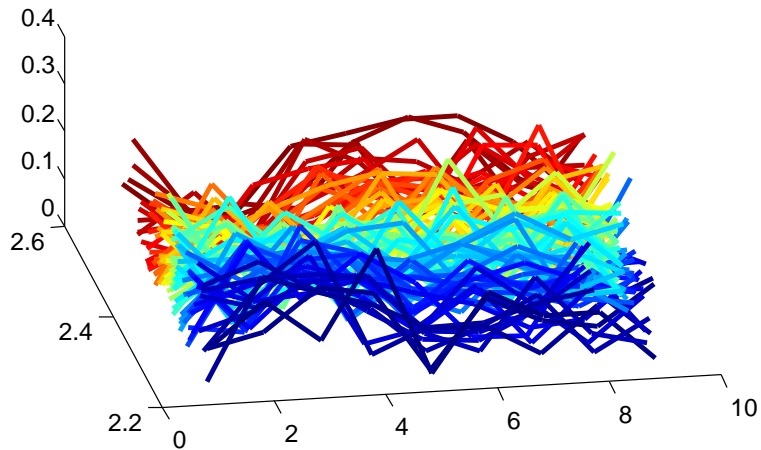
## Mixture Model

$$\sum_{i=1}^9 \beta_i w_i + \sum_{i=1}^9 \sum_{j=i+1}^9 \beta_{ij} w_i w_j$$

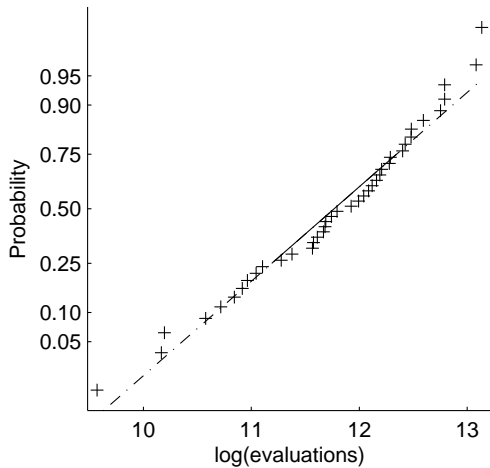
## Draper St. John Model

$$\sum_{i=1}^9 \beta_i w_i + \sum_{i=1}^9 \sum_{j=i+1}^9 \beta_{ij} w_i w_j + \sum_{i=1}^9 \gamma_i w_i^{-1}$$

# Parallel Coordinates



# Normplot



# Model Fit

