Solving Multistage Stochastic Linear Programs on the Computational Grid

JERRY SHEN
JEFF LINDEROTH

Lehigh University
Industrial & System Engineering
Oct 26, 2006
Overview

- Multi-stage stochastic linear programs (MSLP) are **difficult**.
  - They are cast as large-scale optimization problems.
  - There is no viable software tools for solving large-scale MSLP instances.

- Grid is a very **powerful** computational platform but needs to be used wisely.

- This research focus on implementing parallel nested decomposition algorithm on a computational Grid.
  - We developed an MSLP solver **MW-AND** based on a nested-decomposition (ND) algorithm,
  - We discuss the challenges and the approaches.
Outline

- Preliminaries
  - Multi-stage Stochastic Linear Program
  - Nested Decomposition Algorithm
  - Grid Computing
- Challenges and Approaches
  - CDF Framework
  - Asynchronicity
  - Sequencing
  - Cut Management
Multi-stage Stochastic Linear Program (MSLP)

We make decisions everyday under uncertainty; not all at the same time.

\[ \xi_1 x_1 + \xi_2 x_2 + \xi_3 x_T - \xi_T x_T \]

Multi-stage Stochastic Programming: How to make a good decision \((x_1)\) now by taking into account all future uncertainty?
MSLP

We make decisions everyday
MSLP

We make decisions everyday

- Under uncertainty;
- Not all at the same time.
We make decisions everyday

- Under uncertainty;
- Not all at the same time.

Multi-stage Stochastic Programming
We make decisions everyday

- Under uncertainty;
- Not all at the same time.

Multi-stage Stochastic Programming

How to make a good decision ($x_1$) now by taking into account all future uncertainty?
Multi-stage Scenario Tree

- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node $n$ in the tree
- $S(n)$: Set of successor nodes of $n$
- $\hat{p}_{nm}$: Conditional probability that the random events leading from node $n$ to node $m$ occurs
Multi-stage Scenario Tree

- $N$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node $n$ in the tree
- $S(n)$: Set of successor nodes of $n$
- $\hat{p}_{nm}$: Conditional probability that the random events leading from node $n$ to node $m$ occurs
- $x_n$: Decision taken at node $n$
Multi-stage Scenario Tree

- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node $n$ in the tree
- $S(n)$: Set of successor nodes of $n$
- $\hat{p}_{nm}$: Conditional probability that the random events leading from node $n$ to node $m$ occurs
- $x_n$: Decision taken at node $n$
- $Q_n(\cdot)$: Recourse function at node $n$
Multi-stage Scenario Tree

- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node $n$ in the tree
- $S(n)$: Set of successor nodes of $n$
- $\hat{p}_{nm}$: Conditional probability that the random events leading from node $n$ to node $m$ occurs
- $x_n$: Decision taken at node $n$
- $Q_n(\cdot)$: Recourse function at node $n$
- $Q_n(x_n) = \sum_{m \in S(n)} \hat{p}_{nm} Q_m(x_n)$: Expected Recourse function at node $n$
Multi-stage Stochastic Linear Program

Recursive Model

\[
\begin{align*}
\min & \quad c_1^T x_1 + Q_1(x_1) \\
\text{s.t.} & \quad W_1 x_1 = h_1, \\
& \quad x_1 \geq 0,
\end{align*}
\]

where

\[
Q_n(x_n) \overset{\text{def}}{=} \sum_{m \in S(n)} \hat{p}_{nm} Q_m(x_n), \quad \forall n \in N,
\]

and

\[
Q_n(x_{\rho(n)}) \overset{\text{def}}{=} \min_{x_n \geq 0} \left\{ c_n^T x_n + Q_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\},
\quad \forall n \in N \setminus \{1\}.
\]
Multi-stage Stochastic Linear Program

Recursive Model

\[\begin{align*}
\min & \quad c_1^T x_1 + Q_1(x_1) \\
\text{s.t.} & \quad W_1 x_1 = h_1, \\
& \quad x_1 \geq 0,
\end{align*}\]

where (Implicit)

\[Q_n(x_n) \overset{\text{def}}{=} \sum_{m \in S(n)} \hat{p}_{nm} Q_m(x_n), \quad \forall n \in \mathcal{N},\]

and (Recursive)

\[Q_n(x_{\rho(n)}) \overset{\text{def}}{=} \min_{x_n \geq 0} \left\{ c_n^T x_n + Q_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\}, \quad \forall n \in \mathcal{N} \setminus \{1\}.\]

- Bad news: \(Q_1(\cdot)\) is extremely difficult to evaluate;

Jerry Shen (Lehigh University)
Multi-stage Stochastic Linear Program

Recursive Model

\[
\begin{align*}
\min & \quad c^T_1 x_1 + Q_1(x_1) \\
\text{s.t.} & \quad W_1 x_1 = h_1, \\
& \quad x_1 \geq 0,
\end{align*}
\]

where (Implicit)

\[
Q_n(x_n) \overset{\text{def}}{=} \sum_{m \in S(n)} \hat{p}_{nm} Q_m(x_n), \quad \forall n \in \mathcal{N},
\]

and (Recursive)

\[
Q_n(x_{\rho(n)}) \overset{\text{def}}{=} \min_{x_n \geq 0} \left\{ c^T_n x_n + Q_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\}, \quad \forall n \in \mathcal{N} \setminus \{1\}.
\]

- **Bad news:** \( Q_1(\cdot) \) is extremely difficult to evaluate;
- **Good news:** Evaluation of \( Q_n(\cdot) \) can be broken down into smaller function evaluation \( Q_n(\cdot) \).
Multi-stage Stochastic Linear Program

Recursive Model

\[
\begin{align*}
\min & \quad c_1^T x_1 + Q_1(x_1) \\
\text{s.t.} & \quad W_1 x_1 = h_1, \\
& \quad x_1 \geq 0,
\end{align*}
\]

where (Implicit)

\[
Q_n(x_n) \overset{\text{def}}{=} \sum_{m \in S(n)} \hat{p}_{nm} Q_m(x_n), \quad \forall n \in \mathcal{N},
\]

and (Recursive)

\[
Q_n(x_{\rho(n)}) \overset{\text{def}}{=} \min_{x_n \geq 0} \left\{ c_n^T x_n + Q_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\},
\]

\[
\forall n \in \mathcal{N} \setminus \{1\}.
\]

- **Bad news:** $Q_1(\cdot)$ is extremely difficult to evaluate;
- **Good news:** Evaluation of $Q_n(\cdot)$ can be broken down into smaller function evaluation $Q_n(\cdot)$.
- **Better news:** $Q_n(\cdot)$ is convex function. (So is $Q_n(\cdot)$)
Evaluation of $Q_n(\cdot)$
Evaluation of $Q_n(\cdot)$
Evaluation of $Q_n(\cdot)$
Evaluation of $Q_n(\cdot)$
Evaluation of $Q_n(\cdot)$

\[ Q_n(\cdot) \]

\[ M^k_n(\cdot) \]
Evaluation of $Q_n(\cdot)$
Evaluation of $Q_n(\cdot)$

![Graph](image-url)
Evaluation of $Q_n(\cdot)$

Inexact evaluation: $Q_n(\cdot) > M_k^n(\cdot)$
Evaluation of $Q_n(\cdot)$

Exact evaluation: $Q_n(\cdot) = M^k_n(\cdot)$
Nested Decomposition Algorithm

A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)

Natural to parallelize.

Synchronously

Asynchronously

Jerry Shen (Lehigh University)
Nested Decomposition Algorithm

A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)

Natural to parallelize.

Policy
Nested Decomposition Algorithm

- Policy
- Feasibility Cuts
- Optimality Cuts

A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)

Natural to parallelize.
Nested Decomposition Algorithm

- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.) Natural to parallelize.
A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)
Nested Decomposition Algorithm

- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

- A lot of freedom when choosing the directions. (FFF, FF, FB, etc.)
Nested Decomposition Algorithm

- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)
Nested Decomposition Algorithm

- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

- A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)
A lot of freedom when choosing the directions. (FFFFB, FF, FB, etc.)
Nested Decomposition Algorithm

- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering
- New Iteration

A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)
Natural to parallelize.
Nested Decomposition Algorithm

- A lot of freedom when choosing the directions. (FFF, FF, FB, etc.)
- Natural to parallelize.
  - Synchronously

- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering
- New Iteration
Nested Decomposition Algorithm

- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering
- New Iteration

- A lot of freedom when choosing the directions. (FFFB, FF, FB, etc.)
- Natural to parallelize.
  - Synchronously
  - Asynchronously
Is Parallel Enough?

How large is the problem?
100 children at each node
6 stages
→ Last stage scenarios = 10

How many computers do we need?
As many as possible. Even yours
Answer: Grid Computing
Is Parallel Enough?

How large is the problem?

- 100 children at each node → Last stage scenarios = $10^{10}$
- 6 stages
Is Parallel Enough?

How large is the problem?

- 100 children at each node
- 6 stages

$\rightarrow$ Last stage scenarios $= 10^{10}$

How many computers do we need?

- As many as possible. Even yours
Is Parallel Enough?

How large is the problem?
- 100 children at each node
- 6 stages
  \[ \rightarrow \text{Last stage scenarios} = 10^{10} \]

How many computers do we need?
- As many as possible. Even yours when you are at INFORMS.
Is Parallel Enough?

How large is the problem?

- 100 children at each node
- 6 stages

→ Last stage scenarios = $10^{10}$

How many computers do we need?

- As many as possible. Even yours when you are at INFORMS.

Answer:
Is Parallel Enough?

**How large is the problem?**

- 100 children at each node
- 6 stages

→ Last stage scenarios = 10^{10}

**How many computers do we need?**

- As many as possible. Even yours when you are at INFORMS.

---

Answer: Grid Computing
Grid Computing

Tools

- Condor (http://www.cs.wisc.edu/condor)
  - User need not have an account or access to the machines
  - Machine owner specifies conditions under which jobs are allowed to run
  - Condor use matchmaking to schedule jobs among the pool
  - Jobs can be check-pointed and migrated

MW (http://www.cs.wisc.edu/condor/MW)
- Master assigns tasks to the workers
- Workers execute tasks and report results to the master
- Workers need not to communicate with each other
- Simple and Fault-Tolerant

A set of C++ abstract base classes
Grid Computing

Tools

- **Condor** (http://www.cs.wisc.edu/condor)
  - User need not have an account or access to the machines
  - Machine owner specifies conditions under which jobs are allowed to run
  - Condor use matchmaking to schedule jobs among the pool
  - Jobs can be check-pointed and migrated

- **MW** (http://www.cs.wisc.edu/condor/MW)
  - Master assigns tasks to the workers
  - Workers execute tasks and report results to the master
  - Workers need not to communicate with each other
  - **Simple and Fault-Tolerant**
  - A set of C++ abstract base classes
We want

A solver for large-scale MSLP instances
We want

A solver for large-scale MSLP instances

1. Correctness
   - To ensure algorithm termination and convergence.

2. Flexibility
   - To easily allow testing different sequencing mechanisms.
   - To allow different aggregations and/or buffering of nodes and model functions.

3. Efficiency
   - To allow acting in asynchronous manner.
We want

A solver for large-scale MSLP instances

1. Correctness
   - To ensure algorithm termination and convergence.

2. Flexibility
   - To easily allow testing different sequencing mechanisms.
   - To allow different aggregations and/or buffering of nodes and model functions.

3. Efficiency
   - To allow acting in asynchronous manner.

MW-AND with CDF
CDF Framework – Node Status

- Iteration Counter $k_n$
- Child Counter $\phi_{k_n}^n$
- Cut Counter $\psi_{k_n}^n$
- CDF Status: $ST_n = (\text{COLOR}, \text{DIRECTION}, \text{FLAG})$

**COLOR**
- Red: Node has finished computation.
- Yellow: Node is ready for computation.
- Green: Node is under process.

**DIRECTION**
- $\rightarrow$ Forward: Forward job is under process or information will be passed from parent
- $\leftarrow$ Backward: Backward job is under process or information will be passed from children

**FLAG**
- $\star$ Star: Exact evaluation ($M_{n}^{k}(\cdot) = Q_{n}(x_{n}^{k})$)
- $\emptyset$ Null: Inexact evaluation ($M_{n}^{k}(\cdot) < Q_{n}(x_{n}^{k})$)
CDF Framework – Trigger Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Destination</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>$\rho(n) \rightarrow n$</td>
<td>Start to evaluate $Q_{\rho(n)}(\cdot)$ under policy $x^{k\rho(n)}_{\rho(n)}$</td>
</tr>
<tr>
<td>Update</td>
<td>$\rho(n) \rightarrow n$</td>
<td>Update model $M_{\rho(n)}(\cdot)$ given policy $x^{k\rho(n)}_{\rho(n)}$</td>
</tr>
<tr>
<td>Restart</td>
<td>$n \rightarrow \rho(n)$</td>
<td>find a new policy $x^{k\rho(n)}_{\rho(n)}$</td>
</tr>
<tr>
<td>Done</td>
<td>$n \rightarrow \rho(n)$</td>
<td>new model updated, but $M_{\rho(n)}(\cdot) &lt; Q_{\rho(n)}(\cdot)$</td>
</tr>
<tr>
<td>End</td>
<td>$n \rightarrow \rho(n)$</td>
<td>new model updated, and $M_{\rho(n)}(\cdot) = Q_{\rho(n)}(\cdot)$</td>
</tr>
<tr>
<td>Terminate</td>
<td>$n \rightarrow Siblings$</td>
<td>Do not evaluate $Q_{\rho(n)}(\cdot)$ under policy $x^{k\rho(n)}_{\rho(n)}$</td>
</tr>
<tr>
<td>Go</td>
<td>$n \rightarrow Siblings$</td>
<td>Join the task and go to the Grid</td>
</tr>
</tbody>
</table>

Table: Type of Signals.
Challenge (Synchronicity is BAD in the Grid!)
Asynchronicity

Challenge: What is a proper level of asynchronicity?
Asynchronicity

Challenge: What is a proper level of asynchronicity?

Asynchronicity Level

- **High:**
  - High utilization of the resources
  - Less accurate recourse function evaluation at each iteration
  - More iterations required

- **Low:**
  - More accurate recourse function evaluation at each iteration
  - Lower overall parallel performance

Approach: Dynamic asynchronicity level

- Stage-dependent (test the impact of asynchronicity level to different stages)
- Resource-dependent (enable more accurate evaluation when resources are limited)
## Asynchronicity

### Challenge: What is a proper level of asynchronicity?

<table>
<thead>
<tr>
<th>Asynchronicity Level</th>
<th>High:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High utilization of the resources</td>
</tr>
<tr>
<td></td>
<td>Less accurate recourse function evaluation at each iteration</td>
</tr>
<tr>
<td></td>
<td>More iterations required</td>
</tr>
<tr>
<td>Low:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>More accurate recourse function evaluation at each iteration</td>
</tr>
<tr>
<td></td>
<td>Lower overall parallel performance</td>
</tr>
</tbody>
</table>

### Approach: Dynamic asynchronicity level
- **Stage-dependent** (test the impact of asynchronicity level to different stages)
- **Resource-dependent** (enable more accurate evaluation when resources are limited)
Asynchronicity is a must
Sequencing Mechanism
Sequencing Mechanism
Sequencing Mechanism

Challenge: To ensure non-blocking behavior of the algorithm

Sequencing Method
- Algorithm may be blocking even though the asynchronicity level is set to high.
- More flexibility is preferred.
Sequencing Mechanism

Challenge: To ensure non-blocking behavior of the algorithm

Sequencing Method

- Algorithm may be blocking even though the asynchronicity level is set to high.
- More flexibility is preferred.

Approach: Dynamic double layer sequencing protocol

- First layer: main iteration, suggest FFFB
- Second layer: fine tune, (whenever resource is available)
Data Management

Challenge: To handle the massive amounts of cuts that the algorithm generated

- Large amount of data – Cuts
- Required memory to store the cuts may be huge
  - For example: 27,000 nodes in period T − 1, each node has 20 cuts, \( x_n \in \mathbb{R}^{100} \), requires \( \geq 400\text{MB} \) to store cuts.
Data Management

Challenge: To handle the massive amounts of cuts that the algorithm generated

Large amount of data – Cuts

- Required memory to store the cuts may be huge
- For example: 27,000 nodes in period \( T - 1 \), each node has 20 cuts, \( x_n \in \mathbb{R}^{100} \), requires \( \geq 400\)MB to store cuts.
Data Management

Challenge: To handle the massive amounts of cuts that the algorithm generated

Large amount of data – Cuts

- We can not store cuts on the workers as we do not have control over workers, and do not know when the worker will be leaving;
- Master memorizes all the cuts, and will be very busy handling these cuts as the number increases.
- We must do our best to compress or reduce the amount of data.
Data Management

Challenge: To handle the massive amounts of cuts that the algorithm generated.

Large amount of data – Cuts
- We can not store cuts on the workers as we do not have control over workers, and do not know when the worker will be leaving;
- Master memorizes all the cuts, and will be very busy handling these cuts as the number increases.
- We must do our best to compress or reduce the amount of data.

Approach: Cut Management
- Cut Hashing: To quickly sort and locate identical cuts
- Cut Sharing: To allow information sharing among nodes;
- Cut Purging: To reduce the number of inactive or loose cuts;
- Cut Aggregation: To generate aggregated cuts by clustering the nodes.