Performance Optimization of a Parallel, Two Stage Stochastic Linear Program: The Military Aircraft Allocation Problem

Akhil Langer, Ramprasad Venkataraman, Udatta Palekar, Laxmikant Kale, *Steven Baker

Parallel Programming Laboratory
University of Illinois

*MITRE Corp

ICPADS 2012

Stochastic Programming

Linear Program (LP)

Cost minimization under constraints

$$\begin{aligned} \min \quad & cx \\ s.t. \quad & Ax \leq b, x \in \mathbb{R}^n \end{aligned}$$

Stochastic Programming

Linear Program (LP)

Cost minimization under constraints

$$\begin{aligned} \min \quad & cx \\ s.t. \quad & Ax \leq b, x \in \mathbb{R}^n \end{aligned}$$

In many real-world applications - A, b, c are unknown

- e.g. agricultural planning, investment decisions, transportation, etc.

Stochastic Programming

Linear Program (LP)

Cost minimization under constraints

$$\begin{aligned} \min \quad & cx \\ s.t. \quad & Ax \leq b, x \in \mathbb{R}^n \end{aligned}$$

In many real-world applications - A, b, c are unknown

- e.g. agricultural planning, investment decisions, transportation, etc.
- but known probabilistic distributions

Stochastic Programming

Linear Program (LP)

Cost minimization under constraints

$$\min \quad cx$$
$$s.t. \quad Ax \leq b, x \in \mathbb{R}^n$$

In many real-world applications - A, b, c are unknown

- e.g. agricultural planning, investment decisions, transportation, etc.
- but known probabilistic distributions

Stochastic Program

divide into certain and uncertain parameters

Scenario: a particular realization of the uncertain parameters

$$\begin{aligned} \min \quad & cx + E_s[q_s y_s] \\ s.t. \quad & Ax = b, \\ & T_s x + W_s y_s = h_s, \quad s = 1, ..., S \\ & x \geq 0, y_s \geq 0, \quad s = 1...., S \end{aligned}$$

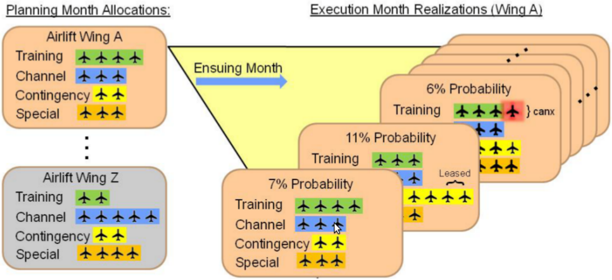
Military Aircraft Allocation

US Air Mobility Command (AMC) handles fleet of 1300 aircrafts:

- Worldwide Airlift
- Worldwide Air-Refueling
- Aeromedical Evacuation
- Presidential and DV Support
- Civil Reserve AirFleet (CRAF)

Military Aircraft Allocation

Myriad possible outcomes confound decision support, e.g. aircraft breakdowns, weather, natural disasters, conflicts, etc.



The Tanker Airlift Control Center (TACC) must reconcile diverse uncertainy when predicting monthly aircraft allocation

Military Aircraft Allocation

Sample Model Sets (120 scenarios)

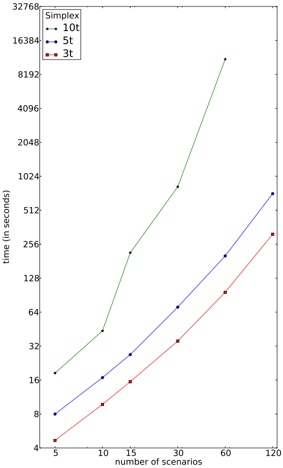
Model Name	Num variables	Num constraints
$3t$	1076655	668640
$5t$	1663785	1064280
$10t$	3069330	1988640
$15t$	4157835	2805000
$30t$	7957950	5573400

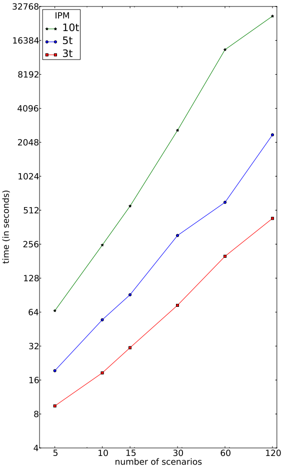
Available in Stochastic MPS Format (SMPS) at http://charm.cs.uiuc.edu/jetAlloc

Documentation: http://www.mitre.org/work/tech_papers/2012/11_5412/

Solving as a Linear Program (Extensive Formulation)

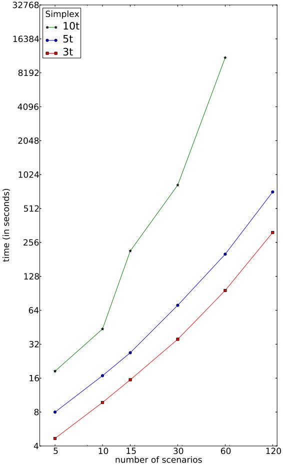
Optimization time using *Simplex* and *Interior Point Methods (IPM)* of Gurobi optimizer (1 processor)

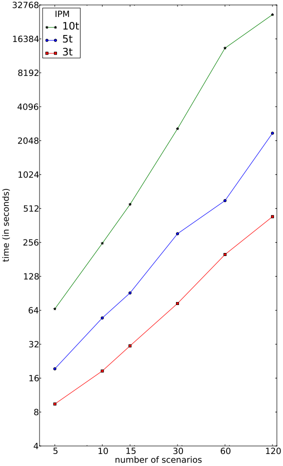




Solving as a Linear Program (Extensive Formulation)

Optimization time using *Simplex* and *Interior Point Methods (IPM)* of Gurobi optimizer (1 processor)





Superlinear increase in time with # scenarios

Problem Statement

Efficient parallelization of the two-stage stochastic programs

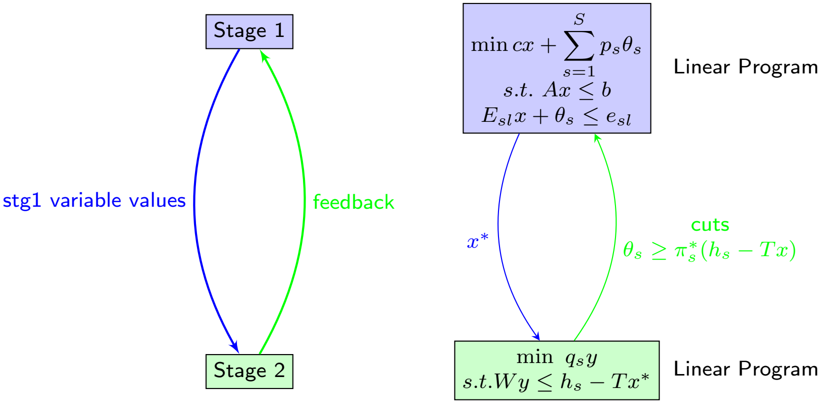
Two-stage Stochastic Programs

- Stage 1 - strategic decisions
  - Aircraft allocation - mission, location, day
- Stage 2 - operational decisions
  - Aircraft scheduling - meeting mission demands

$$\begin{aligned} \min \quad & cx + \sum_{s=1}^{S} p_s Q_s(x) \\ s.t. \quad & Ax \leq b \end{aligned}$$

where, $Q_s(x) = \min\{q_s y | W_s y \leq h_s - T_s x\}, \qquad s = 1...S$

Benders Decomposition



Parallel Design

Implementation

- Charm++[1] as the parallel programming framework
  - express computation as interacting collection of objects
  - one-sided communication and asynchronous computation
- Delegate individual LP solves to highly optimized LP library e.g. Gurobi[2]

---

[1] charm.cs.uiuc.edu

Kale et.al. Migratable Objects + Active Messages + Adaptive Runtime = Productivity + Performance. A Submission to 2012 HPC Class II Challenge.

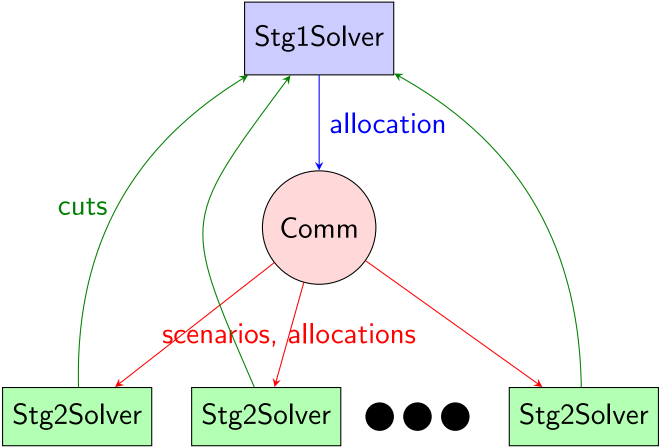
[2] www.gurobi.com

Parallel Design

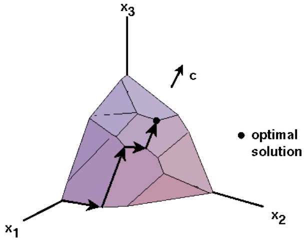
Design

- Stage 1 Solver
  - Allocation Generator
- Stage 2 Solver
  - Scenario Evaluator
- Communicator
  - Work Allocator

Parallel Design



Advanced Starts



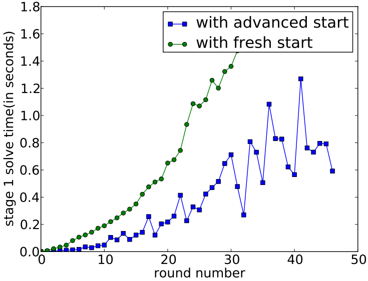
- Start from a prespecified basis and solution
- saves computation of initial feasible basis
- number of simplex iterations depends on distance from optimal solution

[1] picture borrowed from "Mysteries in Linear Programming", K. Fukuda

Optimizing Stage 1

Advanced Starts

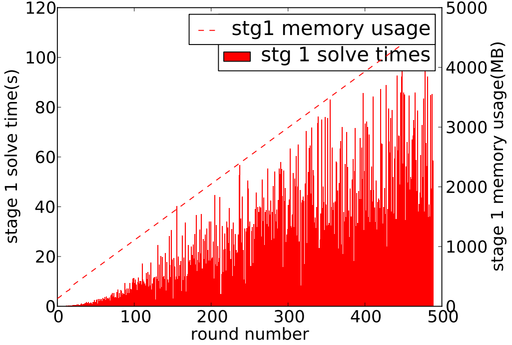
Start from basis of the optimal solution of the previous iteration



Faster Stage 1 LP solves with advanced start

Optimizing stage 1

Memory Footprint



Increasing Memory Footprint with iteration Number

Optimizing Stage 1

Curbing Solver Memory Footprint

Active cuts - cuts that influence the final result of the optimization

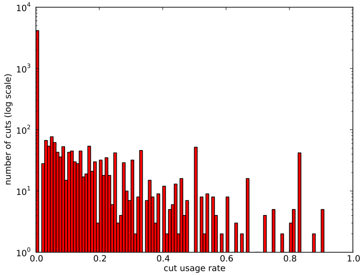
$$\text{Cut Usage Rate} = \frac{\text{num rounds in which cut is active}}{\text{num rounds since its generation}}$$

Optimizing Stage 1

Curbing Solver Memory Footprint

Active cuts - cuts that influence the final result of the optimization

$$\text{Cut Usage Rate} = \frac{\text{num rounds in which cut is active}}{\text{num rounds since its generation}}$$



Cut usage rate is very low for large fraction of the cuts

Optimizing Stage 1

Cut Retirement

Discard Cuts with low usage rate whenever total number of cuts exceed a configurable threshold

Optimizing Stage 1

Cut Retirement

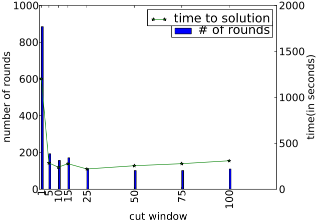
Discard Cuts with low usage rate whenever total number of cuts exceed a configurable threshold

Time to solution: $19ks -> 8ks$, 57% improvement

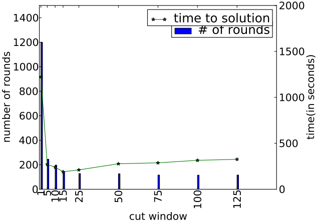
Optimizing Stage 1

Effect of cut-window

max number of cuts = (cut-window)*(number of scenarios)



(a) 5 time period model



(b) 10 time period model

Optimizing Stage 1

Evaluating Cut-Retirement Strategies

Least Frequently Used (LFU)

$$\text{Cut Usage Rate} = \frac{\text{num rounds in which cut is active}}{\text{num rounds since its generation}}$$

Least Recently Used (LRU)

$$LRU\_Score = \text{Last active round for the cut}$$

Least Recently/Frequently Used (LRFU)[2]

$$LRFU\_Score = \sum_{i=1}^{k} \mathcal{F}(t_{base} - t_i)$$

Memory and time consuming!!

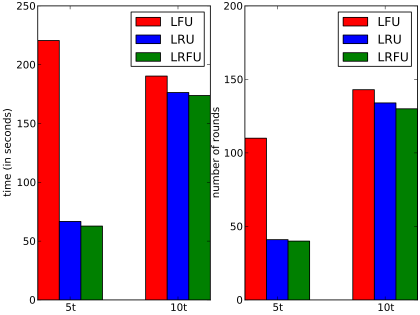
Approximation, $\mathcal{F}(x) = (\frac{1}{p})^{\lambda x} (p \geq 2)$,

$$S_{t_k} = \mathcal{F}(0) + \mathcal{F}(\delta) S_{t_{k-1}}, \delta = t_k - t_{k-1}$$

[2] C.S. Kim. LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies. IEEE Transactions on Computers, 50(12), 2001

Optimizing Stage 1

Evaluating Cut-Retirement Strategies



Performance of different cut scoring strategies for 5 and 10 time period model

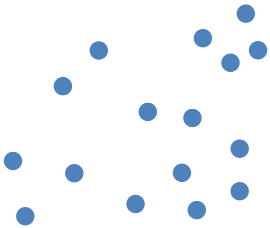
Optimizing Stage 2

Stage 2 constitutes significant fraction of total computation

- Dual polytope remains the same
- Use advanced start
- Evaluate similar scenarios in succession
- Cluster scenarios into equal sized clusters

Optimizing Stage 2

The Scenario Clustering Algorithm

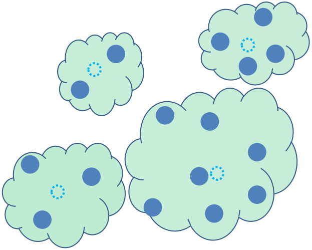


Algorithm 1 The Scenario Clustering Algorithm

Input
D_i- Demand set for scenario i $(i = 1, 2, ..., n)$
k - number of clusters
Output
k equally sized clusters of scenarios
Algorithm
{label, centroids} = kMeans($\{D_1, D_2, D_3, ..., D_n\}$, k)
IdealClusterSize = $\frac{n}{k}$
$size_i$ = size of cluster i
{Identify Oversized clusters}
$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$
{Identify Undersized clusters}
$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$
$\mathcal{S}$: set of adjustable points
for $c \in \mathcal{O}$ **do**
  Find $(size_i - IdealClusterSize)$ points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$
end for
while $size(\mathcal{S}) > 0$ **do**
  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$
  Add p to cluster c
  Remove p from $\mathcal{S}$
  if $size_c == IdealClusterSize$ **then**
    Remove c from $\mathcal{U}$
  end if
end while

Optimizing Stage 2

The Scenario Clustering Algorithm

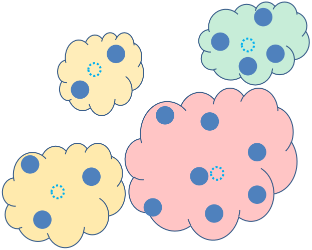


Algorithm 1 The Scenario Clustering Algorithm

Input
D_i- Demand set for scenario i ($i = 1, 2, ..., n$)
k - number of clusters
Output
k equally sized clusters of scenarios
Algorithm
{label, centroids} = kMeans({D_1, D_2, D_3, ..., D_n}, k)
IdealClusterSize = $\frac{n}{k}$
$size_i$ = size of cluster i
{Identify Oversized clusters}
$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$
{Identify Undersized clusters}
$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$
$\mathcal{S}$: set of adjustable points
for $c \in \mathcal{O}$ **do**
  Find $(size_i - IdealClusterSize)$ points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$
end for
while $size(\mathcal{S}) > 0$ **do**
  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$
  Add p to cluster c
  Remove p from $\mathcal{S}$
  if $size_c == IdealClusterSize$ **then**
    Remove c from $\mathcal{U}$
  end if
end while

Optimizing Stage 2

The Scenario Clustering Algorithm



Algorithm 1 The Scenario Clustering Algorithm

Input

D_i- Demand set for scenario i $(i = 1, 2, ..., n)$

k - number of clusters

Output

k equally sized clusters of scenarios

Algorithm

{label, centroids} = kMeans($\{D_1, D_2, D_3, ..., D_n\}$, k)

IdealClusterSize $= \frac{n}{k}$

$size_i$ = size of cluster i

{Identify Oversized clusters}

$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$

{Identify Undersized clusters}

$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$

$\mathcal{S}$: set of adjustable points

for $c \in \mathcal{O}$ **do**

  Find $(size_i - IdealClusterSize)$ points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$

end for

while $size(\mathcal{S}) > 0$ **do**

  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$

  Add p to cluster c

  Remove p from $\mathcal{S}$

  if $size_c == IdealClusterSize$ **then**

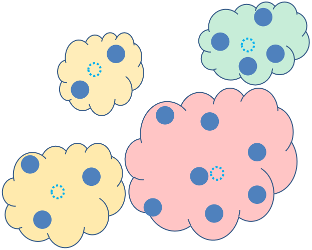
    Remove c from $\mathcal{U}$

  end if

end while

Optimizing Stage 2

The Scenario Clustering Algorithm



Algorithm 1 The Scenario Clustering Algorithm

Input

D_i- Demand set for scenario i ($i = 1, 2, ..., n$)

k - number of clusters

Output

k equally sized clusters of scenarios

Algorithm

{label, centroids} = kMeans({D_1, D_2, D_3, ..., D_n}, k)

IdealClusterSize $= \frac{n}{k}$

$size_i$ = size of cluster i

{Identify Oversized clusters}

$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$

{Identify Undersized clusters}

$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$

$\mathcal{S}$: set of adjustable points

for $c \in \mathcal{O}$ **do**

  Find $(size_i - IdealClusterSize)$ points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$

end for

while $size(\mathcal{S}) > 0$ **do**

  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$

  Add p to cluster c

  Remove p from $\mathcal{S}$

  if $size_c == IdealClusterSize$ **then**

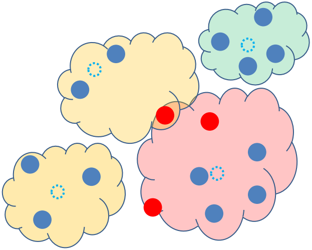
    Remove c from $\mathcal{U}$

  end if

end while

Optimizing Stage 2

The Scenario Clustering Algorithm



Algorithm 1 The Scenario Clustering Algorithm

Input

D_i- Demand set for scenario i ($i = 1, 2, ..., n$)

k - number of clusters

Output

k equally sized clusters of scenarios

Algorithm

{label, centroids} = kMeans({D_1, D_2, D_3, ..., D_n}, k)

IdealClusterSize = $\frac{n}{k}$

$size_i$ = size of cluster i

{Identify Oversized clusters}

$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$

{Identify Undersized clusters}

$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$

$\mathcal{S}$: set of adjustable points

for $c \in \mathcal{O}$ **do**

  Find $(size_i - IdealClusterSize)$ points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$

end for

while $size(\mathcal{S}) > 0$ **do**

  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$

  Add p to cluster c

  Remove p from $\mathcal{S}$

  if $size_c == IdealClusterSize$ **then**

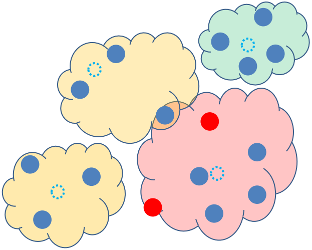
    Remove c from $\mathcal{U}$

  end if

end while

Optimizing Stage 2

The Scenario Clustering Algorithm

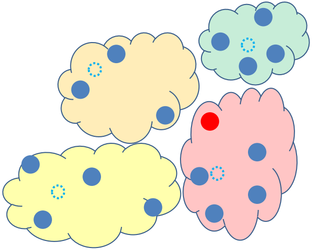


Algorithm 1 The Scenario Clustering Algorithm

Input
D_i- Demand set for scenario i ($i = 1, 2, ..., n$)
k - number of clusters
Output
k equally sized clusters of scenarios
Algorithm
{label, centroids} = kMeans($\{D_1, D_2, D_3, ..., D_n\}$, k)
IdealClusterSize $= \frac{n}{k}$
$size_i$ = size of cluster i
{Identify Oversized clusters}
$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$
{Identify Undersized clusters}
$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$
$\mathcal{S}$: set of adjustable points
for $c \in \mathcal{O}$ **do**
  Find $(size_i - IdealClusterSize)$ points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$
end for
while $size(\mathcal{S}) > 0$ **do**
  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$
  Add p to cluster c
  Remove p from $\mathcal{S}$
  if $size_c == IdealClusterSize$ **then**
    Remove c from $\mathcal{U}$
  end if
end while

Optimizing Stage 2

The Scenario Clustering Algorithm

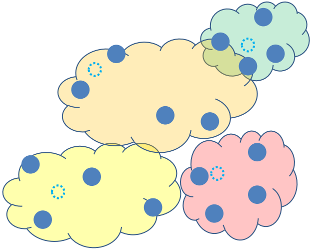


Algorithm 1 The Scenario Clustering Algorithm

Input
D_i- Demand set for scenario i ($i = 1, 2, ..., n$)
k - number of clusters
Output
k equally sized clusters of scenarios
Algorithm
{label, centroids} = kMeans($\{D_1, D_2, D_3, ..., D_n\}$, k)
IdealClusterSize $= \frac{n}{k}$
$size_i$ = size of cluster i
{Identify Oversized clusters}
$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$
{Identify Undersized clusters}
$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$
$\mathcal{S}$: set of adjustable points
for $c \in \mathcal{O}$ **do**
  Find $(size_i - IdealClusterSize)$ points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$
end for
while $size(\mathcal{S}) > 0$ **do**
  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$
  Add p to cluster c
  Remove p from $\mathcal{S}$
  if $size_c == IdealClusterSize$ **then**
    Remove c from $\mathcal{U}$
  end if
end while

Optimizing Stage 2

The Scenario Clustering Algorithm

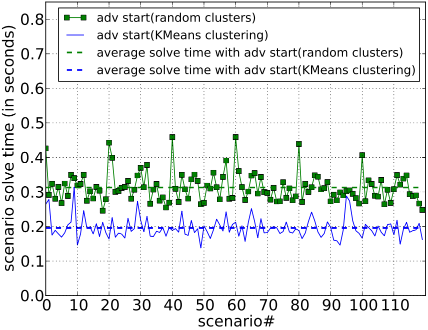


Algorithm 1 The Scenario Clustering Algorithm

Input
D_i- Demand set for scenario i ($i = 1, 2, ..., n$)
k - number of clusters
Output
k equally sized clusters of scenarios
Algorithm
{label, centroids} = kMeans({D_1, D_2, D_3, ..., D_n}, k)
IdealClusterSize = $\frac{n}{k}$
$size_i$ = size of cluster i
{Identify Oversized clusters}
$\mathcal{O} = \{c \in Clusters \mid size_c > IdealClusterSize\}$
{Identify Undersized clusters}
$\mathcal{U} = \{c \in Clusters \mid size_c < IdealClusterSize\}$
$\mathcal{S}$: set of adjustable points
for $c \in \mathcal{O}$ **do**
  Find ($size_i - IdealClusterSize$) points in cluster c that are farthest from $centroid_c$ and add them to the set $\mathcal{S}$
end for
while $size(\mathcal{S}) > 0$ **do**
  Find the closest pair of cluster $c \in (U)$ and point $p \in \mathcal{S}$
  Add p to cluster c
  Remove p from $\mathcal{S}$
  if $size_c == IdealClusterSize$ **then**
    Remove c from $\mathcal{U}$
  end if
end while

Optimizing Stage 2

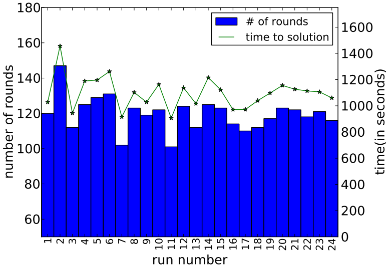
Scenario Clustering Performance



33% reduction in scenario solve times (10 time period model)

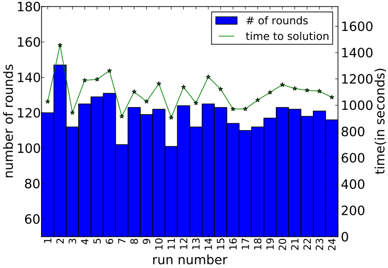
Results

A Note: Variation Across Identical Runs



Results

A Note: Variation Across Identical Runs

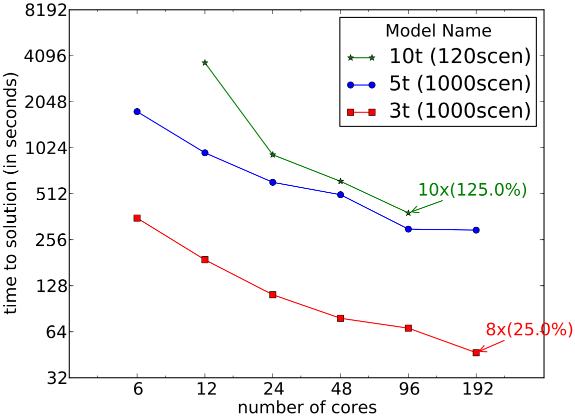


Variability across identical runs

- scenario assignment upon work requests
- variable message latencies, LP solve times
- simplex starts from previous basis
- identical scenario evaluations yields different cuts

Results

Scalability



Future Work

- *Clustering*
  - based on critical missions
- *Scenario Based Decomposition*
  - Solve with subset of scenarios in parallel
  - combine cuts and solve with full set of scenarios
- *Lagrangean Decomposition*
  - stage 1 bottleneck
  - decompose using lagrangean relaxation

Thank You!

Performance Optimization of a Parallel, Two Stage Stochastic Linear Program: The Military Aircraft Allocation Problem

Akhil Langer, Ramprasad Venkataraman, Udatta Palekar, Laxmikant Kale, *Steven Baker

Parallel Programming Laboratory
University of Illinois

*MITRE Corp

ICPADS 2012