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

MSLP

MSLP

We make decisions everyday

MSLP

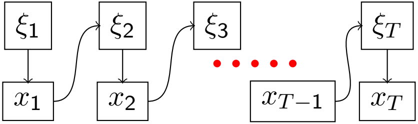
We make decisions everyday

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

MSLP

We make decisions everyday

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

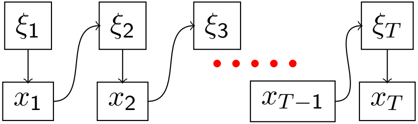


Multi-stage Stochastic Programming

MSLP

We make decisions everyday

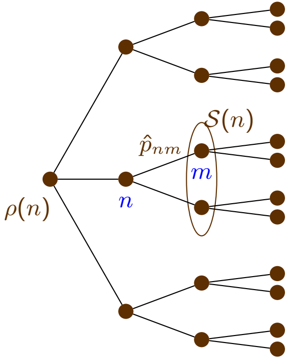
- Under uncertainty;
- Not all at the same time.



Multi-stage Stochastic Programming

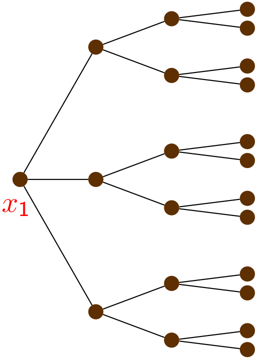
How to make a good decision (x_1) now by taking into account all future uncertainty?

Multi-stage Scenario Tree



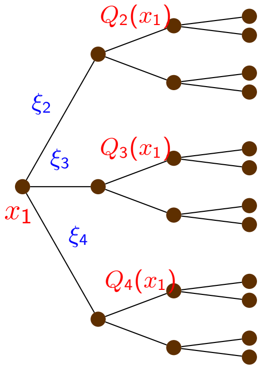
- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_{nm}$: Conditional probability that the random events leading from node n to node m occurs

Multi-stage Scenario Tree



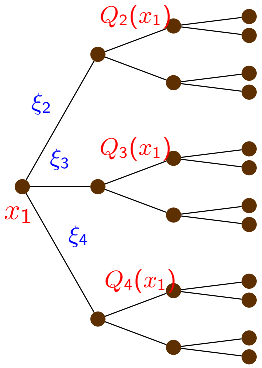
- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: Set of successor nodes of n
- $\hat{p}_{nm}$: Conditional probability that the random events leading from node n to node m occurs
- x_n: Decision taken at node n

Multi-stage Scenario Tree



- $\mathcal{N}$: Set of nodes in the tree
- $\rho(n)$: Unique predecessor of node n in the tree
- $\mathcal{S}(n)$: 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



$$\mathcal{Q}_1(x_1) = \sum_{m \in \mathcal{S}(1)} \hat{p}_{nm} Q_m(x_1)$$

- $\mathcal{N}$: 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
- $\hat{p}_{nm}$: 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
- $\mathcal{Q}_n(x_n) = \sum_{m \in \mathcal{S}(n)} \hat{p}_{nm} Q_m(x_n)$: Expected Recourse function at node n

Multi-stage Stochastic Linear Program

Recursive Model

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

where

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

and

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

Multi-stage Stochastic Linear Program

Recursive Model

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

where (Implicit)

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

and (Recursive)

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

- Bad news: $\mathcal{Q}_1(\cdot)$ is extremely difficult to evaluate;

Multi-stage Stochastic Linear Program

Recursive Model

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

where (Implicit)

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

and (Recursive)

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

- Bad news: $\mathcal{Q}_1(\cdot)$ 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

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

where (Implicit)

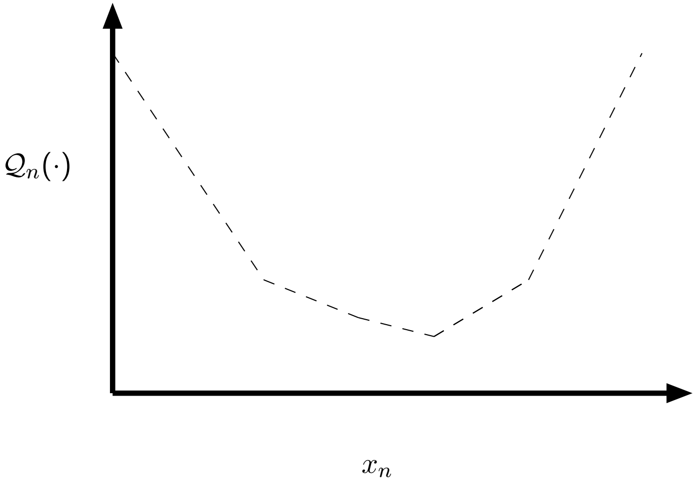
$$\mathcal{Q}_n(x_n) \stackrel{\text{def}}{=} \sum_{m \in \mathcal{S}(n)} \hat{p}_{nm} Q_m(x_n), \quad \forall n \in \mathcal{N},$$

and (Recursive)

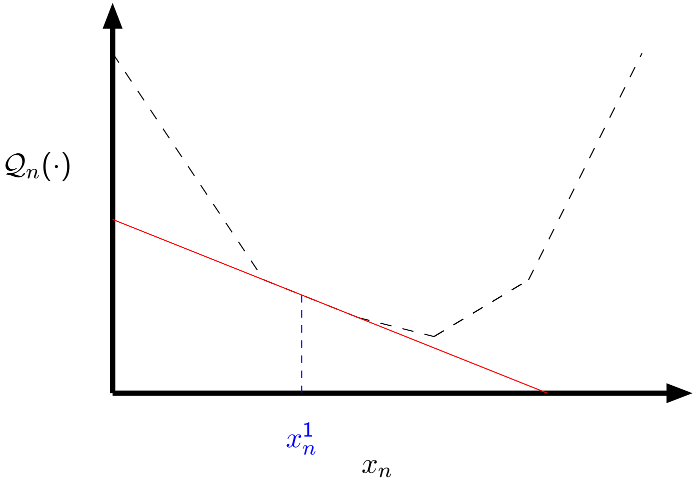
$$Q_n(x_{\rho(n)}) \stackrel{\text{def}}{=} \min_{x_n \geq 0} \left\{ c_n^T x_n + \mathcal{Q}_n(x_n) \mid W_n x_n = h_n - T_n x_{\rho(n)} \right\}, \quad \forall n \in \mathcal{N} \setminus \{1\}.$$

- Bad news: $\mathcal{Q}_1(\cdot)$ 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 $\mathcal{Q}_n(\cdot)$)

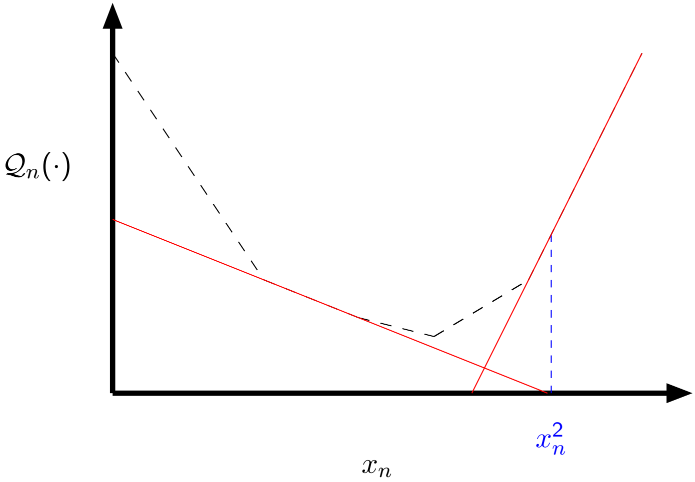
Evaluation of $\mathcal{Q}_n(\cdot)$



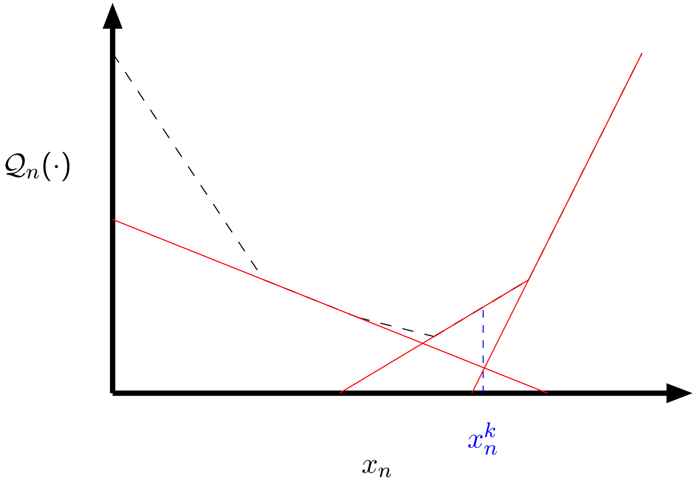
Evaluation of $\mathcal{Q}_n(\cdot)$



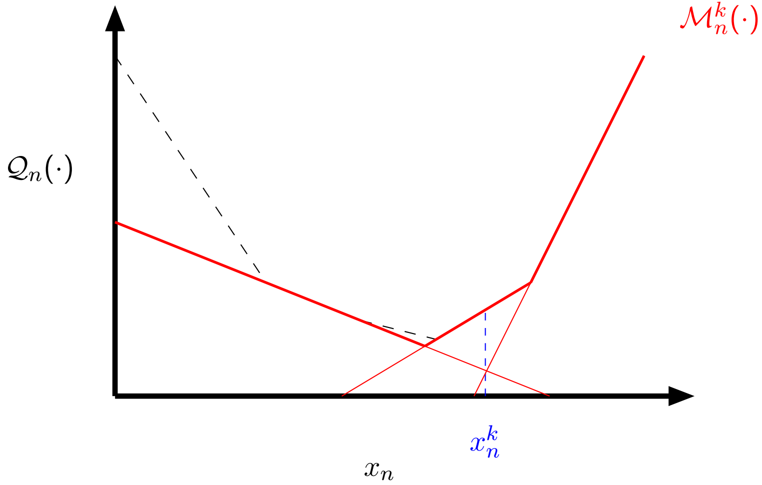
Evaluation of $\mathcal{Q}_n(\cdot)$



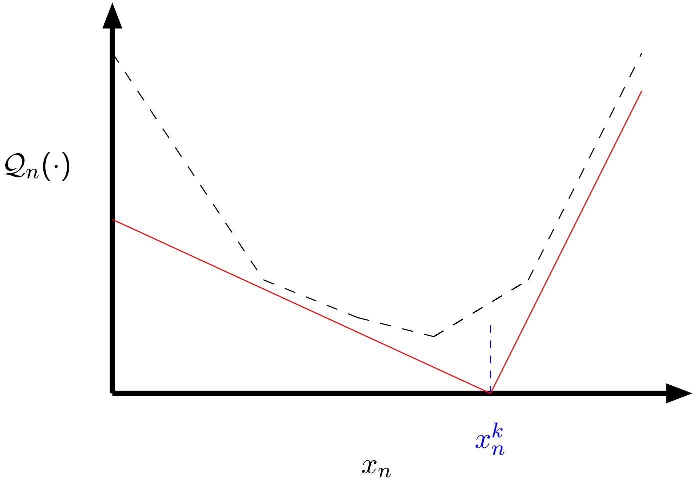
Evaluation of $\mathcal{Q}_n(\cdot)$



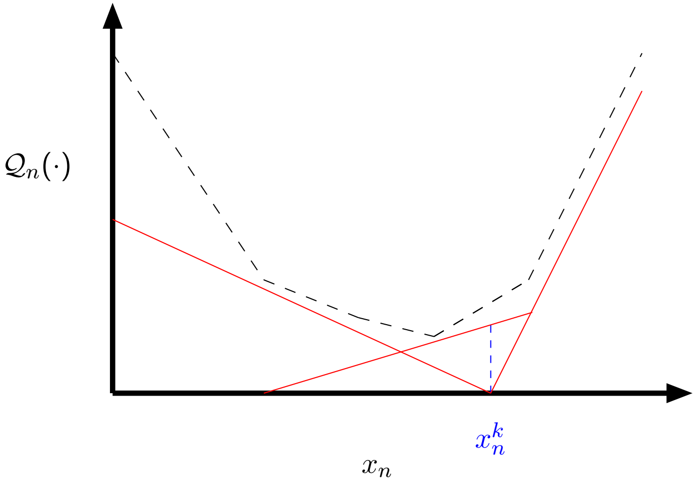
Evaluation of $\mathcal{Q}_n(\cdot)$



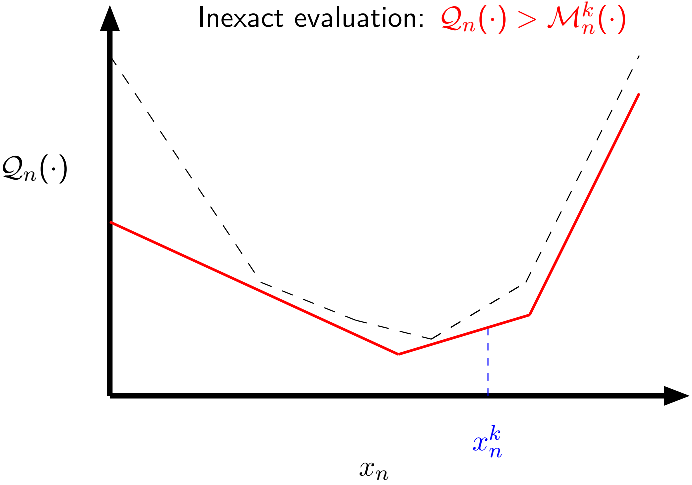
Evaluation of $\mathcal{Q}_n(\cdot)$



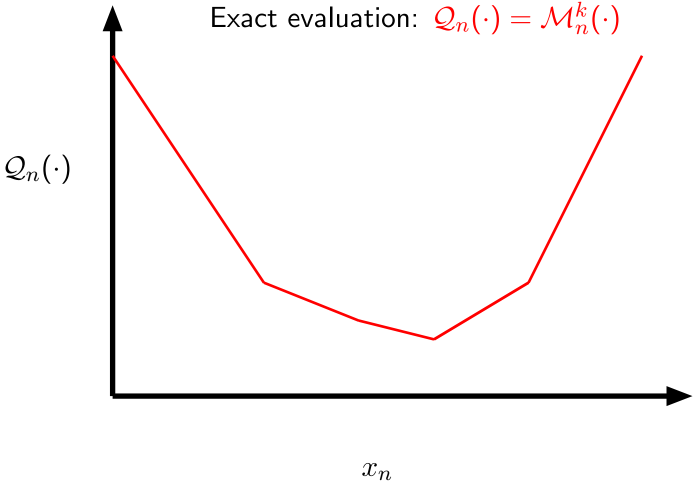
Evaluation of $\mathcal{Q}_n(\cdot)$



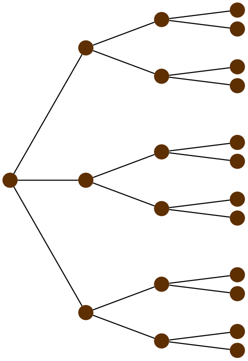
Evaluation of $\mathcal{Q}_n(\cdot)$



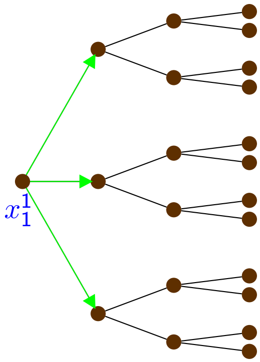
Evaluation of $\mathcal{Q}_n(\cdot)$



Nested Decomposition Algorithm

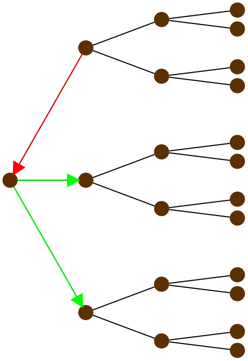


Nested Decomposition Algorithm



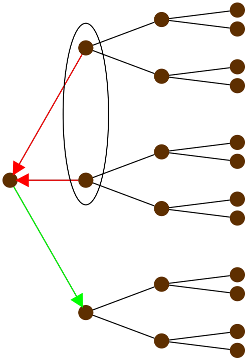
- Policy

Nested Decomposition Algorithm



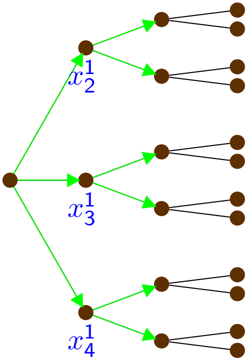
- Policy
- Feasibility Cuts
- Optimality Cuts

Nested Decomposition Algorithm



- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

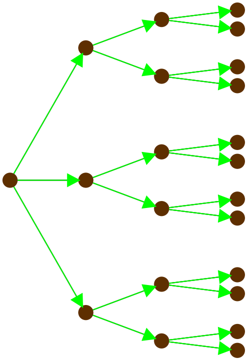
Nested Decomposition Algorithm



- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

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

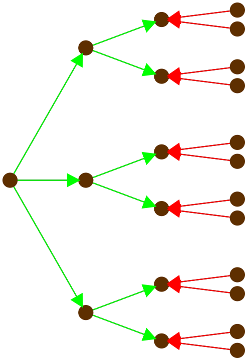
Nested Decomposition Algorithm



- Policy
- Feasibility Cuts
- Optimality Cuts
    - Clustering

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

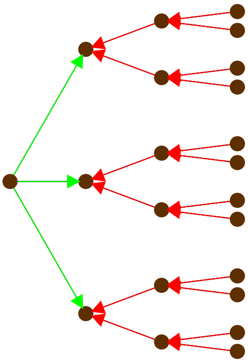
Nested Decomposition Algorithm



- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

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

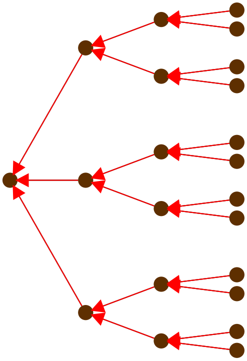
Nested Decomposition Algorithm



- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

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

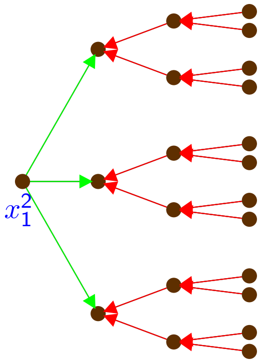
Nested Decomposition Algorithm



- Policy
- Feasibility Cuts
- Optimality Cuts
  - Clustering

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

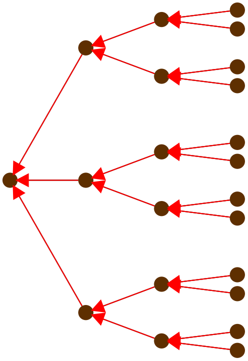
Nested Decomposition Algorithm



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

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

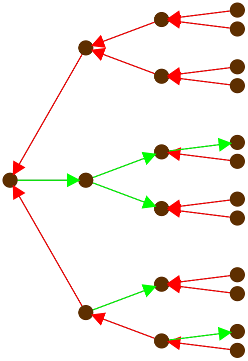
Nested Decomposition Algorithm



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

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

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?

Is Parallel Enough?

How large is the problem?

- 100 children at each node
- 6 stages

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

Is Parallel Enough?

How large is the problem?

- 100 children at each node
- 6 stages

$\longrightarrow$ 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

$\longrightarrow$ 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

$\longrightarrow$ 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

$\longrightarrow$ 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

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

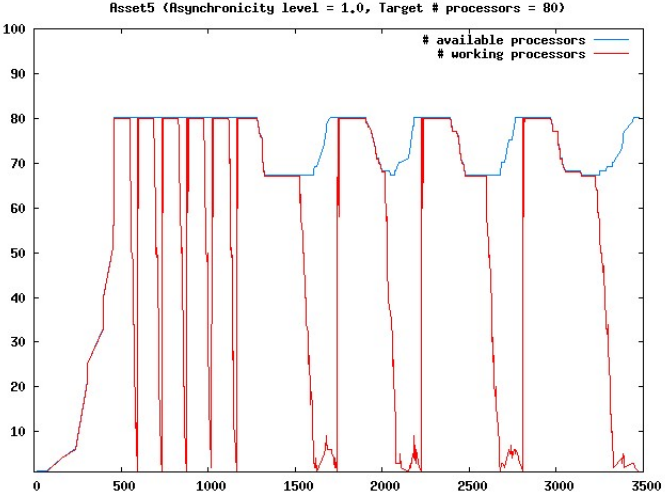
COLOR	• Red: Node has finished computation. • Yellow: Node is ready for computation. • Green: Node is under process.
DIRECTION	• $\rightarrow$ Forward: Forward job is under process or information will be passed from parent • $\leftarrow$ Backward: Backward job is under process or information will be passed from children
FLAG	• $*$ Star: Exact evaluation ($\mathcal{M}_n^k(\cdot) = \mathcal{Q}_n(x_n^{k_n})$) • $\emptyset$ Null: Inexact evaluation ($\mathcal{M}_n^k(\cdot) < \mathcal{Q}_n(x_n^{k_n})$)

CDF Framework – Trigger Signals

Signal	Destination	Command
Start	$\rho(n) \to n$	Start to evaluate $\mathcal{Q}_{\rho(n)}(\cdot)$ under policy $x_{\rho(n)}^{k_{\rho(n)}}$
Update	$\rho(n) \to n$	Update model $\mathcal{M}_{\rho(n)}(\cdot)$ given policy $x_{\rho(n)}^{k_{\rho(n)}}$
Restart	$n \to \rho(n)$	find a new policy $x_{\rho(n)}^{k_{\rho(n)}}$
Done	$n \to \rho(n)$	new model updated, but $\mathcal{M}_{\rho(n)}(\cdot) < \mathcal{Q}_{\rho(n)}(\cdot)$
End	$n \to \rho(n)$	new model updated, and $\mathcal{M}_{\rho(n)}(\cdot) = \mathcal{Q}_{\rho(n)}(\cdot)$
Terminate	$n \to Siblings$	Do not evaluate $\mathcal{Q}_{\rho(n)}(\cdot)$ under policy $x_{\rho(n)}^{k_{\rho(n)}}$
Go	$n \to Siblings$	Join the task and go to the Grid

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

Asynchronicity

Challenge: What is a proper level of asynchronicity?

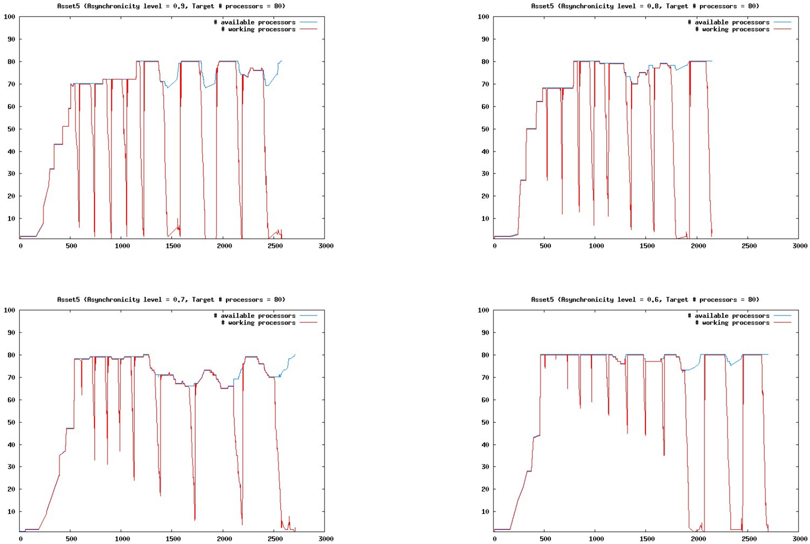
Asynchronicity Level

- 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

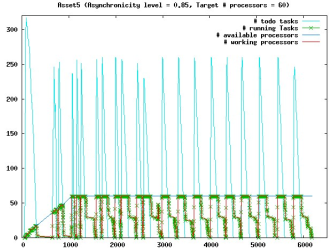
Approach: Dynamic asynchronicity level

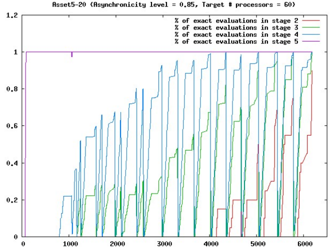
- Stage-dependent (test the impact of asynchronicity level to different stages)
- Resource-dependent (enable more accurate evaluation when resources are limited)

Asynchronicity is a must

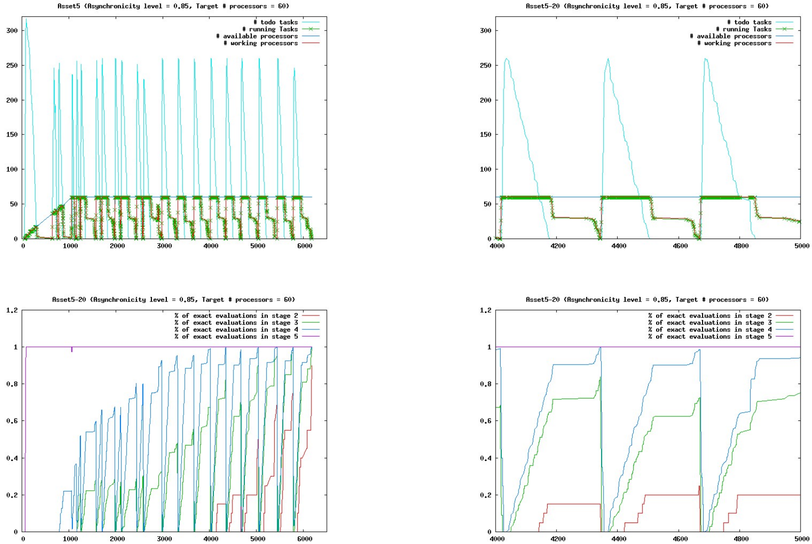


Sequencing Mechanism





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

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