Solving Multistage Stochastic Linear Programs on the Computational Grid

Jerry Shen

Lehigh University Industrial & System Engineering

May 11, 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 discuss the challenges and propose the approaches.
- We also study the value of MSLP as we can do comprehensive numerical testings on large-scale MSLP instances.

Outline

- Stochastic Linear Program
  - Linear Program vs. Stochastic Linear Program
  - Multi-stage Stochastic Linear Program
  - Nested Decomposition Algorithm
- Distributed Computing
  - Grid Computing
  - Condor & MW
- CDF Framework
- Research Challenges
- Value of Multistage Stochastic Linear Program
- Future Research

Outline

- Stochastic Linear Program
  - Linear Program vs. Stochastic Linear Program
  - Multi-stage Stochastic Linear Program
  - Nested Decomposition Algorithm
- Distributed Computing
  - Grid Computing
  - Condor & MW
- CDF Framework
- Research Challenges
- Value of Multistage Stochastic Linear Program
- Future Research

Linear Program v.s Stochastic Linear Program

Linear Program

$$\begin{array}{lrcl} \text{min} & c^T x & & \\ \text{s.t.} & Ax & = & b, \\ & x & \geq & 0, \end{array}$$

$$c \in \Re^{\mathrm{n}}, A \in \Re^{\mathrm{m \times n}}, b \in \Re^{\mathrm{m}}.$$

Linear Program v.s Stochastic Linear Program

Linear Program

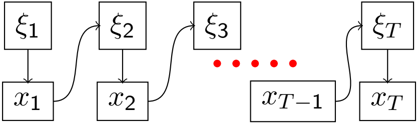
$$\begin{array}{lrcl} \text{min} & c^T x & & \\ \text{s.t.} & Ax & = & b, \\ & x & \geq & 0, \end{array}$$

$$c \in \Re^{\mathsf{n}}, A \in \Re^{\mathsf{m \times n}}, b \in \Re^{\mathsf{m}}.$$

- All functions are linear.
- Does not consider uncertainty in the model.
- Easy.

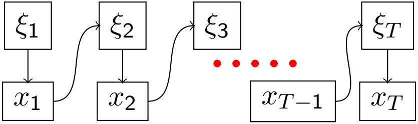
Linear Program v.s Stochastic Linear Program

- Not all decisions are made at the same time.
- There is uncertainty!



Linear Program v.s Stochastic Linear Program

- Not all decisions are made at the same time.
- There is uncertainty!



How to make a good decision (x_1) now by taking into account all future uncertainty?

Linear Program vs Stochastic Linear Program

Stochastic Linear Program

$$\begin{array}{lrcl} \text{min} & c_1^T x_1 & + & \mathcal{Q}_1(x_1) \\ \text{s.t.} & A_1 x_1 & = & b_1, \\ & x_1 & \geq & 0, \end{array}$$

$$c_1 \in \Re^{n}, A_1 \in \Re^{m \times n}, b_1 \in \Re^{m}.$$

where $\mathcal{Q}_1(x_1)$ is the expected recourse function.

Linear Program vs Stochastic Linear Program

Stochastic Linear Program

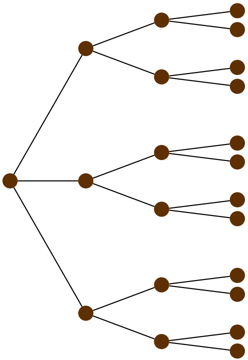
$$\begin{array}{lrcl} \text{min} & c_1^T x_1 & + & \mathcal{Q}_1(x_1) \\ \text{s.t.} & A_1 x_1 & = & b_1, \\ & x_1 & \geq & 0, \end{array}$$

$$c_1 \in \Re^{\mathsf{n}}, A_1 \in \Re^{\mathsf{m \times n}}, b_1 \in \Re^{\mathsf{m}}.$$

where $\mathcal{Q}_1(x_1)$ is the expected recourse function.

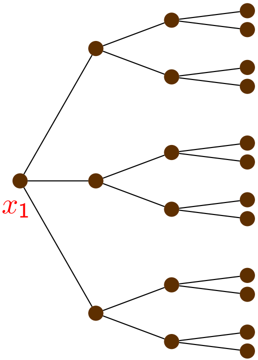
- $\mathcal{Q}_1(x_1)$ measures the cost of making corrective actions on your initial decision x_1, after a random events have taken place.

Multi-stage Scenario Tree



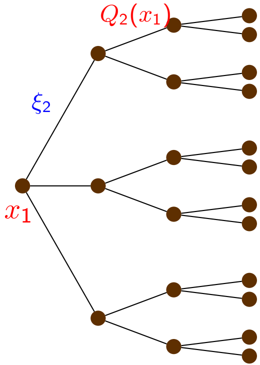
- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_n$: Conditional probability that the sequence of events leading to node n occurs

Multi-stage Scenario Tree



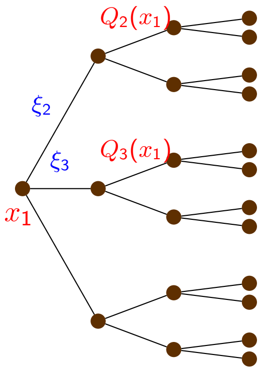
- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_n$: Conditional probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n

Multi-stage Scenario Tree



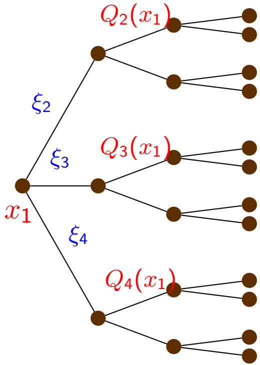
- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_n$: Conditional probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n
- $Q_n(\cdot)$: Recourse function at node n

Multi-stage Scenario Tree



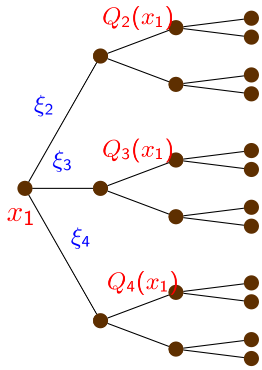
- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_n$: Conditional probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n
- $Q_n(\cdot)$: Recourse function 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
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_n$: Conditional probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n
- $Q_n(\cdot)$: Recourse function at node n
- $\mathcal{Q}_n(x_n) = \sum_{m \in \mathcal{S}(n)} \hat{p}_m Q_m(x_n)$: Expected Recourse function at node n

Multi-stage Scenario Tree



$\mathcal{Q}_1(x_1) = \sum_{m=2}^{4} \hat{p}_m Q_m(x_1)$

- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_n$: Conditional probability that the sequence of events leading to node n occurs
- x_n: Decision taken at node n
- $Q_n(\cdot)$: Recourse function at node n
- $\mathcal{Q}_n(x_n) = \sum_{m \in \mathcal{S}(n)} \hat{p}_m Q_m(x_n)$: Expected Recourse function at node n

Multi-stage Stochastic Linear Program

Recursive Model

$$\begin{array}{rrcl} \mathbf{Z} = \min & c_1^T x_1 & + & \mathcal{Q}_1(x_1) \\ \text{s.t.} & W_1 x_1 & = & h_1, \\ & x_1 & \geq & 0, \end{array}$$

where

$$\mathcal{Q}_n(x_n) \stackrel{\text{def}}{=} \sum_{\tilde{n} \in \mathcal{S}(n)} \hat{p}_{\tilde{n}} Q_{\tilde{n}}(x_n), \quad \forall n \in \mathcal{N},$$

and

$$Q_n(x_{\rho(n)}) \stackrel{\text{def}}{=} \min_{x_n \geq 0} \left\{ c_n^T x_n + \mathcal{Q}_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\}, \qquad \forall n \in \mathcal{N} \setminus \{1\}.$$

Multi-stage Stochastic Linear Program

Recursive Model

$$\begin{array}{rrcl} \mathbf{Z} = \min & c_1^T x_1 & + & \mathcal{Q}_1(x_1) \\ \text{s.t.} & W_1 x_1 & = & h_1, \\ & x_1 & \geq & 0, \end{array}$$

where

$$\mathcal{Q}_n(x_n) \stackrel{\text{def}}{=} \sum_{\tilde{n} \in \mathcal{S}(n)} \hat{p}_{\tilde{n}} Q_{\tilde{n}}(x_n), \quad \forall n \in \mathcal{N},$$

and

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

- Bad news: $\mathcal{Q}_n(\cdot)$ is extremely difficult to evaluate;

Multi-stage Stochastic Linear Program

Recursive Model

$$\begin{array}{rrcl} \mathbf{Z} = \min & c_1^T x_1 & + & \mathcal{Q}_1(x_1) \\ \text{s.t.} & W_1 x_1 & = & h_1, \\ & x_1 & \geq & 0, \end{array}$$

where

$$\mathcal{Q}_n(x_n) \stackrel{\text{def}}{=} \sum_{\tilde{n} \in \mathcal{S}(n)} \hat{p}_{\tilde{n}} Q_{\tilde{n}}(x_n), \quad \forall n \in \mathcal{N},$$

and

$$Q_n(x_{\rho(n)}) \stackrel{\text{def}}{=} \min_{x_n \geq 0} \left\{ c_n^T x_n + \mathcal{Q}_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\}, \qquad \forall n \in \mathcal{N} \setminus \{1\}.$$

- Bad news: $\mathcal{Q}_n(\cdot)$ is extremely difficult to evaluate;
- Good news: Evaluation of $\mathcal{Q}_n(\cdot)$ can be broken down into smaller function evaluation $Q_n(\cdot)$.

Multi-stage Stochastic Linear Program

Recursive Model

$$\begin{array}{rrcl} \mathbf{Z} = \min & c_1^T x_1 & + & \mathcal{Q}_1(x_1) \\ \text{s.t.} & W_1 x_1 & = & h_1, \\ & x_1 & \geq & 0, \end{array}$$

where

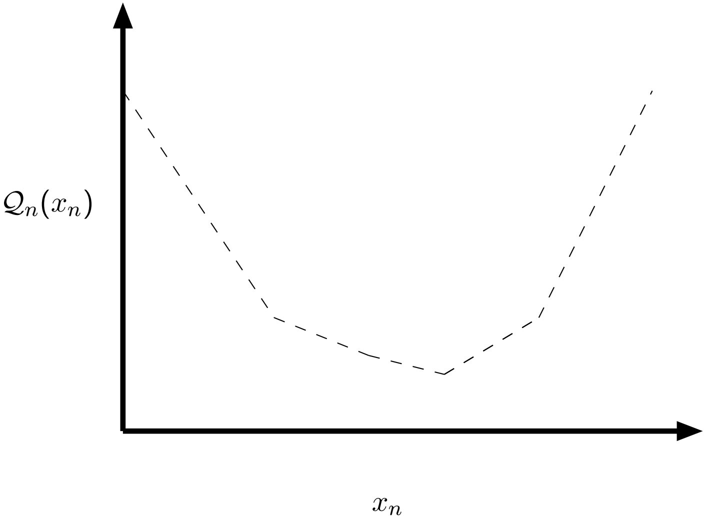
$$\mathcal{Q}_n(x_n) \stackrel{\text{def}}{=} \sum_{\tilde{n} \in \mathcal{S}(n)} \hat{p}_{\tilde{n}} Q_{\tilde{n}}(x_n), \quad \forall n \in \mathcal{N},$$

and

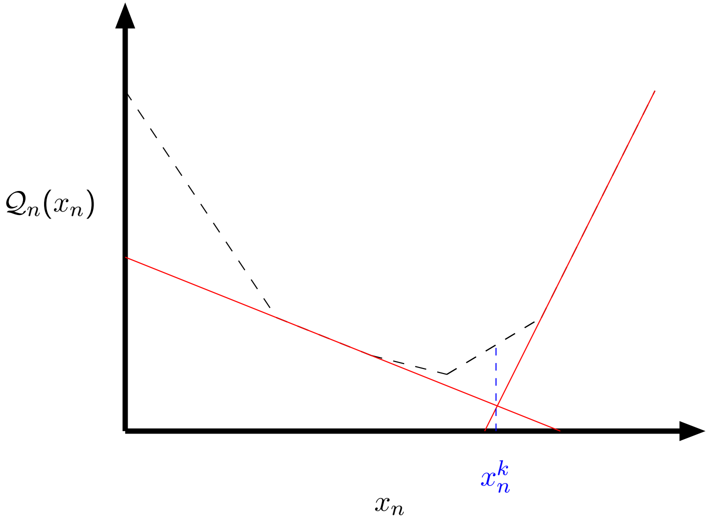
$$Q_n(x_{\rho(n)}) \stackrel{\text{def}}{=} \min_{x_n \geq 0} \left\{ c_n^T x_n + \mathcal{Q}_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\}, \quad \forall n \in \mathcal{N} \setminus \{1\}.$$

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

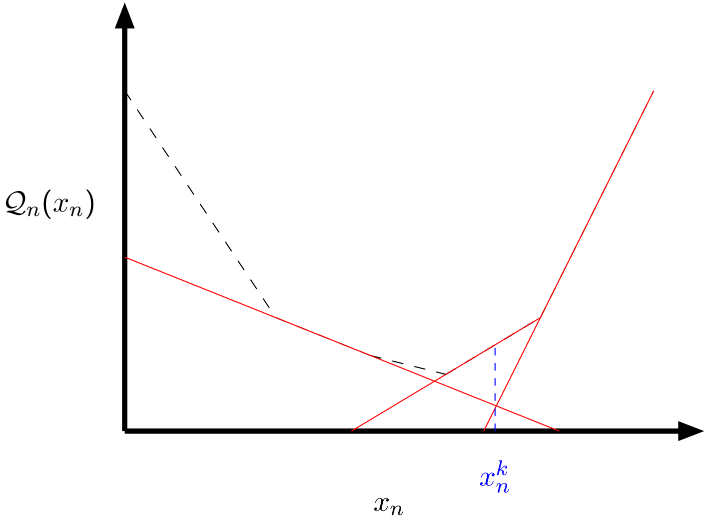
Evaluation of $\mathcal{Q}_n(x_n)$



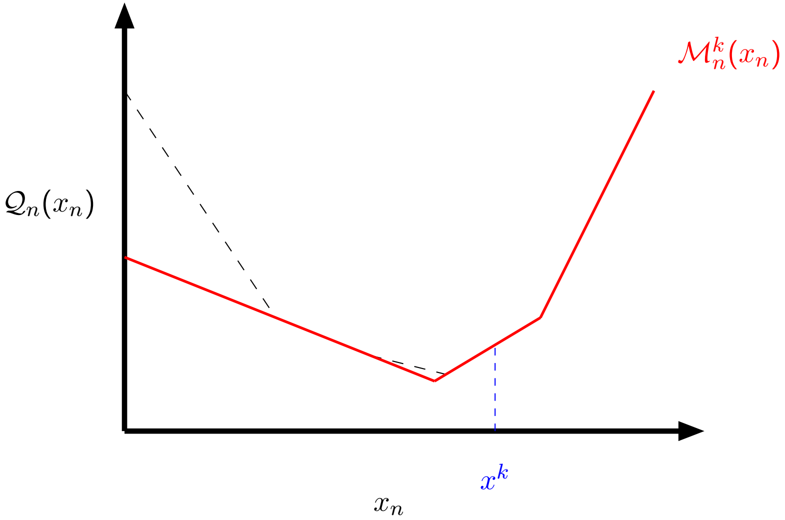
Evaluation of $\mathcal{Q}_n(x_n)$



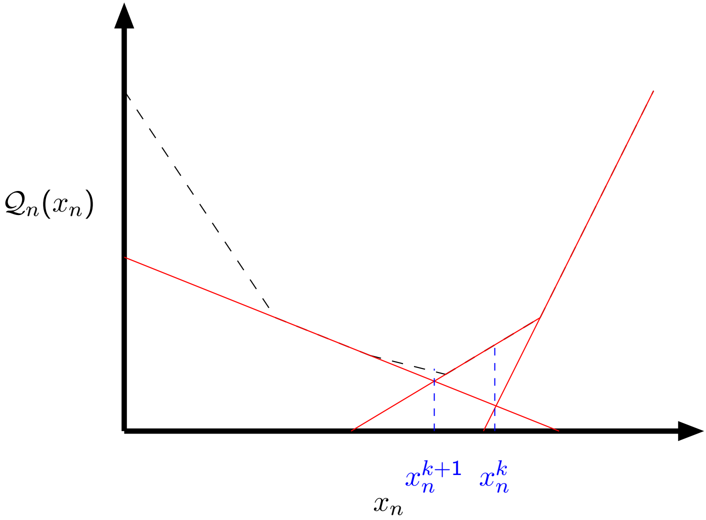
Evaluation of $\mathcal{Q}_n(x_n)$



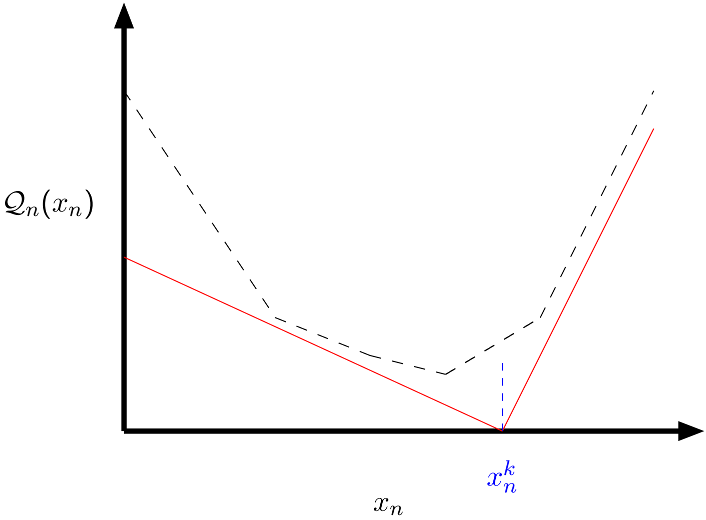
Evaluation of $\mathcal{Q}_n(x_n)$



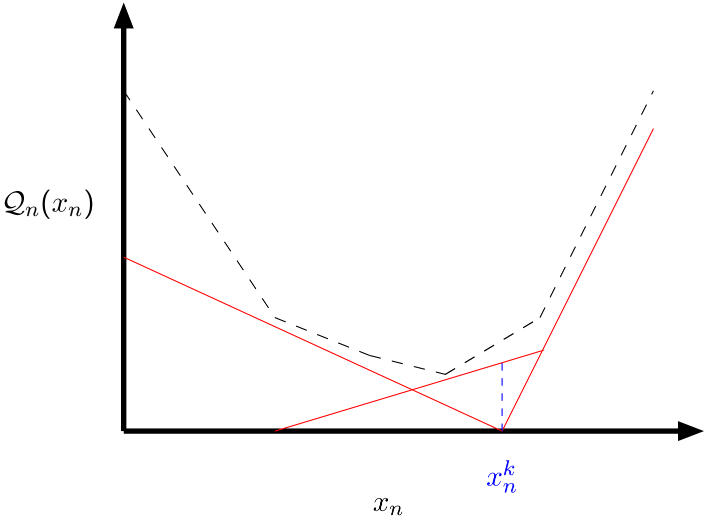
Evaluation of $\mathcal{Q}_n(x_n)$



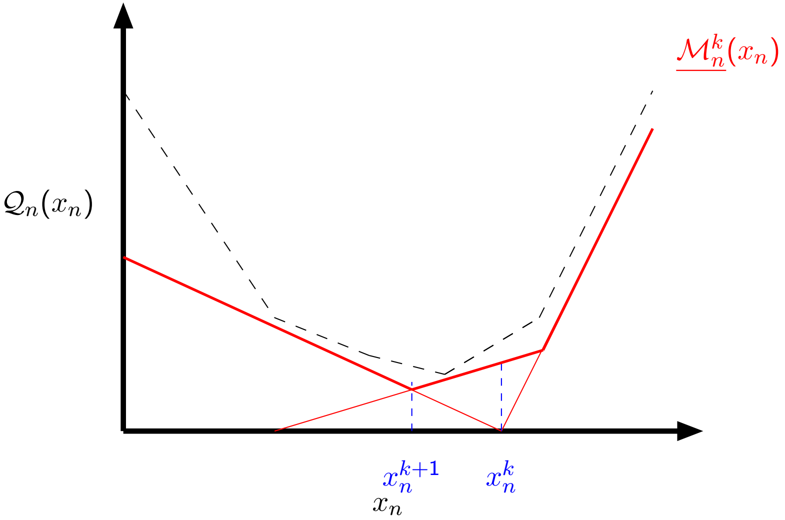
Evaluation of $\mathcal{Q}_n(x_n)$



Evaluation of $\mathcal{Q}_n(x_n)$



Evaluation of $\mathcal{Q}_n(x_n)$



Recourse Function Properties

Model function – lower bound to the True model

$LP_n(x_{\rho(n)})$:

$$
\begin{aligned}
\mathcal{M}^l_n(x_{\rho(n)}) = \min \quad c_n^T x_n \quad &+ \quad \sum_{j=1}^{C_n} \theta_{n,[j]} \\
\text{s.t.} \quad W_n x_n \quad &= \quad h_n - T_n x_{\rho(n)}, \\
0 \quad &\geq \quad -d x_n + \delta, \ \forall (d,\delta) \in \mathcal{D}^l_n \\
\theta_{n,[j]} \quad &\geq \quad -e x_n + \epsilon, \ \forall (e,\epsilon) \in \mathcal{E}^l_{n,[j]}, \ \forall j \in \{1,\ldots,C_n\} \\
x_n \geq 0 \quad &, \quad \theta_{n,[j]} \geq -M, \quad \forall j \in \{1,\ldots,C_n\}
\end{aligned}
$$

- LP_n is the master linear program to be solved on node n.

Nested Decomposition Algorithm

Main steps

Start from root node $n = 1$:

1. Solve $LP_n(x_{\rho(n)})$. If infeasible, go to step 2, otherwise go to step 3.
2. If $n = 1$, STOP. Otherwise, find feasibility cut. Go to step 4.
3. Decide the direction (According to the sequencing method).
   - If *Forward*, record the primal solution as policy. Add nodes $m \in \mathcal{S}(n)$ into queue. Go to step 4.
   - If go *Backward*, record the simplex multiplier and find optimality cut. Add node $\rho(n)$ into queue and go to step 4.
4. If stopping criteria meet, STOP. Otherwise get the next node from queue and return to step 1.

- A lot of freedom when choosing the directions.

Nested Decomposition Algorithm

Sequencing method

- Fast Forward Fast Backward (FFFB)
  Makes complete passes through the tree. Change the direction as little as possible.
- Fast Forward (FF)
  Computational work are focus at the latter stages. Always find the best model function.
- Fast Backward (FB)
  Choose to go to the stage that requires the least amount of work. Try to find the best policy before function evaluation.
- Hybrid Methods

Outline

- Stochastic Linear Program
  - Linear Program vs. Stochastic Linear Program
  - Multi-stage Stochastic Linear Program
  - Nested Decomposition Algorithm
- Distributed Computing
  - Grid Computing
  - Condor & MW
- CDF Framework
- Research Challenges
- Value of Multistage Stochastic Linear Program
- Future Research

Distributed Computing

Supercomputer

- 1960s: Computer with fast scalar processor.
    - Run mathematical operations on one data element at a time.

Distributed Computing

Supercomputer

- 1960s: Computer with fast scalar processor.
  - Run mathematical operations on one data element at a time.
- 1970s: Computer with vector processor.
  - Run mathematical operations on multiple data elements simultaneously.

Distributed Computing

Supercomputer

- 1960s: Computer with fast scalar processor.
  - Run mathematical operations on one data element at a time.
- 1970s: Computer with vector processor.
  - Run mathematical operations on multiple data elements simultaneously.
- 1980s: Computers with a vector processors work in parallel.
  - Typical numbers of parallel processors only limited to 16 or less.

Distributed Computing

Supercomputer

- 1960s: Computer with fast scalar processor.
  - Run mathematical operations on one data element at a time.
- 1970s: Computer with vector processor.
  - Run mathematical operations on multiple data elements simultaneously.
- 1980s: Computers with a vector processors work in parallel.
  - Typical numbers of parallel processors only limited to 16 or less.
- 1990s: Massive parallel processing systems.
  - Thousands of processors. BlueGene/L (IBM): 131,072 processors (Nov. 2005)

Distributed Computing

Supercomputer

- 1960s: Computer with fast scalar processor.
  - Run mathematical operations on one data element at a time.
- 1970s: Computer with vector processor.
  - Run mathematical operations on multiple data elements simultaneously.
- 1980s: Computers with a vector processors work in parallel.
  - Typical numbers of parallel processors only limited to 16 or less.
- 1990s: Massive parallel processing systems.
  - Thousands of processors. BlueGene/L (IBM): 131,072 processors (Nov. 2005)
- Next: ?

Why do we need a supercomputer?

We want to have a supercomputer that

Why do we need a supercomputer?

We want to have a supercomputer that

- can solve bigger problems (large-scale MSLP)

Why do we need a supercomputer?

We want to have a supercomputer that

- can solve bigger problems (large-scale MSLP)
- can be used over a long time horizon (days/weeks)

Why do we need a supercomputer?

We want to have a supercomputer that

- can solve bigger problems (large-scale MSLP)
- can be used over a long time horizon (days/weeks)

Grid Computing

Grid Computing

Challenges

- Security
  - Who should be allowed to tap in?
- Interfaces
  - How should they tap in?
- Heterogeneous
  - Different hardware, operating systems, software...
- Dynamic
  - You don't know when one machine will leave the pool
  - You don't know when one machine will join the pool
  - Fault-Tolerance is a very important issue when designing the program
- Distributed
  - Machines may be very far apart $\Rightarrow$ slow communication

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

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

Master-Worker (MW) Structure

- MWDriver
  - $get\_userinfo()$
  - $setup\_initial\_tasks()$
  - $pack\_worker\_init\_data()$
  - $act\_on\_completed\_task()$
- MWTask
  - $pack\_work()$
  - $unpack\_work()$
  - $pack\_results()$
  - $unpack\_results()$
- MWWorker
  - $unpack\_init\_data()$
  - $execute\_task()$

Outline

- Stochastic Linear Program
  - Linear Program vs. Stochastic Linear Program
  - Multi-stage Stochastic Linear Program
  - Nested Decomposition Algorithm
- Distributed Computing
  - Grid Computing
  - Condor & MW
- CDF Framework
- Research Challenges
- Value of Multistage Stochastic Linear Program
- Future Research

Goal

Goal

1. Correctly implement parallel nested decomposition algorithm on the Grid
   - Algorithm terminates and converges to the correct solution

Goal

1. Correctly implement parallel nested decomposition algorithm on the Grid
   - Algorithm terminates and converges to the correct solution
2. Support different sequencing mechanisms
   - FFFB, FF, FB, hybrid

Goal

1. Correctly implement parallel nested decomposition algorithm on the Grid
   - Algorithm terminates and converges to the correct solution
2. Support different sequencing mechanisms
   - FFFB, FF, FB, hybrid
3. Allow asynchronous algorithm behaviors
   - Partial evaluation of $\mathcal{Q}_n(\cdot)$

Goal

1. Correctly implement parallel nested decomposition algorithm on the Grid
   - Algorithm terminates and converges to the correct solution
2. Support different sequencing mechanisms
   - FFFB, FF, FB, hybrid
3. Allow asynchronous algorithm behaviors
   - Partial evaluation of $\mathcal{Q}_n(\cdot)$
4. Be efficient with the MW-Grid Framework
   - Buffering, aggregating evaluations into reasonable-sized tasks (computational units)

CDF Framework – Record Node Status

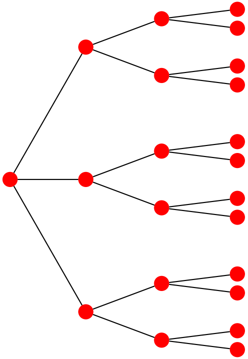
- Iteration Counter ν_n
- Child Counter $\phi_n^{\nu_n}$
- Cut Counter $\psi_n^{\nu_n}$
- CDF Status: ST_n = (COLOR, DIRECTION, FLAG)

COLOR	• Red: Job is completed • Green: Job 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: True model is obtained $(M_n^l(x_{\rho(n)}) = Q_n(x_{\rho(n)}))$ • $\emptyset$ Null: True model is not obtained $(M_n^l(x_{\rho(n)}) < Q_n(x_{\rho(n)}))$

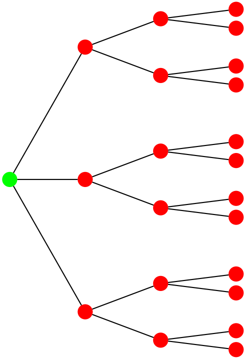
CDF Framework – Trigger Signals

- Start
  - To children: “Here is a policy, start to evaluate your recourse function.”
- Update
  - To children: “Here is a policy, update my model function.”
- Restart
  - To parent: “Give me a new policy.”
- Done
  - To parent: “Model function is not a true model, but we are done.”
- End
  - To parent: “Model function is a true model, we are done.”
- Terminate
  - To siblings: “Terminate, do not evaluate the recourse function any more.”

CDF Framework – Example 1 (FFFB)

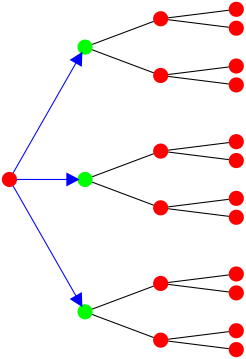


CDF Framework – Example 1 (FFFB)



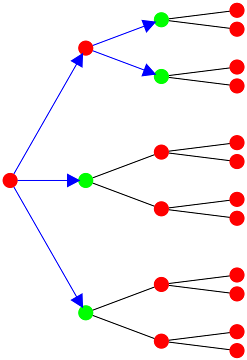
- $n = 1$: Start

CDF Framework – Example 1 (FFFB)



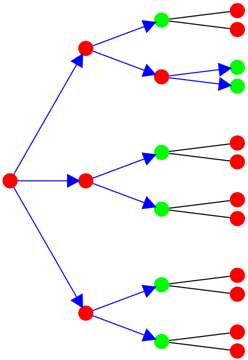
- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start

CDF Framework – Example 1 (FFFB)



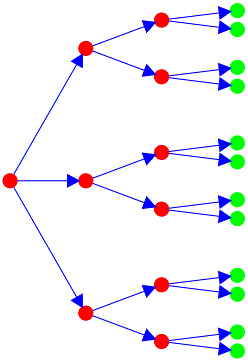
- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start

CDF Framework – Example 1 (FFFB)



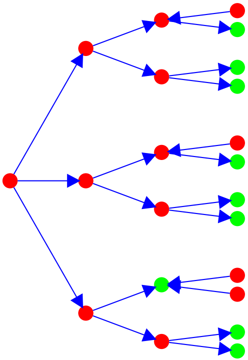
- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(3) \cup \mathcal{S}(4)$: Start
- $n \in \mathcal{S}(6)$: Update

CDF Framework – Example 1 (FFFB)



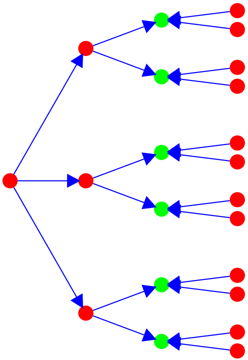
- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(3) \cup \mathcal{S}(4)$: Start
- $n \in \mathcal{S}(6)$: Update
- $\vdots$

CDF Framework – Example 1 (FFFB)



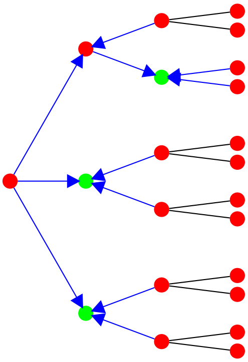
- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(3) \cup \mathcal{S}(4)$: Start
- $n \in \mathcal{S}(6)$: Update
- $\vdots$
- $n = \rho(19)$: Done

CDF Framework – Example 1 (FFFB)



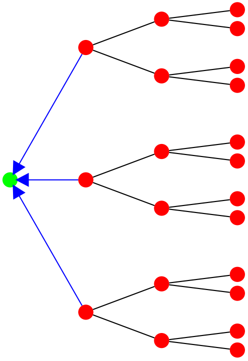
- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(3) \cup \mathcal{S}(4)$: Start
- $n \in \mathcal{S}(6)$: Update
- $\vdots$
- $n = \rho(19)$: Done
- $\vdots$

CDF Framework – Example 1 (FFFB)



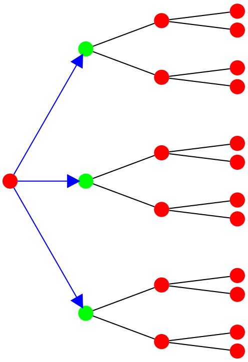
- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(3) \cup \mathcal{S}(4)$: Start
- $n \in \mathcal{S}(6)$: Update
- $\vdots$
- $n = \rho(19)$: Done
- $\vdots$

CDF Framework – Example 1 (FFFB)



- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(3) \cup \mathcal{S}(4)$: Start
- $n \in \mathcal{S}(6)$: Update
- $\vdots$
- $n = \rho(19)$: Done
- $\vdots$
- $n = \rho(2)$: Restart

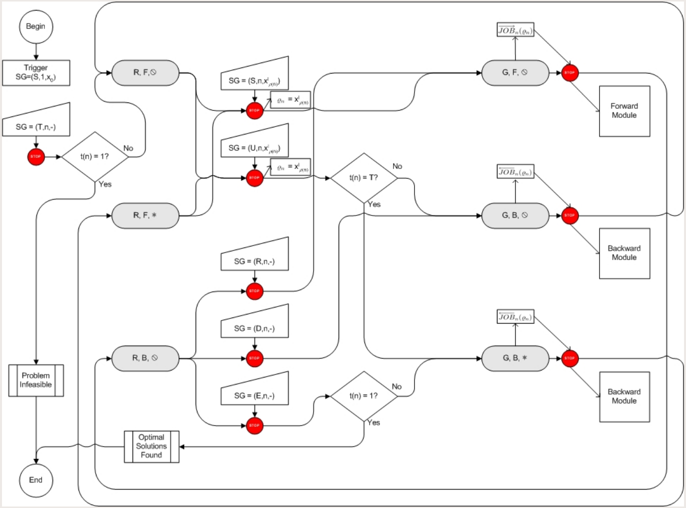
CDF Framework – Example 1 (FFFB)



- $n = 1$: Start
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(3) \cup \mathcal{S}(4)$: Start
- $n \in \mathcal{S}(6)$: Update
- $\vdots$
- $n = \rho(19)$: Done
- $\vdots$
- $n = \rho(2)$: Restart

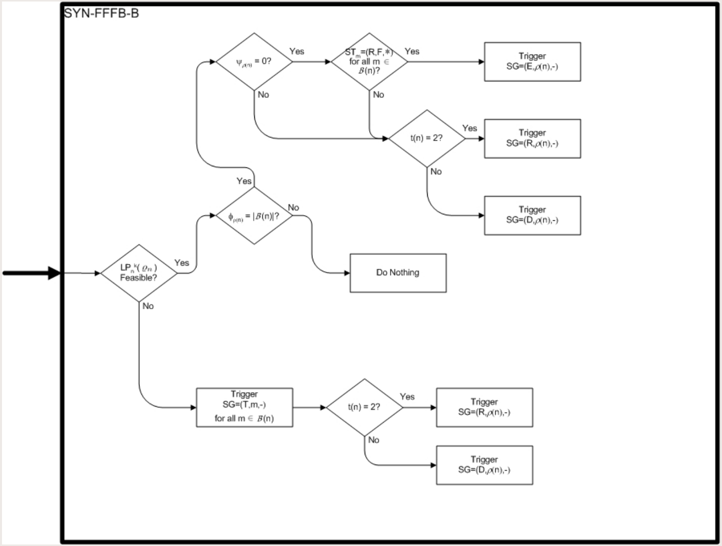
CDF Framework

CDF State Evolving Procedure

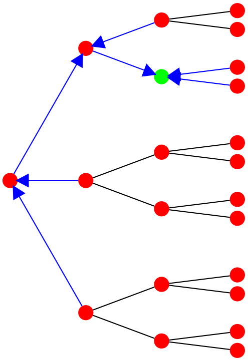


CDF Framework

Signal Triggering Module SYN-FFFB-B



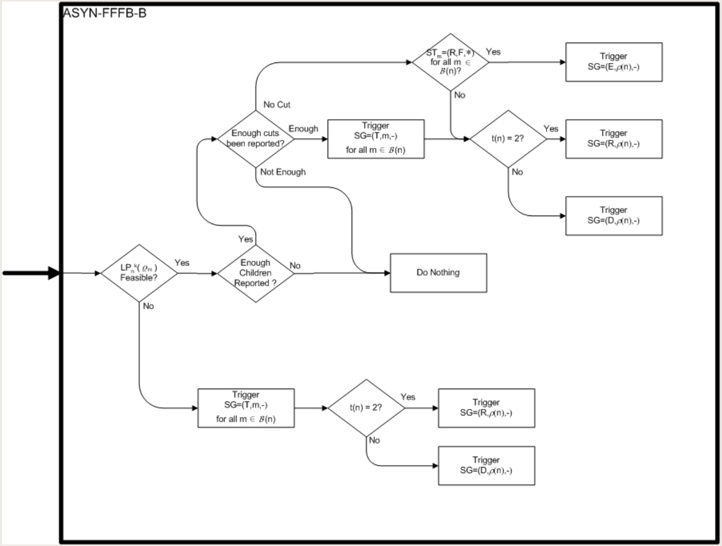
CDF Framework



Synchronicity is
Bad
in the Grid

CDF Framework (Asynchronous)

Signal Triggering Module ASYN-FFFB-B



CDF Framework (Asynchronous)

Asynchronicity Level

- Enough children α_1.
- Enough cuts α_2.

$\phi_{\rho(n)}^{\nu_{\rho(n)}} = \|\mathcal{B}(\rho(n))\|, \quad \psi_{\rho(n)}^{\nu_{\rho(n)}} = 0$	No Cut
$\phi_{\rho(n)}^{\nu_{\rho(n)}} \neq \|\mathcal{B}(\rho(n))\|, \quad \psi_{\rho(n)}^{\nu_{\rho(n)}} = 0$	Not Enough
$\phi_{\rho(n)}^{\nu_{\rho(n)}} = \|\mathcal{B}(\rho(n))\|, \quad 0 < \psi_{\rho(n)}^{\nu_{\rho(n)}} < \alpha_2 \Gamma_{\rho(n)}$	Enough
$\phi_{\rho(n)}^{\nu_{\rho(n)}} \neq \|\mathcal{B}(\rho(n))\|, \quad 0 < \psi_{\rho(n)}^{\nu_{\rho(n)}} < \alpha_2 \Gamma_{\rho(n)}$	Not Enough
$\phi_{\rho(n)}^{\nu_{\rho(n)}} = \|\mathcal{B}(\rho(n))\|, \quad \psi_{\rho(n)}^{\nu_{\rho(n)}} \geq \alpha_2 \Gamma_{\rho(n)}$	Enough
$\phi_{\rho(n)}^{\nu_{\rho(n)}} \neq \|\mathcal{B}(\rho(n))\|, \quad \psi_{\rho(n)}^{\nu_{\rho(n)}} \geq \alpha_2 \Gamma_{\rho(n)}$	Enough

Table: The criteria for enough cutting plain information.

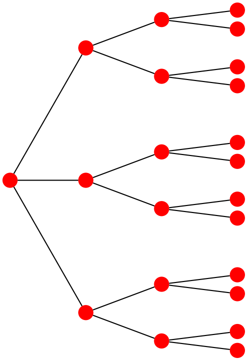
CDF Framework (with buffering)

Purpose: To obtain a reasonable task (computational unit) size.

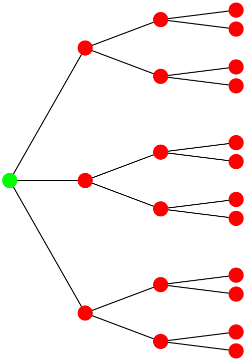
Buffering

- Buffer
  - Temporary storage room for jobs
- new COLOR – Yellow
  - Create jobs on the node, and add the job into the buffer
- new Signal – Go
  - Aggregate the jobs in the buffer and start to execute the task in Grid

CDF Framework – Example 2 (buffer size = 3)

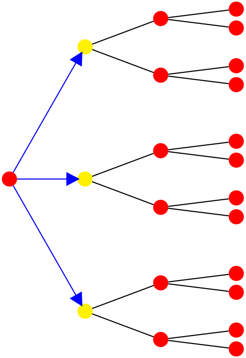


CDF Framework – Example 2 (buffer size = 3)



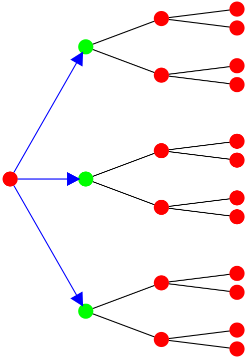
- $n = 1$: Go

CDF Framework – Example 2 (buffer size = 3)



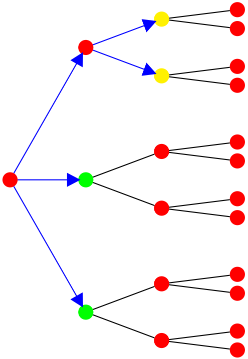
- $n = 1$: Go
- $n \in \mathcal{S}(1)$: Start

CDF Framework – Example 2 (buffer size = 3)



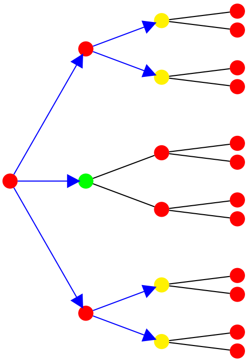
- $n = 1$: Go
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(1)$: Go

CDF Framework – Example 2 (buffer size = 3)



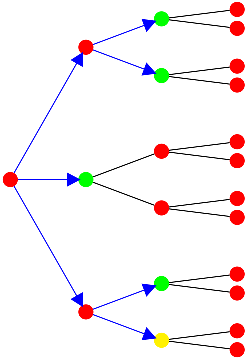
- $n = 1$: Go
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(1)$: Go
- $n \in \mathcal{S}(2)$: Start

CDF Framework – Example 2 (buffer size = 3)



- $n = 1$: Go
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(1)$: Go
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(4)$: Start

CDF Framework – Example 2 (buffer size = 3)



- $n = 1$: Go
- $n \in \mathcal{S}(1)$: Start
- $n \in \mathcal{S}(1)$: Go
- $n \in \mathcal{S}(2)$: Start
- $n \in \mathcal{S}(4)$: Start
- n in buffer 1: Go

Outline

- Stochastic Linear Program
  - Linear Program vs. Stochastic Linear Program
  - Multi-stage Stochastic Linear Program
  - Nested Decomposition Algorithm
- Distributed Computing
  - Grid Computing
  - Condor & MW
- CDF Framework
- Research Challenges
- Value of Multistage Stochastic Linear Program
- Future Research

Research Challenges

- How to ensure effective algorithm performance?
- How to efficiently utilize the Grid resource?
- How to manage the large amount of data?

Algorithmic Challenges

Challenge: To determine the proper level of asynchronicity level

Algorithmic Challenges

Challenge: To determine the proper level of asynchronicity level

Asynchronicity Level

- High level:
  - High utilization of the resources
  - Less accurate recourse function evaluation at each iteration
  - More iterations required
- Low level:
  - More accurate recourse function evaluation at each iteration
  - Lower overall parallel performance

Algorithmic Challenges

Challenge: To determine the proper level of asynchronicity level

Asynchronicity Level

- High level:
  - High utilization of the resources
  - Less accurate recourse function evaluation at each iteration
  - More iterations required
- Low level:
  - More accurate recourse function evaluation at each iteration
  - Lower overall parallel performance

Proposal: Dynamic asynchronicity level

- Stage-dependent (later stage $\Rightarrow$ lower asynchronicity level)
- Iteration-dependent (later iteration $\Rightarrow$ lower asynchronicity level)

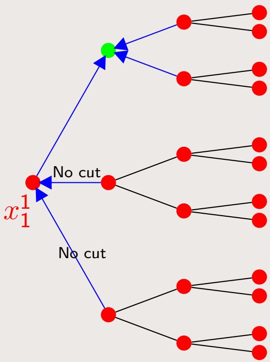
Algorithmic Challenges

Challenge: To ensure non-blocking behavior of the algorithm

Algorithmic Challenges

Challenge: To ensure non-blocking behavior of the algorithm

Sequencing Method

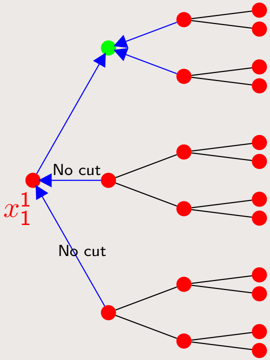


- $ST_3 = (R, F, \emptyset)$
- $ST_4 = (R, F, \emptyset)$

Algorithmic Challenges

Challenge: To ensure non-blocking behavior of the algorithm

Sequencing Method

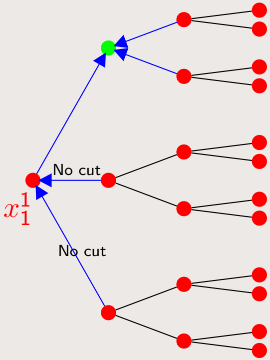


- $ST_3 = (R, F, \emptyset)$
- $ST_4 = (R, F, \emptyset)$
- If using FFFB, nothing can be done regardless of the asynchronicity level

Algorithmic Challenges

Challenge: To ensure non-blocking behavior of the algorithm

Sequencing Method



- $ST_3 = (R, F, \emptyset)$
- $ST_4 = (R, F, \emptyset)$
- If using FFFB, nothing can be done regardless of the asynchronicity level
- Potential work: continue evaluate $Q_3(x_1^1)$ and $Q_4(x_1^1)$

Algorithmic Challenges

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.

Algorithmic Challenges

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.

Proposal: Dynamic double layer sequencing protocol

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

Grid Resource Utilization Challenges

Challenge: To limit the negative effects of both contention and starvation

Grid Resource Utilization Challenges

Challenge: To limit the negative effects of both contention and starvation

Buffer size

Too small:

- Workers report results frequently
- Contention occurs

Too big:

- Can not create enough tasks to meet the demand.
- Starvation occurs

Grid Resource Utilization Challenges

Challenge: To limit the negative effects of both contention and starvation

Buffer size

Too small:

- Workers report results frequently
- Contention occurs

Too big:

- Can not create enough tasks to meet the demand.
- Starvation occurs

Proposal: Dynamic tasking scheme with node aggregation

- Dynamic tasking based on stages, LP size, and # available workers
- Aggregating nodes to increase task size

Large Data Management Challenges

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

Large Data Management Challenges

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, $x_n \in \Re^{100}$, requires $\geq$ 400MB to store cuts.

Large Data Management Challenges

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.

Large Data Management Challenges

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.

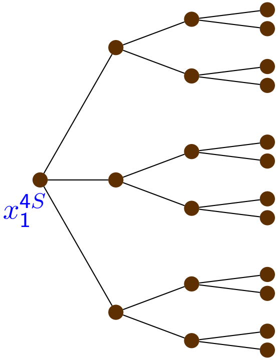
Proposal: Cut Management

- 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.

Outline

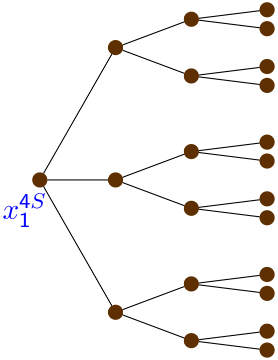
- Stochastic Linear Program
  - Linear Program vs. Stochastic Linear Program
  - Multi-stage Stochastic Linear Program
  - Nested Decomposition Algorithm
- Distributed Computing
  - Grid Computing
  - Condor & MW
- CDF Framework
- Research Challenges
- Value of Multistage Stochastic Linear Program
- Future Research

Value of MSLP



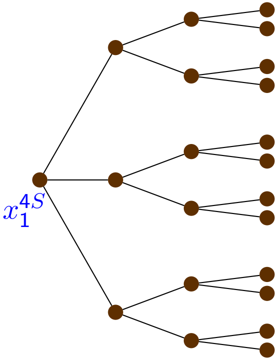
Value of MSLP

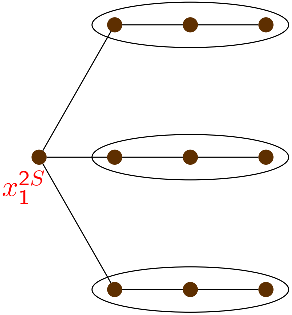
$$v^{4S} = c_1^T x_1^{4S} + \mathcal{Q}_1(x_1^{4S})$$



Value of MSLP

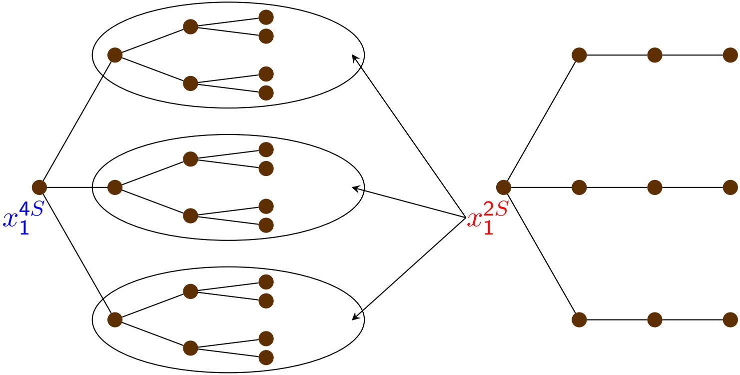
$$v^{4S} = c_1^T x_1^{4S} + \mathcal{Q}_1(x_1^{4S})$$





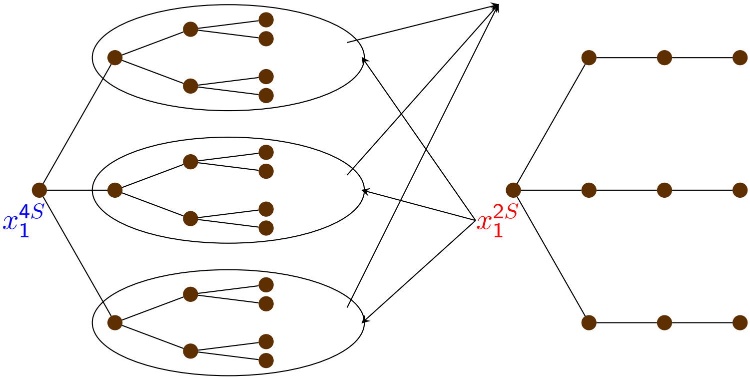
Value of MSLP

$$v^{4S} = c_1^T x_1^{4S} + \mathcal{Q}_1(x_1^{4S})$$



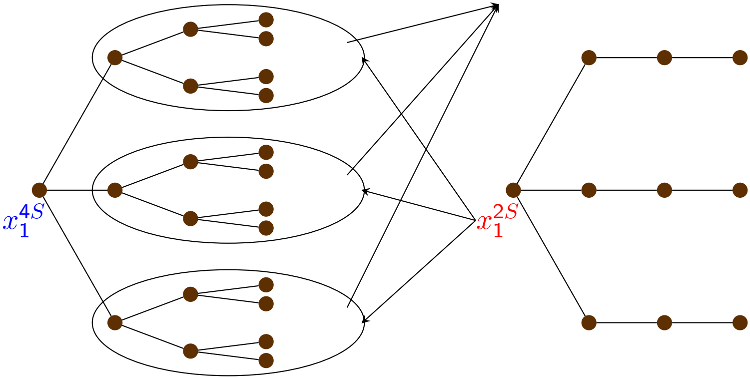
Value of MSLP

$v^{4S} = c_1^T x_1^{4S} + \mathcal{Q}_1(x_1^{4S})$ $v^{2S} = c_1^T x_1^{2S} + \mathcal{Q}_1(x_1^{2S})$



Value of MSLP

$v^{4S} = c_1^T x_1^{4S} + \mathcal{Q}_1(x_1^{4S})$ $v^{2S} = c_1^T x_1^{2S} + \mathcal{Q}_1(x_1^{2S})$



- $$VMS_{4S}^{2S} = v^{2S} - v^{4S} \geq 0$$

Value of MSLP

Test Problems

- Asset Management Problem (Blomvall and Shapiro [2004])
- Network Planning Problem (Sen, Doverspike and Cosares [1994])
- Enterprise-wide Optimization Problem (EWO-AP project)
- Dynamic Vehicle Allocation Problem (Powell [1988])
- More?

Outline

Accomplishments

1. Defined a set of consistent notations
2. Laying out challenges for Grid computing implementation.
3. Developed a *CDF framework* that enables an easy implementation of parallel nested decomposition algorithm
4. Extended the *CDF framework* to an asynchronous version with buffering
5. Created two large scale test problem instances (Asset Management and Network Planning)
6. Implemented node aggregation
7. Created a Cut-Management class in the code that enables cut hashing

Things To Do

1. Implement the *dynamic asynchronicity level* (Algorithmic Challenge)
2. Implement the *double layer sequencing protocol* (Algorithmic Challenge)
3. Implement the *dynamic tasking scheme* (Grid Resource Utilization Challenge)
4. Increase functions in Cut-Management including *cut sharing purging*, and *aggregation* (Large Date Management Challenge)
5. Create other large-scale MSLP instances (SMPS files)
6. Study the value of MSLP

Road Map

- Clean up the current code, add Cut Management functions, create more instances. (4-6 weeks)
- Test dynamic asynchronicity, sequencing, tasking. Prepare to run problem on larger Grid. (8-10 weeks)
  - Keep all results for further VMS analysis.
- Run larger computational problems, study VMS (8-10 weeks)
- Thesis writing (n months)