Assessment of an asynchronous parallel implementation for the stochastic mid-term hydrothermal planning problem

Lilian C. Brandão and André L. Diniz

September 13, 2018

Outline

Hydrothermal Planning Problem

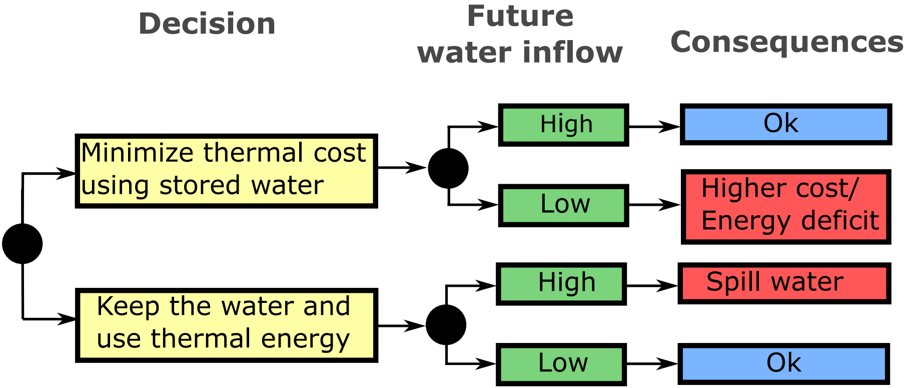
$\Rightarrow$ Resource: Energy (thermal and hydraulic)

$\Rightarrow$ Minimize thermal generation cost

$\Rightarrow$ Subject to : (1) system demand
(2) physical constraint of the units
(3) stochastic future water inflow

Hydrothermal Planning Problem

- ⇒ Resource: Energy (thermal and hydraulic)
- ⇒ Minimize thermal generation cost
- ⇒ Subject to : (1) system demand
  (2) physical constraint of the units
  (3) stochastic future water inflow

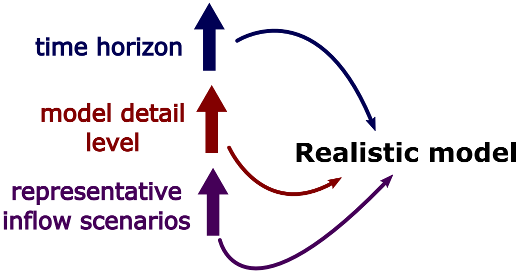


Hydrothermal Planning Problem

- Real life problem
- Brazilian operation planning is made using stochastic optimization models developed by Cepel

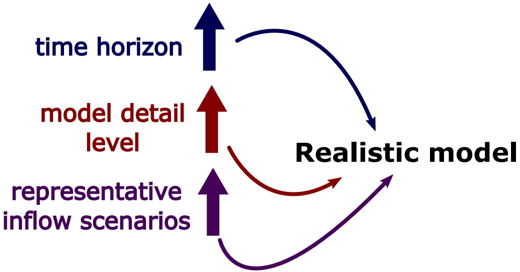
Hydrothermal Planning Problem

- Real life problem
- Brazilian operation planning is made using stochastic optimization models developed by Cepel



Hydrothermal Planning Problem

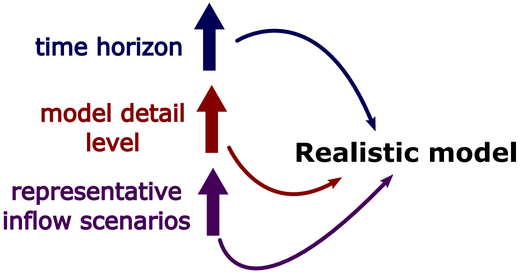
- Real life problem
- Brazilian operation planning is made using stochastic optimization models developed by Cepel



- Usually the problem has high dimension
- May take considerable CPU time to be solved

Hydrothermal Planning Problem

- Real life problem
- Brazilian operation planning is made using stochastic optimization models developed by Cepel



- Usually the problem has high dimension
- May take considerable CPU time to be solved

- Application on the Brazilian short/mid term model DECOMP
- Enable considering uncertainty in the first month, larger time horizon and improve the detailing of the features

Hydrothermal Planning Problem

Problem considered: Mid-term Hydrothermal Coordination (2-months / 1 year):

$$\min \sum_{t=1}^{NT} \sum_{s=1}^{NS} \sum_{i=1}^{NT} \sum_{p=1}^{NL} ct_{i,t} \times gt_i^{p,t,s} + \sum_{s=1}^{NS} FCF(V_i^{T,s})$$

$\rightarrow$ Objective Function

$$\sum_{i=1}^{NT} gt_i^{p,t,s} + \sum_{j=1}^{NH} gh_j^{p,t,s} = d^{p,t}, \forall p, s, t$$

$\rightarrow$ Energy Balance

$$v_i^{t,s} + \sum_{p=1}^{NL} (q_i^{p,t,s} + s_i^{p,t,s}) - \sum_{p=1}^{NL} \sum_{j \in \Omega^{up}} (q_j^{p,t,s} + s_j^{p,t,s}) = v_i^{t-1,r} + I_i^{t,s}, \ \forall i, s, t$$

$\rightarrow$ Water Balance

$$gh_i^{p,t,s} \leq \gamma_{0,i,t,s}^{(k)} + \gamma_{v,i,t,s}^{(k)} (v_i^{t-1,r} + v_i^{t,s})/2 + \gamma_{q,i,t,s}^{(k)} q_i^{p,t,s} + \gamma_{s,i,t,s}^{(k)} s_i^{p,t,s}$$

$\forall k, i, p, t, s$ $\rightarrow$ Hydro Production Function

and others...

Contributions of this work

- Dual Dynamic programming is widely used to solve stochastic optimization problems
- As an iterative method it can take a large number of iterations to converge
- We seek to accelerate the solution process by using several cores
- We propose an **Asynchronous dual dynamic programming (ADDP)** which is more suitable for parallel applications

Outline

Dual Dynamic Programming

- Decompose the problem into time stages and scenarios subproblems (scenario tree)
- The time stages are coupled by state variables
- Iteratively traverse the tree solving the subproblems
- Build costs approximation based on dual solutions
- Stops by some convergence criteria, usually when the cost approximation is close enough to the real cost

Algorithm iteration:

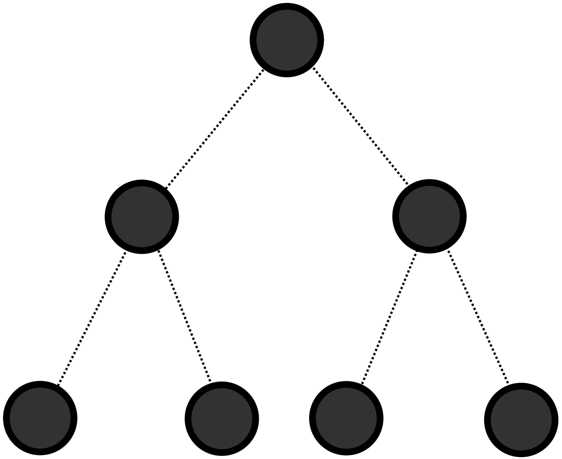
- Forward pass: travel forward along the tree transmitting state variables
- Backward pass: travel backward along the tree transmitting cost information
- Convergence test

Traditional Parallelization

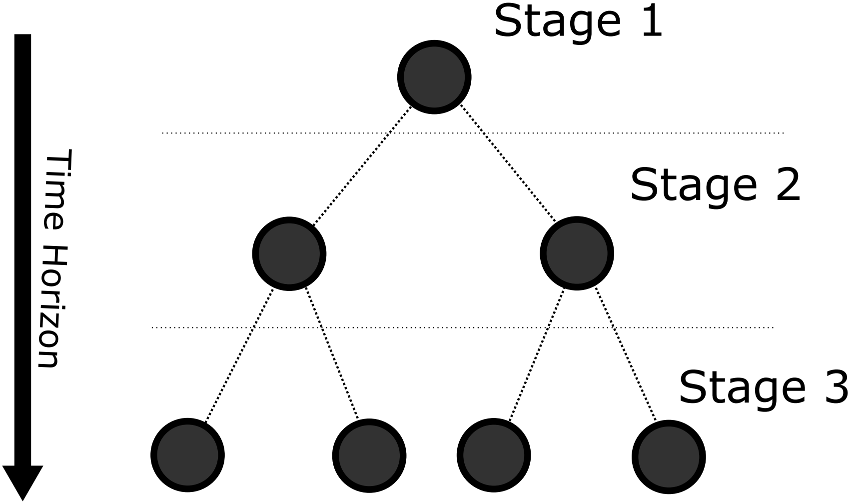
[Dempster and Thompson, 1998]

- Nodes at the same time period are solved in parallel
- In-between time periods nodes must wait either for states from the previous period or cuts from next period
- There is a synchronization point between the time periods
- The maximum number of processors that can be used is number of leaf nodes on the tree

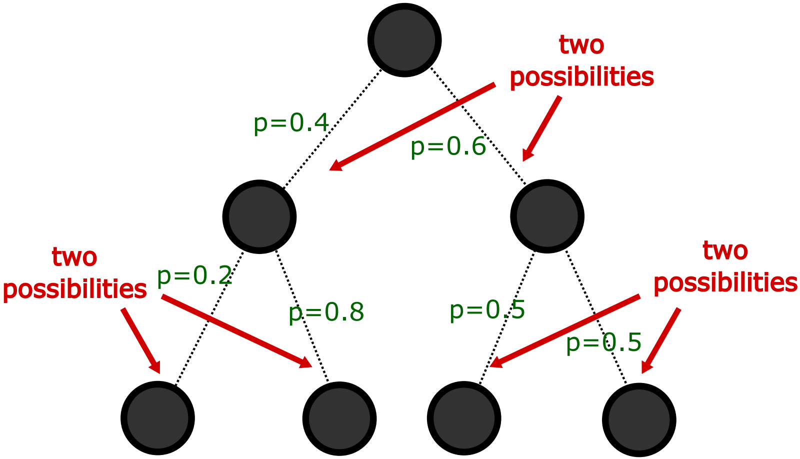
Dual Dynamic Programming (Parallel)



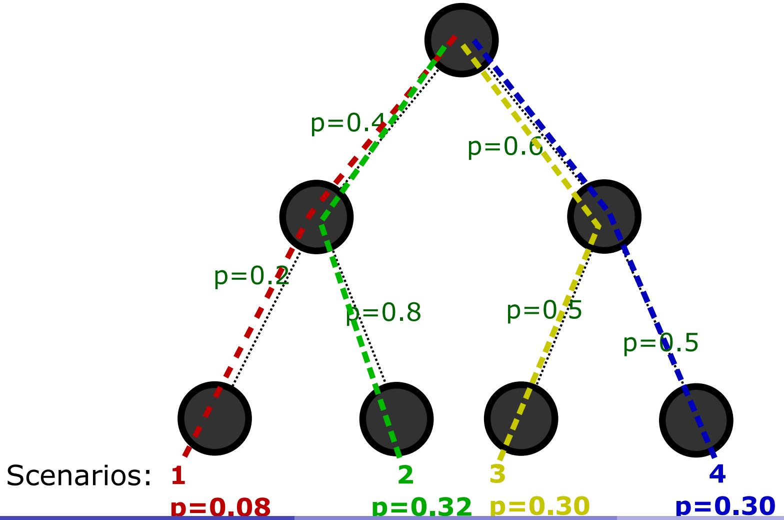
Dual Dynamic Programming (Parallel)



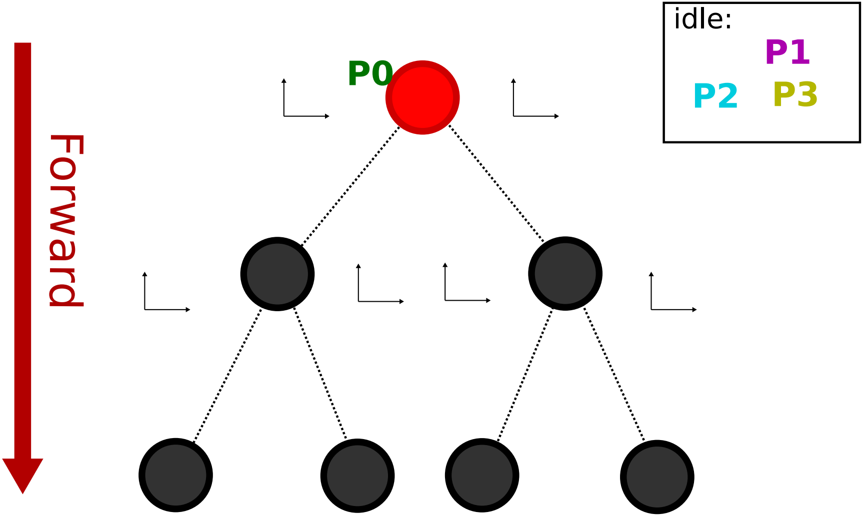
Dual Dynamic Programming (Parallel)



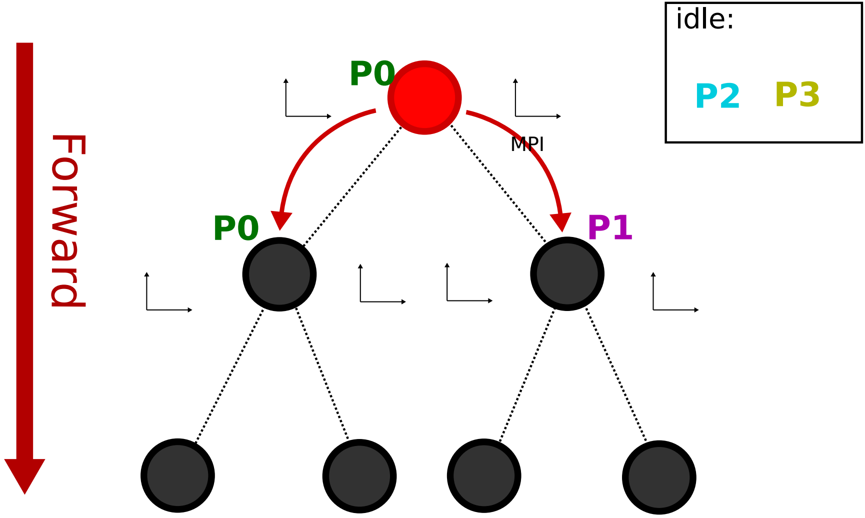
Dual Dynamic Programming (Parallel)



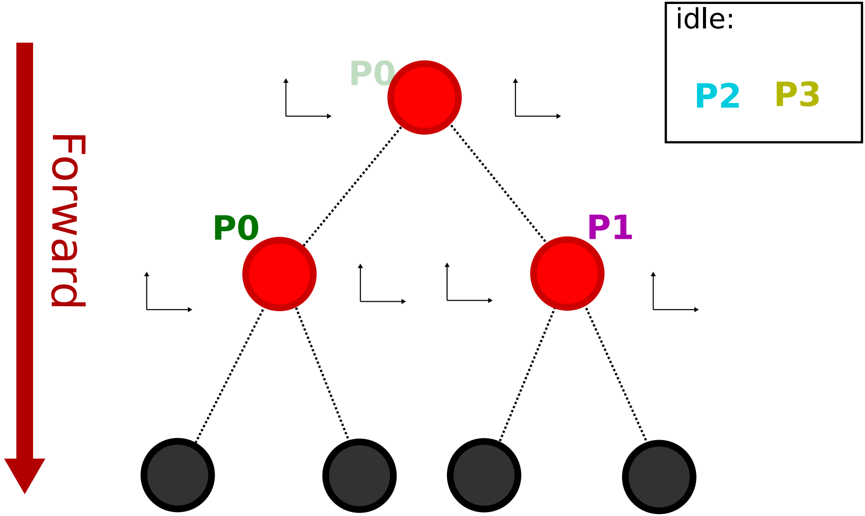
Dual Dynamic Programming (Parallel)



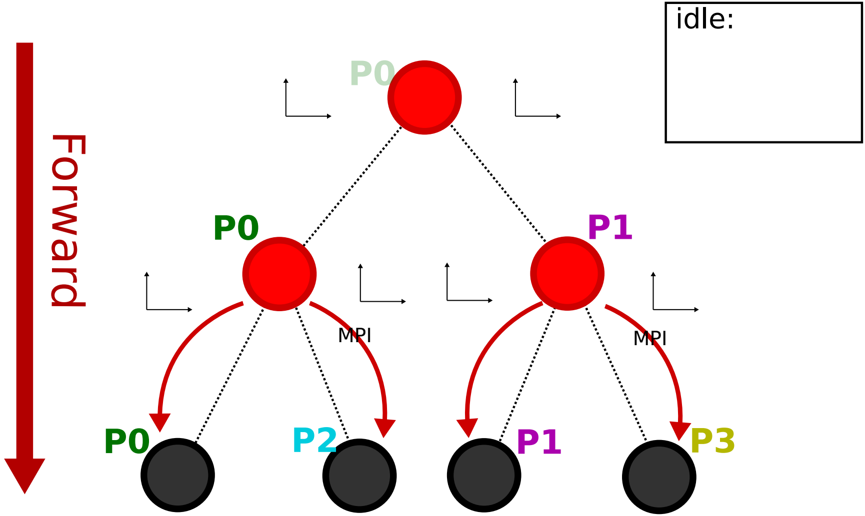
Dual Dynamic Programming (Parallel)



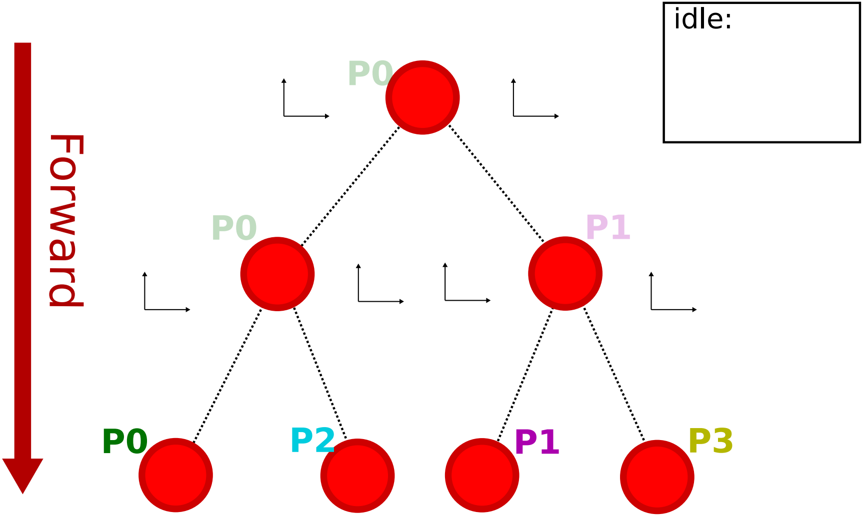
Dual Dynamic Programming (Parallel)



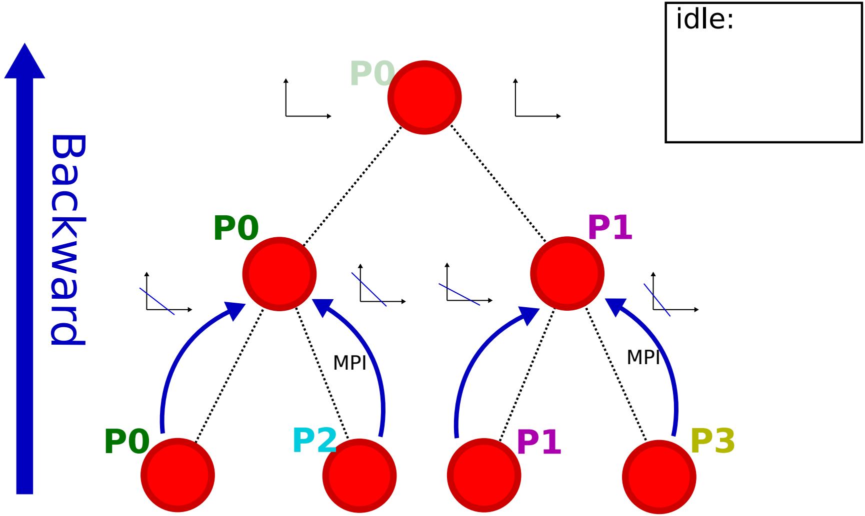
Dual Dynamic Programming (Parallel)



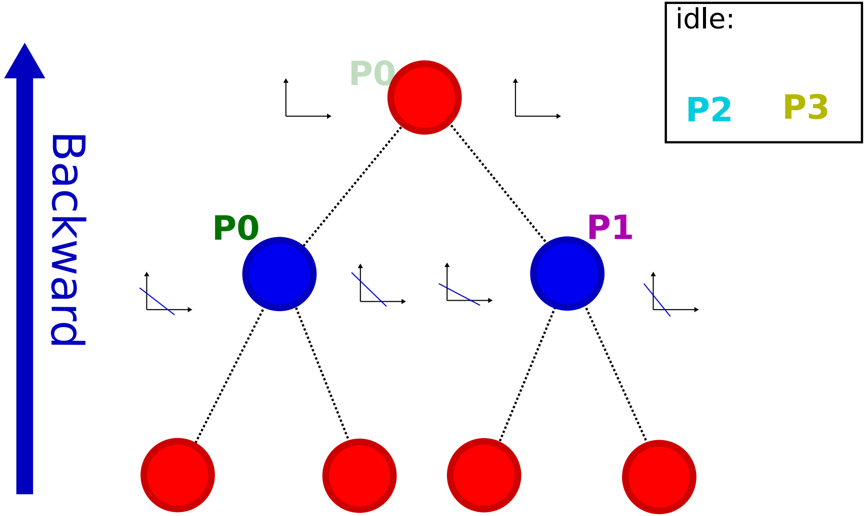
Dual Dynamic Programming (Parallel)



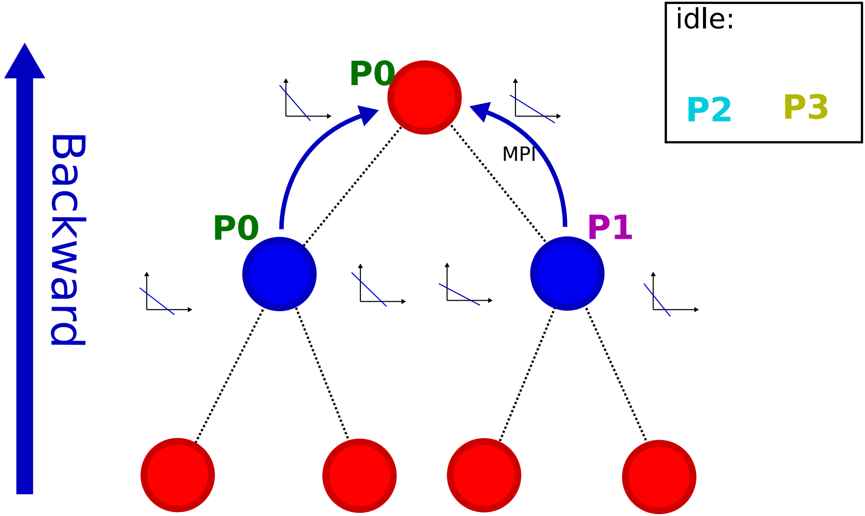
Dual Dynamic Programming (Parallel)



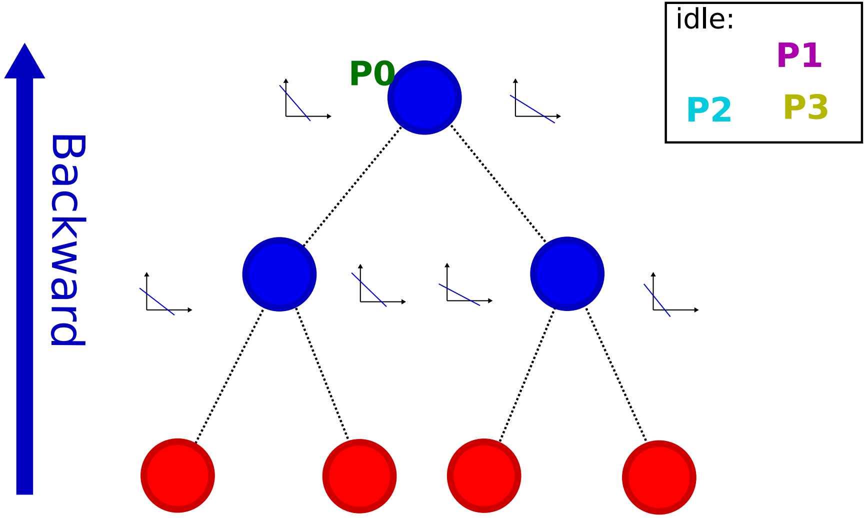
Dual Dynamic Programming (Parallel)



Dual Dynamic Programming (Parallel)



Dual Dynamic Programming (Parallel)



Outline

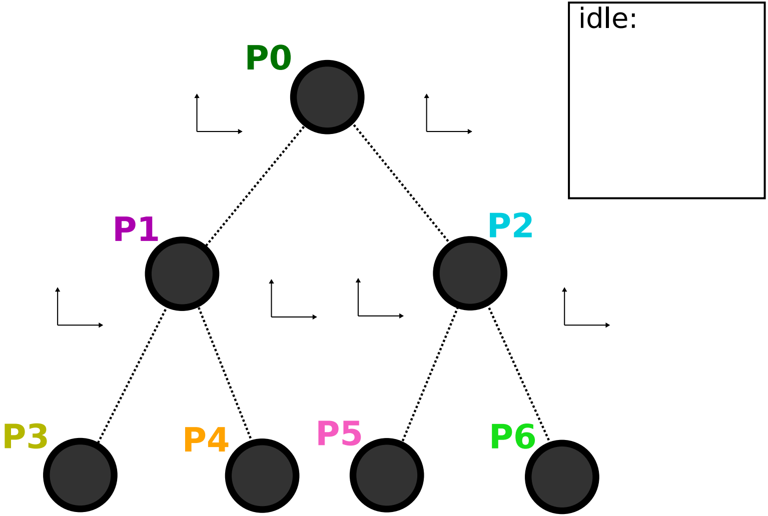
Asynchronous Dual Dynamic Programming (ADDP)

- Performs **steps** instead of iterations
- Allows full node-wise parallelization within each step
- Information is transmitted between steps
- Convergence criteria is similar to DDP

Algorithm steps:

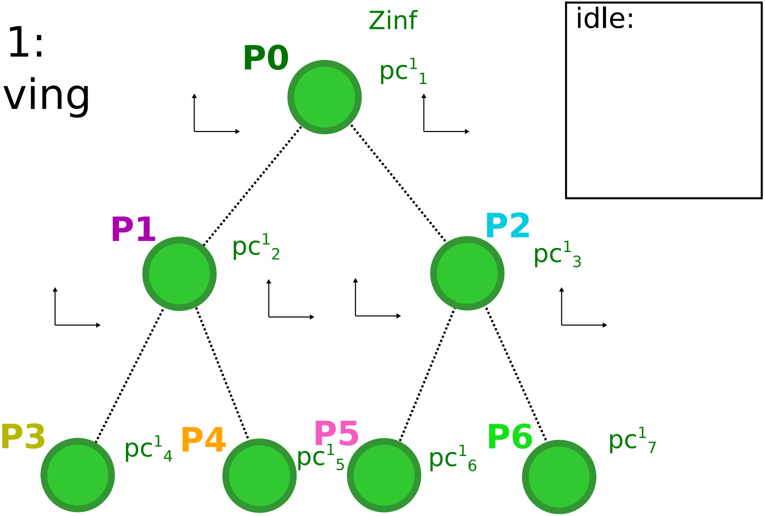
- Solve **all** nodes independently
- Test convergence by comparing the cost approximation (Benders cuts) and the real cost (sum of nodes solutions) of the corresponding step
- Transmit information, state variables and Benders cuts, through the tree

Asynchronous Dual Dynamic Programming (ADDP)



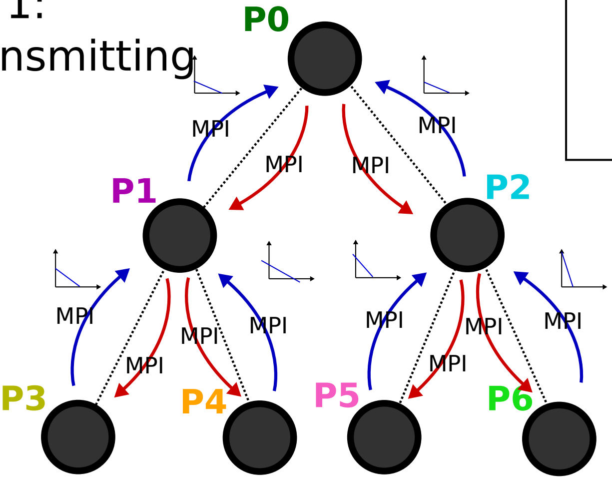
Asynchronous Dual Dynamic Programming (ADDP)

Step 1:
- Solving



Asynchronous Dual Dynamic Programming (ADDP)

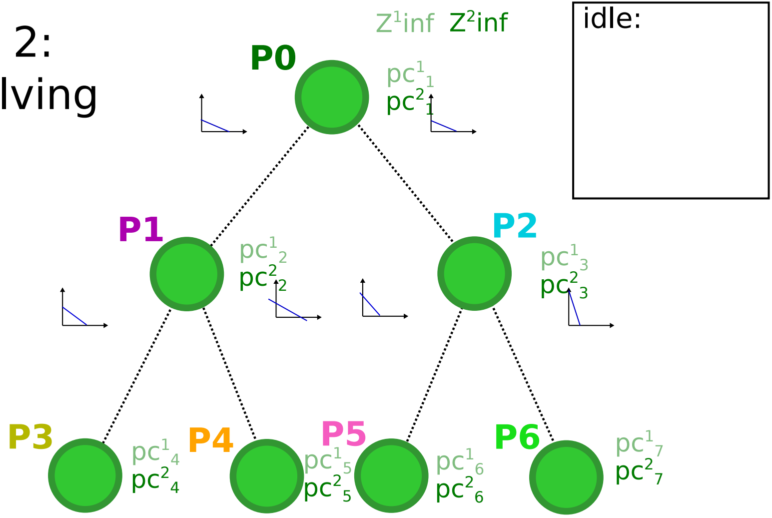
Step 1:
- transmitting



idle:

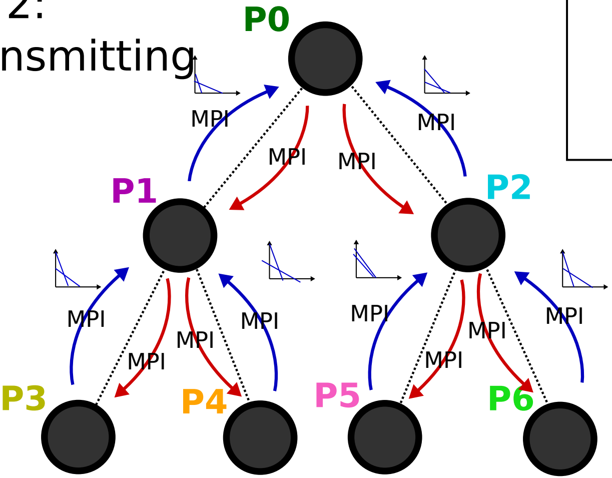
Asynchronous Dual Dynamic Programming (ADDP)

Step 2:
- Solving



Asynchronous Dual Dynamic Programming (ADDP)

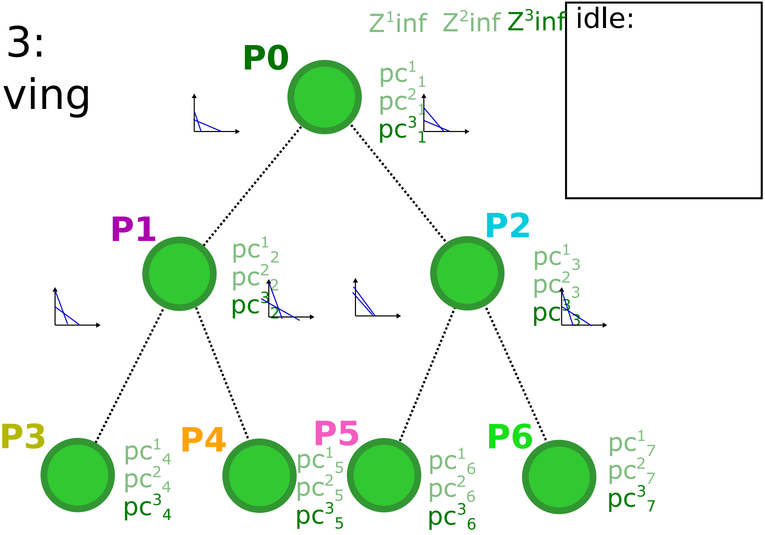
Step 2:
- transmitting



idle:

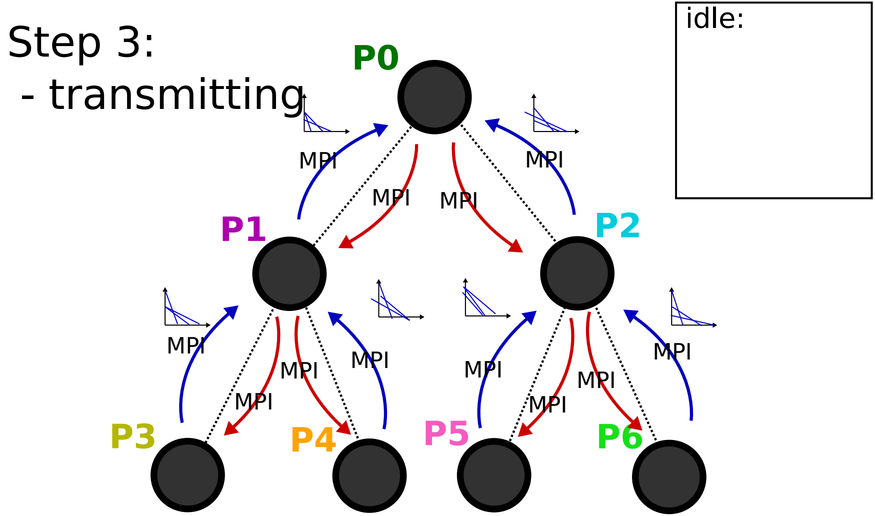
Asynchronous Dual Dynamic Programming (ADDP)

Step 3:
 - Solving



$Z^1sup = pc^1{}_1+p_2pc^2{}_2+p_3pc^2{}_3+p_4pc^3{}_4+p_5pc^3{}_5+p_6pc^3{}_6+p_7pc^3{}_7$

Asynchronous Dual Dynamic Programming (ADDP)

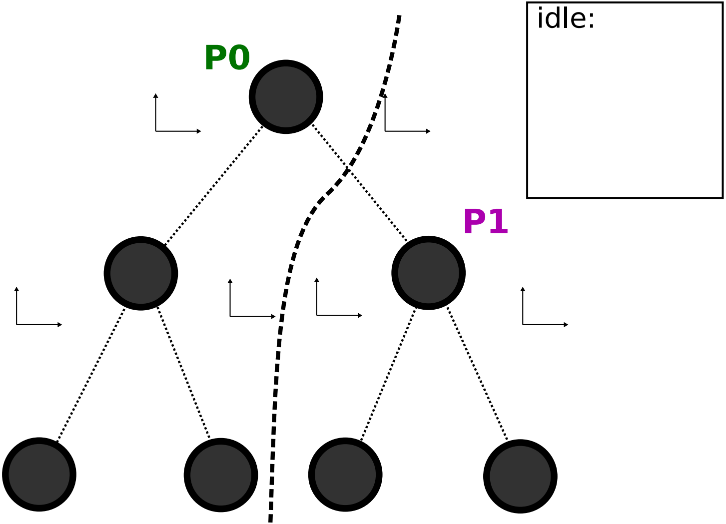


Outline

Partial Asynchronous Dual Dynamic Programming (PADDP)

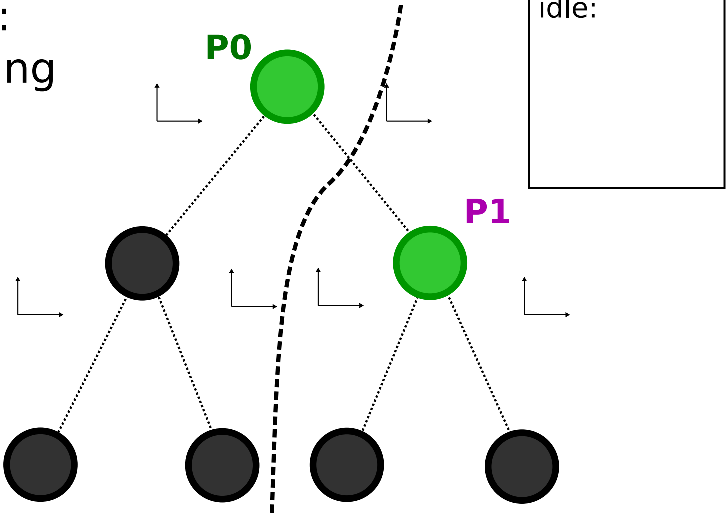
- The scenario tree may have a very large number of nodes. In that case, even for ADDP, nodes may share the same processor
- Makes use information of nodes in the same processor
- A partial synchronism is conveniently introduced at the algorithm
- We seek to increase the convergence rate, as the number of processors decrease

Partial Asynchronous Dual Dynamic Programming (PADDP)



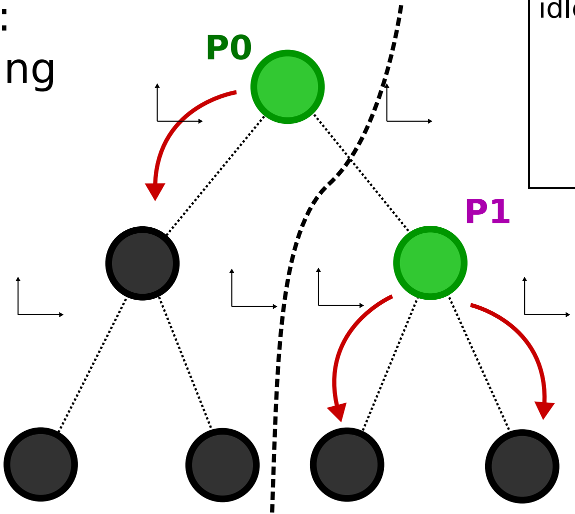
Partial Asynchronous Dual Dynamic Programming (PADDP)

Step 1:
- Solving



Partial Asynchronous Dual Dynamic Programming (PADDP)

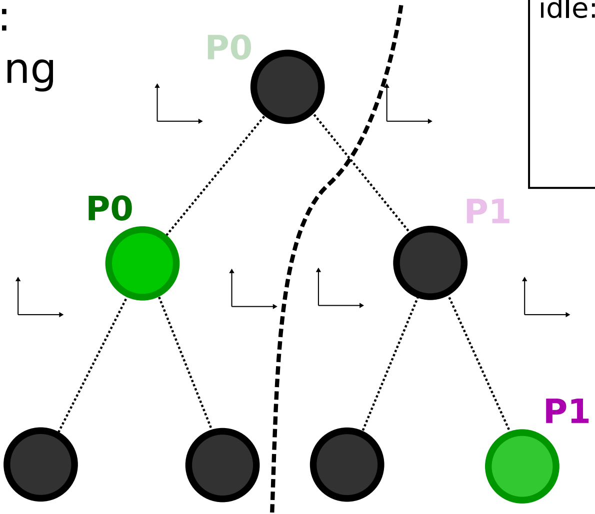
Step 1:
- Solving



idle:

Partial Asynchronous Dual Dynamic Programming (PADDP)

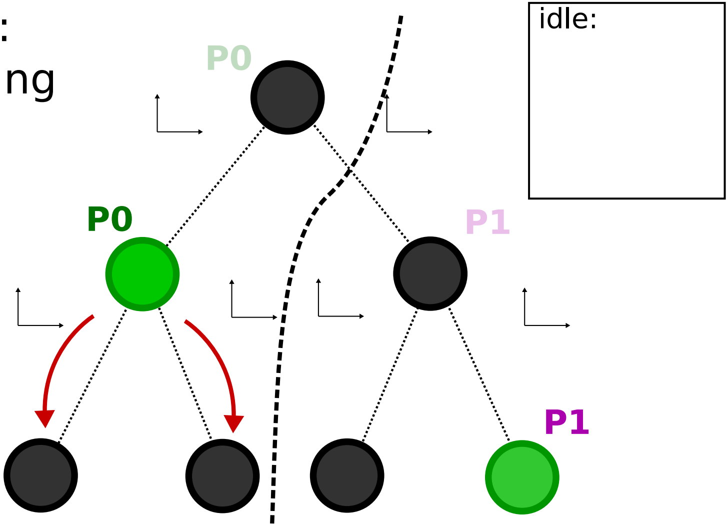
Step 1:
- Solving



idle:

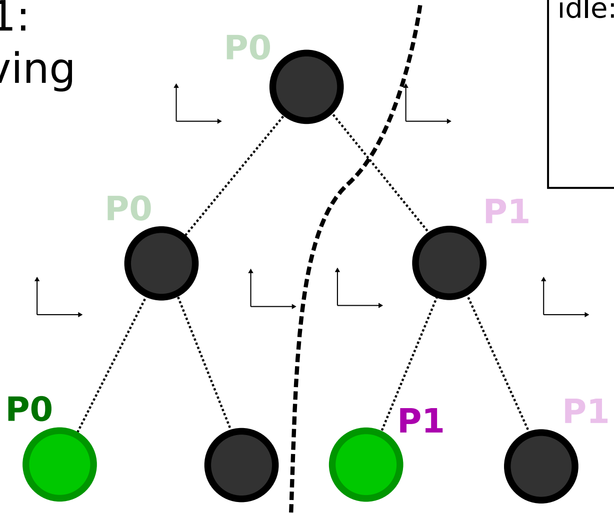
Partial Asynchronous Dual Dynamic Programming (PADDP)

Step 1:
- Solving



Partial Asynchronous Dual Dynamic Programming (PADDP)

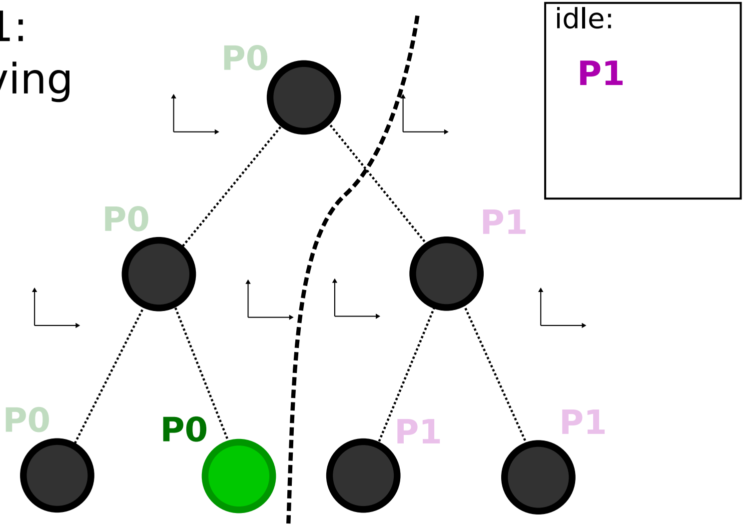
Step 1:
- Solving



idle:

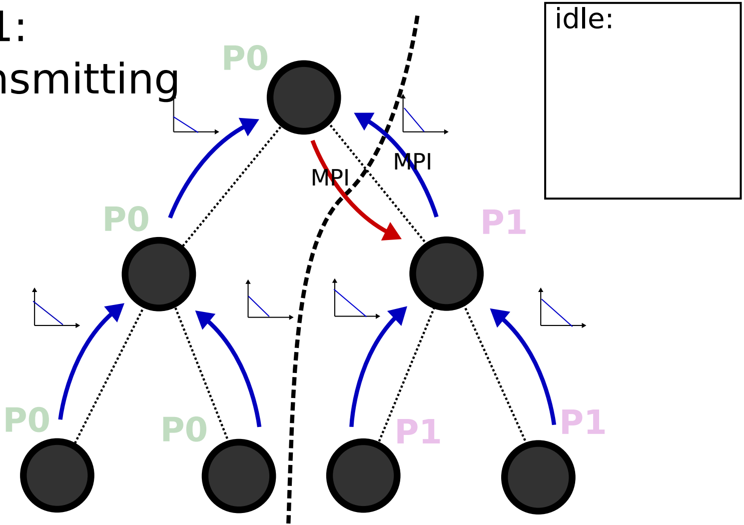
Partial Asynchronous Dual Dynamic Programming (PADDP)

Step 1:
- Solving



Partial Asynchronous Dual Dynamic Programming (PADDP)

Step 1:
- Transmitting



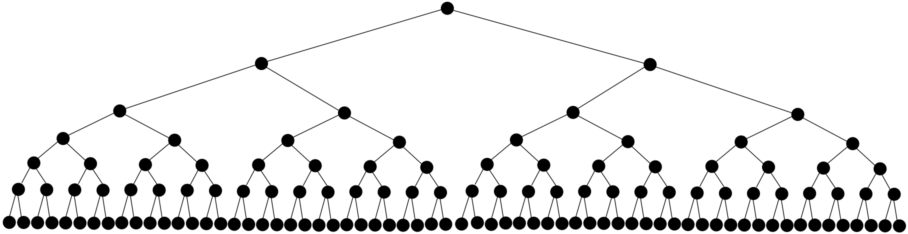
Outline

Study cases

- 84 hydro plants and 46 thermal plants;
- Constrains: load supply, water balance, future cost function (long term model), piecewise linear hydro power function;
- 4 different inflow scenario trees:

Study case 1:

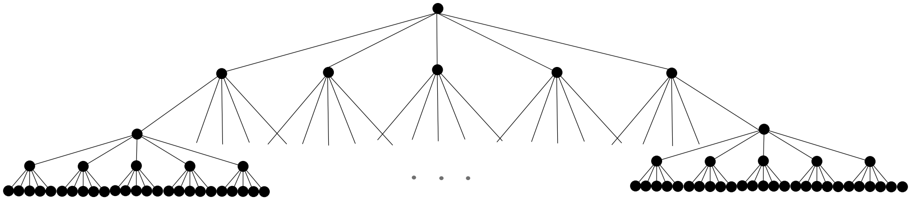
Total number of nodes:	127						
Time periods:	1	2	3	4	5	6	7
# Scenarios per period:	1	2	2	2	2	2	2



Study cases

Study case 2:

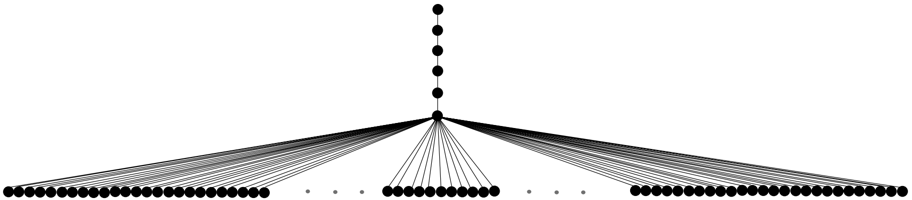
Total number of nodes:	781				
Time periods:	1	2	3	4	5
# Scenarios per period:	1	5	5	5	5



Study cases

Study case 3 (official PMO scenario tree):

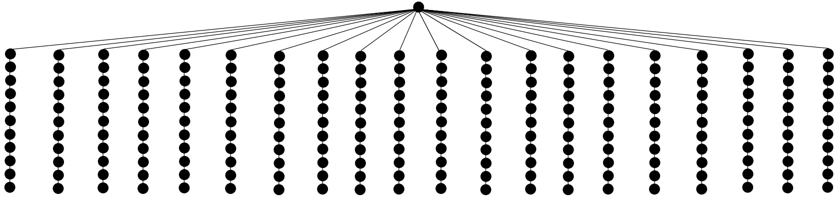
Total number of nodes:	306						
Time periods:	1	2	3	4	5	6	7
# Scenarios per period:	1	1	1	1	1	1	300



Study cases

Study case 4:

Number of nodes:	221										
Time periods:	1	2	3	4	5	6	7	8	9	10	11	12
Scenarios per period:	1	20	1	1	1	1	1	1	1	1	1	1

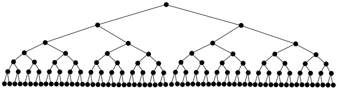


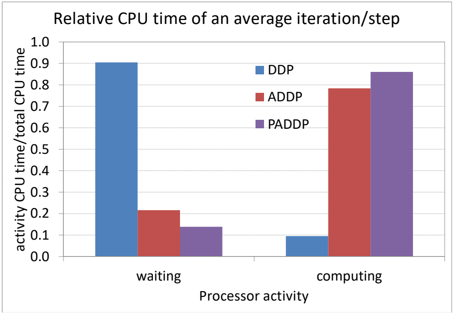
Results on Parallel Processing

Algorithms assessed:

- **DDP**: traditional Dual Dynamic Programming parallelization [Dempster and Thompson, 1998]
- **ADDP**: the proposed asynchronous approach
- **PADDP**: the alternative partial asynchronous framework

Parallel Results - Case 1

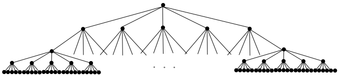


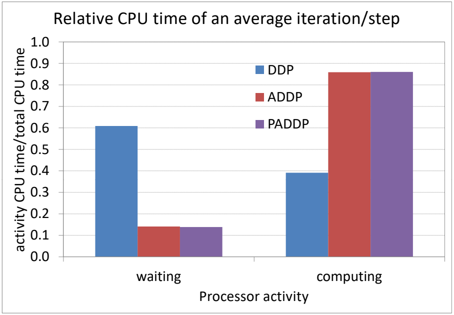


Number of Nodes: 127

Number of Processors										
1	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
DDP t(s):										
1845	655	517	347	358	317	321	321	326	323	327
ADDP t(s):										
3563	408	302	175	147	135	103	104	101	93	89
PADDP t(s):										
1897	337	249	177	110	85	84	73	65	71	70

Parallel Results - Case 2

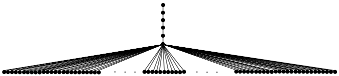


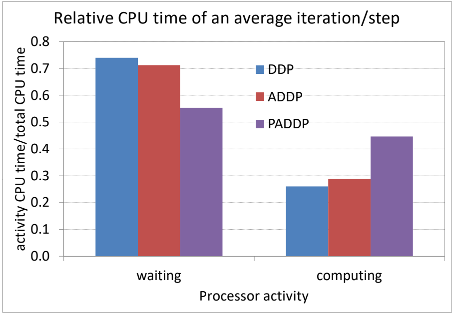


Number of Nodes: 781

Number of Processors										
1	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
DDP t(s):										
9536	2727	1676	1164	925	817	590	526	514	519	504
ADDP t(s):										
19042	2017	1102	752	578	470	408	363	310	290	272
PADDP t(s):										
9944	1432	825	533	431	373	313	268	243	233	236

Parallel Results - Case 3

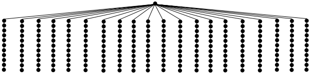


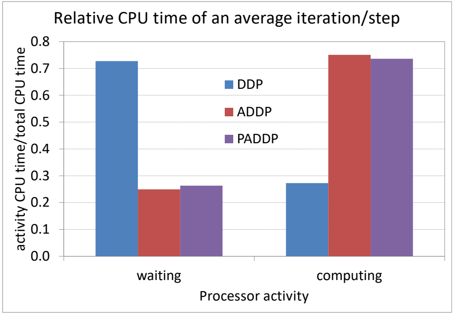


Number of Nodes: 306

Number of Processors									
1	**12**	**24**	**36**	**48**	**60**	**84**	**96**	**108**	**120**
DDP t(s):									
5132	1211	804	643	562	506	478	476	452	428
ADDP t(s):									
13075	2053	1397	1293	1222	1175	1088	1054	1060	1037
PADDP t(s):									
6266	998	724	473	576	429	492	682	466	475

Parallel Results - Case 4





Number of Nodes: 221

Number of Processors										
1	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
DDP t(s):										
6179	1024	725	722	748	730	715	735	725	736	735
ADDP t(s):										
10652	1143	806	581	480	437	409	350	357	348	282
PADDP t(s):										
6566	641	400	320	270	274	247	208	221	202	171

Outline

1. Application on Hydrothermal Planning Problem
2. Parallel Dual Dynamic Programming
3. Asynchronous Dual Dynamic Programming
4. Partial Asynchronous Dual Dynamic Programming
5. Simulations and Results
6. Conclusions

Conclusions and further work

Main results:

- Asynchronous dual dynamic programming (ADDP):
  - Allows full node-wise parallelization
  - The sequential time is worse than DDP
  - However with a small number of processors the wall time is better than DDP
  - Speedup and efficiency are better than DDP

Conclusions and further work

Main results:

- Asynchronous dual dynamic programming (ADDP):
  - Allows full node-wise parallelization
  - The sequential time is worse than DDP
  - However with a small number of processors the wall time is better than DDP
  - Speedup and efficiency are better than DDP
- Partial asynchronous dual dynamic programming (PADDP):
  - Has better convergence rate than ADDP
  - Algorithm changes with the number of processors, producing different results
  - Yields the best computational time

Conclusions and further work

Main results:

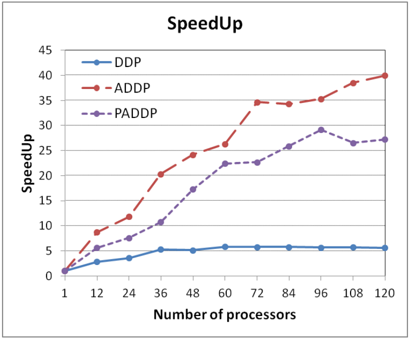
- Asynchronous dual dynamic programming (ADDP):
  - Allows full node-wise parallelization
  - The sequential time is worse than DDP
  - However with a small number of processors the wall time is better than DDP
  - Speedup and efficiency are better than DDP
- Partial asynchronous dual dynamic programming (PADDP):
  - Has better convergence rate than ADDP
  - Algorithm changes with the number of processors, producing different results
  - Yields the best computational time

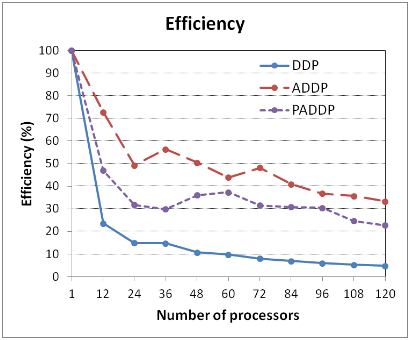
Future work:

- Study better distribution of the nodes into the processors (PADDP)
- Study the use of a dynamic parallelization scheme

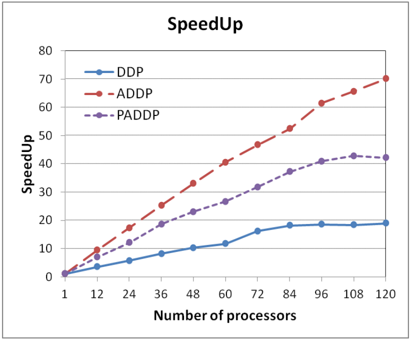
Dempster, M. A. H. and Thompson, R. T. (1998).
Parallelization and aggregation ofnested benders decomposition.
Annals of Operations Research, 81(0):163–188.

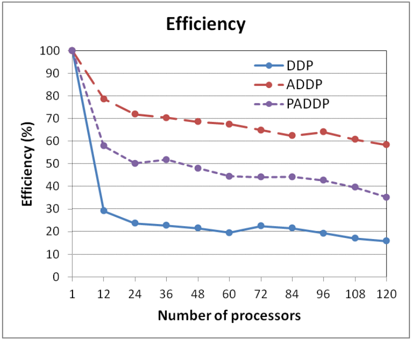
Speedup and efficiency - Study case 1



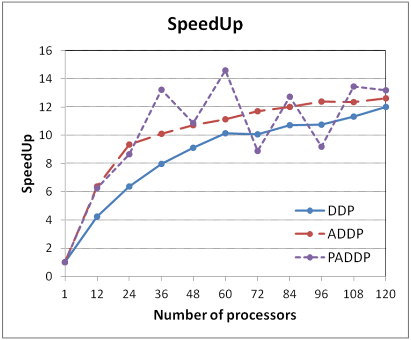


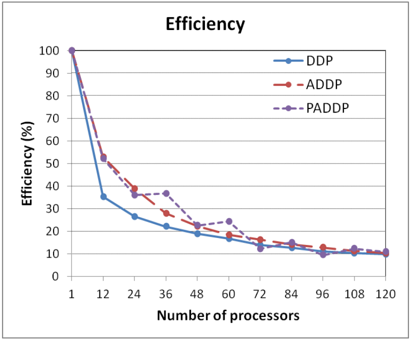
Speedup and efficiency - Study case 2



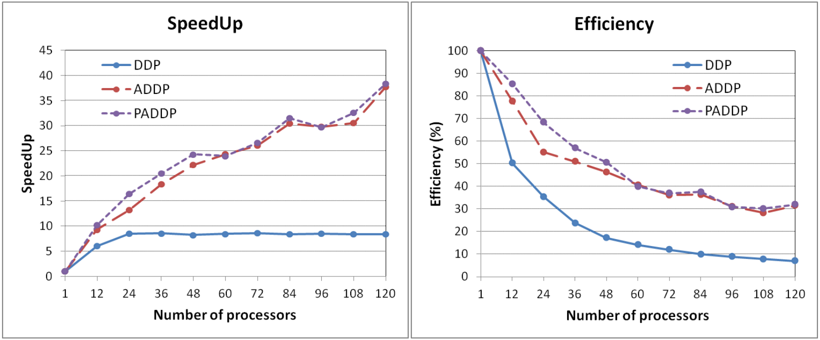


Speedup and efficiency - Study case 3

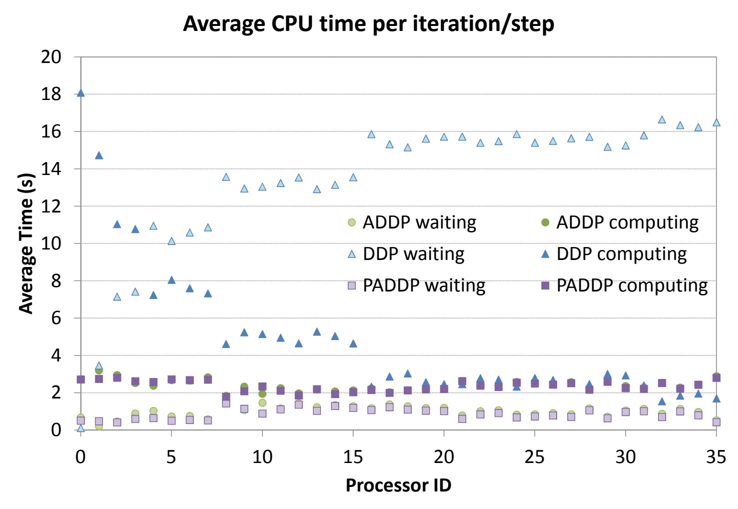




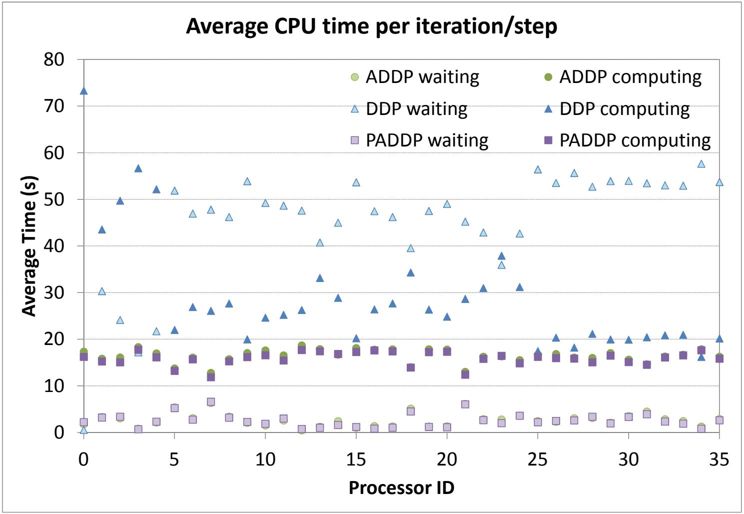
Speedup and efficiency - Study case 4



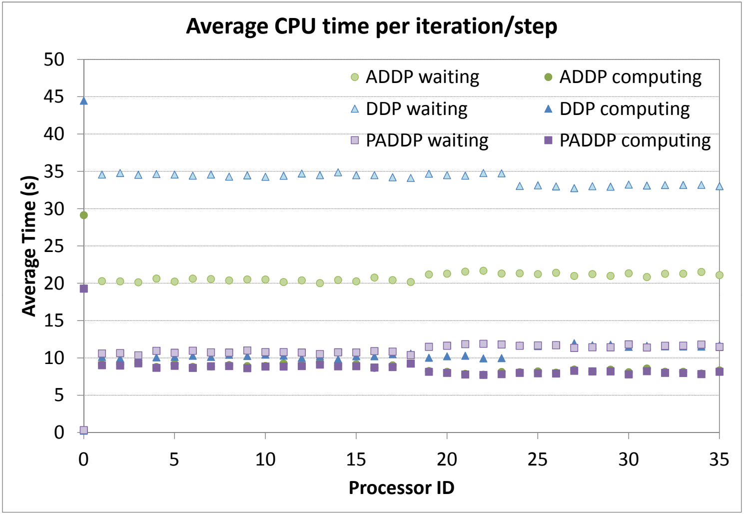
Average CPU time in each processor - Study case 1



Average CPU time in each processor - Study case 2



Average CPU time in each processor - Study case 3



Average CPU time in each processor - Study case 4