Grupo de Estudo de Operação de Sistemas Elétricos-GOP

Programação Dinâmica Dual Assíncrona - aplicação ao problema de planejamento de curto/médio prazos da operação hidrotérmica

LILIAN CHAVES BRANDAO DOS SANTOS(1); ANDRE LUIZ DINIZ(1);
CEPEL(1);

RESUMO

Problemas planejamento hidrotérmico de curto/médio prazo podem ser modelados como um problema de otimização estocástica multi-estágios. Quanto maior o horizonte de tempo, dimensão do sistema, detalhamento na representação de incertezas, mais o modelo se aproxima da realidade, porém mais custoso é para resolvê-lo. Este trabalho propõe um método de solução para este problema, baseado na Programação Dinâmica Dual (PDD), porém mais naturalmente adaptável à ambientes multiprocessados, o que visa permitir uma resolução mais eficiente. A estratégia proposta, denominada Programação Dinâmica Dual Assíncrona (PDDA), é validada em estudos de caso de grande porte com o sistema brasileiro.

PALAVRAS-CHAVE

Programação Dinâmica Dual, Planejamento Hidrotérmico, Processamento Paralelo, Otimização Estocástica Multi-estágio

1.0 - INTRODUÇÃO

O problema de planejamento hidrotérmico de curto/médio prazo visa determinar o despacho hídrico e térmico semanal ou mensal que minimiza os custos de operação considerando também riscos e impacto na operação futura. São representadas diversas restrições de geração e transmissão, assim como as incertezas associadas à parâmetros do sistema (1). Dessa forma este problema rapidamente adquire grandes dimensões sendo modelado como um problema estocástico multi-estágios. Para resolvê-lo é normalmente necessário usar técnicas de decomposição para lidar com as grandes dimensionalidades, como por exemplo, a Programação Dinâmica Dual (PDD) (2), *Progressive Hedging* (3), relaxação Lagrangeana (4), entre outros. Este trabalho considera o método da PDD, que é largamente utilizado na literatura, cuja principal ideia é decompor a árvore de cenários do problema e iterativamente construir aproximações da função de recurso (*recourse function*) de cada nó da árvore. Como resultado, um problema de grandes dimensões torna-se um conjunto de subproblemas menores que são resolvidos individualmente e iterativamente até que a convergência do problema global seja atingida. Uma vez que a PDD pode levar um grande número de iterações para convergir, acelerar esse processo é crucial, especialmente para problemas com um horizonte grande e vários cenários

Uma estratégia extremamente eficiente capaz de reduzir drasticamente o tempo de execução de um algoritmo é o paralelismo, cujo intuito é identificar tarefas que podem ser executadas concomitantemente e distribuí-las entre diferentes processadores. As tecnologias de hardware evoluíram muito em termos de capacidade de processamento, o que tornou acessíveis arquiteturas multi-processadas tanto em computadores desktop remotos, que chegam a ter dezenas de processadores, quanto em servidores e clusters que podem ter centenas e milhares de processadores disponíveis. Dessa forma, a capacidade do multi-processamento de reduzir tempo computacional

(*) Av. Horácio Macedo, n˚ 354, Cidade Universitária – CEP 21.941-911 Rio de Janeiro, RJ, – Brasil
Tel: (+55 21) 2598-6059 – Fax: (+55 21) 2260-1340 – Email: liliancbs@cepel.br

está hoje atrelada à capacidade do algoritmo de se adaptar ao paralelismo. Características que afetam a performance de algoritmo em ambiente paralelo, e consequentemente sua capacidade de redução de tempo, estão ligadas a escalabilidade, interdependência entre fluxos de execução, homogeneidade (balanceamento), granularidade, entre outras.

A computação paralela tem sido estudada e aplicada em diversas áreas incluindo otimização estocástica, onde algumas estratégias foram propostas no intuito de melhor paralelizar o algoritmo da PDD. No entanto, o fluxo intrínseco à PDD possui alto nível de dependência, o que exige pontos de sincronismo entre os processadores causando gargalos e reduzindo seu desempenho paralelo. Apesar disso, os subproblemas associados aos nós da árvore em um mesmo estágio são independentes e a paralelização pode ser explorada entre eles, enquanto que a dependência entre-estágios restringe alguns algoritmos paralelos da PDD que foram propostos na literatura, como por exemplo em (5), onde a ideia é resolver sub-árvores independentes em paralelo, ou em (6), que explora o paralelismo entre os nós de um mesmo estágio. Em ambos, várias instâncias de problemas estocásticos multi-estágios foram testadas e os resultados mostraram que o *speedup* e eficiência paralela são altamente sensíveis ao tamanho e forma da árvore de cenários. Adicionalmente, os autores constataram uma queda considerável no tempo de execução ao usar um pequeno número de processadores, porém observou-se uma saturação prematura do *speedup* ao empregar uma quantidade maior de processadores.

Algumas técnicas tentaram amenizar esta dependência temporal da PDD. Em (7) os autores propuseram um assincronismo parcial para resolver problemas estocásticos de dois estágios, onde o primeiro estágio é resolvido mesmo sem que todos os nós do segundo estágio tenham terminado seu processamento, o que evita ociosidade e desbalanceamento. Os autores mostraram que a técnica não afeta a convergência do método e se mostrou muito adequada para problemas heterogêneos em ambiente paralelo. Outra técnica de paralelismo assíncrono foi proposta em (8) para problemas multi-estágios, onde um subproblema é resolvido assim que uma nova entrada está disponível, seja ela estado ou corte. Apesar de a técnica mostrar redução de ociosidade entre os processadores, o critério de convergência deste método não é claro e os resultados mostrados são pouco conclusivos. Mais recentemente, (9) propôs um versão assíncrona para a PDD Determinística (PDDD), que é utilizada para resolver problemas multi-estágios determinísticos, onde não existe nenhum tipo de independência entre estágios. Os autores resolvem então os subproblemas de forma independente e as informações são trocadas (estados e cortes) após esta resolução. Os resultados mostraram uma redução significante no tempo de processamento comparado ao algoritmo sequencial.

Este trabalho apresenta dois algoritmos assíncronos baseados na PDD que visam diminuir a dependência temporal viabilizando uso completo dos recursos paralelos, de forma a obter maior eficiência e *speedup* em ambientes multiprocessados. Os resultados mostram melhores tempos de execução ao resolver um problema de despacho hidrotérmico de curto-médio prazo, atualmente utilizado no modelo DECOMP (10) para calcular o preço marginal da operação (PMO) e o preço de liquidação de diferenças (PLD)

2.0 - PROGRAMAÇÃO DINÂMICA DUAL

A PDD é um método de decomposição baseado em aproximações lineares por partes das funções de recurso de cada nó compostas por cortes, e que é capaz de resolver problemas estocásticos multi-estágios de uma forma iterativa. O método estrutura o problema como uma árvore de cenários. Cada nó desta árvore representa um subproblema de um cenário e um período de tempo, ou seja, cada nível da árvore representa um estágio temporal e nós no mesmo nível representam cenários associados a realizações da variável estocástica. A Figura 1 mostra um exemplo de uma árvore de cenários que contém três estágios de tempo (*t=1,2,3*) e a cada estágio possui dois possíveis cenários, associados à probabilidades (*p1* e *p2*), totalizando sete subproblemas ou nós.

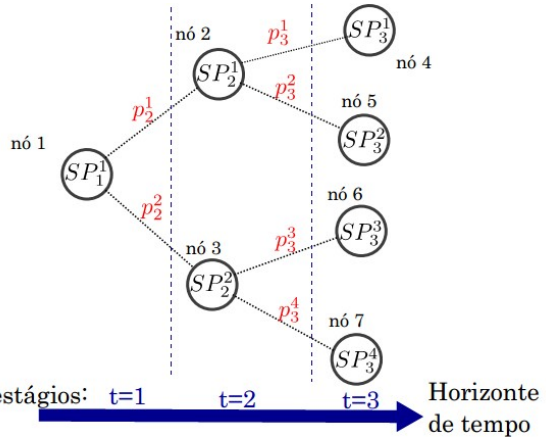


Figura 1 – Exemplo de árvore de cenários, estágios e probabilidades associadas.

A ideia do algoritmo da PDD é obter uma solução viável em cada subproblema e construir uma aproximação linear do seu custo em relação a um estado inicial, associado à esta solução. A medida que o processo iterativo avança e

vários estados iniciais são visitados, um modelo linear por partes é construído para essa função de custo. O método converge quanto esse modelo está suficientemente próximo da função real, resultando em uma equivalência dos limite inferior e superior do valor ótimo do problema, calculados ao longo do algoritmo. Cada iteração da PDD é dividida em dois passos:

- Passo *Forward*: consiste em resolver todos os subproblemas da árvore começando do estágio 1 até o último estágio, propagando os valores dos estados, ou seja, ao resolver o estágio 1 e tomar uma decisão sobre o volume final dos reservatórios, este torna-se o volume inicial dos reservatórios no estágio 2, e assim por diante.

- Passo *Backward*: consiste em resolver todos os subproblemas novamente, porém agora começando do último estágio até o primeiro, onde cada nó constrói uma aproximação linear que representa o custo associado ao estado inicial enviado no passo *Forward*.

Ao longo das iterações, vários passos *Forward* e *Backward* são executados, dessa forma, valores de estados são visitados e cortes são construídos. Após determinado número de iterações, o modelo linear passa a ser uma boa aproximação do custo real de cada subproblema e, quando isso acontece, a solução de cada subproblema individual também passa a ser uma boa aproximação da solução global do problema.

3.0 - PDD PARALELA TRADICIONAL

Um método clássico de paralelizar o algoritmo da PDD foi proposto em (6) e será usado neste trabalho como base para comparação e avaliação dos resultados das técnicas aqui propostas. A ideia é manter a mesma estrutura de iterações da PDD e explorar a independência entre cenários que estão em um mesmo estágio, resolvendo-os em paralelo. Pontos de sincronização são inseridos a cada estágio, onde os processadores devem esperar por informações dos nós descendentes (no passo *Backward*) e do nó ascendente (no passo *Forward*). Esse ponto de sincronização é um grande inconveniente, pois ele cria ociosidade nos processadores, tanto quando estão esperando pelos dados, como também limitando a granularidade máxima do algoritmo. Ou seja, como em cada estágio o número máximo de processadores ativos está limitado à quantidade de nós deste estágio, nos primeiros estágios, onde a quantidade de nós é menor, o número de processadores ativos também será menor. Além disso, notamos que o desempenho do paralelismo, que está relacionado com a granularidade e ociosidade, estará altamente relacionado com o tamanho e forma da árvore de cenários.

A Figura 2(a) mostra como seria a distribuição dos nós nos processadores neste algoritmo, para árvore de cenários apresentada na Figura 1. Neste exemplo, a PDD é capaz de utilizar até quatro processadores, nomeados *p0*, *p1*, *p2* e *p3*. No primeiro estágio (*t=1*), apenas o processador *p0* é utilizado para resolver o primeiro subproblema, no segundo estágio (*t=2*), o processador *p0* é utilizado novamente para resolver o primeiro subproblema, em paralelo à resolução do segundo subproblema, realizada no processador *p1*. Finalmente, no terceiro estágio (*t=3*), os quatro processadores são utilizados, resolvendo cada um dos subproblemas paralelamente.

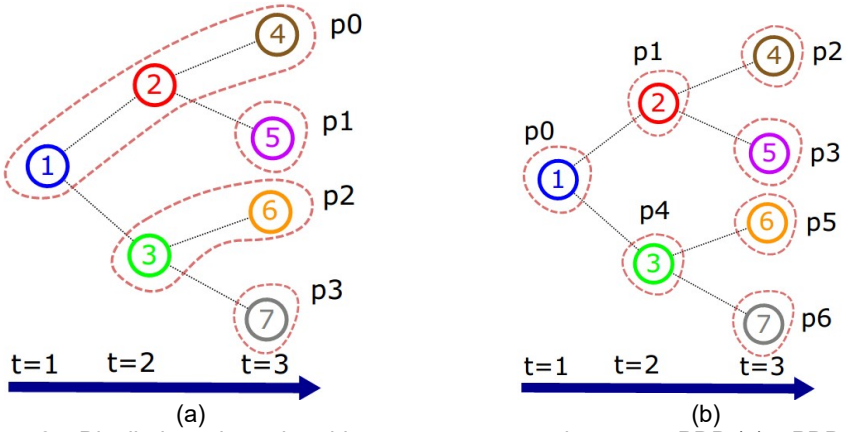


Figura 2 – Distribuição dos subproblemas nos processadores para PDD (a) e PDDA (b)

É possível notar que o tempo ocioso dos processadores pode ser considerável neste algoritmo, considerando que somente o processador *p0* está ativo em todos os estágios, além do mais, o tempo de resolução de cada estágio fica atrelado à resolução mais lenta dentre os subproblemas deste estágio, devido ao ponto de sincronismo entre-estágios.

4.0 - PDD ASSINCRONA

Devido as dificuldades apresentadas relacionadas a capacidade de se paralelizar o algoritmo da PDD, propõem-se neste trabalho uma metodologia denominada PDD Assíncrona (PDDA) que é naturalmente adequada para ambientes multiprocessados. Uma ideia similar foi apresentada em (9) para os problemas multi-estágios determinísticos, PDDD, onde os subproblemas foram resolvidos simultaneamente. Aqui, adapta-se esta ideia para

o caso estocástico com um conjunto de novas metodologias cujo objetivo é resolver de forma satisfatória os problemas estocásticos.

O algoritmo PDDA não executa passos *Forward* ou *Backward*, ou seja, as iterações não têm o mesmo formato das iterações da PDD tradicional. A PDDA itera através do que chamados *steps*, onde cada *step* é definido como uma resolução independente de todos os subproblemas da árvore de cenários. A troca de informações entre os subproblemas ocorre simultaneamente entre os nós no final de cada *step*: as variáveis de estado são transmitidas para os nós descendentes e os cortes são transmitidos para os nós ascendentes. Sendo assim, os nós podem ser resolvidos, um em cada processador, simultaneamente, e o ponto de sincronismo ocorre apenas no final de cada *step*, onde as informações entre os nós são trocadas.

A Figura 2(b), mostra a distribuição dos subproblemas entre os processadores na PDDA, para a árvore de cenários da Figura 1. A primeira diferença a ser notada em relação ao algoritmo paralelo tradicional é a quantidade de processadores que pode ser utilizada. No caso da PDDA a quantidade máxima de processadores é igual à quantidade de nós da árvore de cenários, onde cada processador é responsável por resolver um subproblema. No exemplo, são denominados sete processadores: p0 até p6. Primeiramente os problemas são inicializados em cada processador, assim, ainda sem nenhuma informação proveniente de outros nós, o primeiro *step* é iniciado. A primeira parte do *step* consiste na resolução simultânea de todos os problemas pelos processadores. Em seguida ocorre a transmissão de informação, onde os processadores transmitem os valores decididos como estados para os responsáveis dos nós descendentes, e o corte construído é transmitido para o processador responsável pelo nó ascendente. Também são transmitidas informações de custo para o processador p0, que por sua vez, com estas informações, realiza o teste de convergência, enviando um sinal de interrupção para os demais processadores em caso de convergência atingida, caso contrário um novo *step* é executado.

Apesar de proceder diferentemente da PDD tradicional, a PDDA possui as mesmas propriedades de convergência, o que garante que, sendo possível encontrar uma solução ótima para o problema utilizando a PDD, também o será com a PDDA. Isso ocorre pois, apesar de não realizar operação *Forward* explicitamente (operação utilizada para cálculo da convergência da PDD), pode-se extrair o passo Forward ao longo dos *steps* da PDDA.

5.0 - PDD PARCIALMENTE ASSINCRONA

A PDDA é capaz de resolver todos os subproblemas da árvore de cenários de forma completamente independente, o que possibilita paralelização da resolução de todos os nós. No entanto, quando a quantidade de processadores disponíveis é menor que a quantidade de nós na árvore de cenários, alguns nós podem compartilhar de um mesmo processador, dessa forma, estes nós serão resolvidos de forma sequencial, porém assíncrona.

Considerando isso, uma variante do método, denominada PDD Parcialmente Assíncrona (PDDPA) é apresentada. O objetivo deste método é introduzir intencionalmente um sincronismo entre subproblemas de um mesmo *step*, caso estes subproblemas estejam compartilhando o mesmo processador. Com isso, permite-se que as informações computadas pela resolução de um subproblema e disponíveis no processador sejam usadas na resolução do subproblema seguinte, ainda no mesmo *step*. É importante notar que, para problemas onde a quantidade de nós na árvore de cenários é igual a quantidade de processadores disponíveis, os métodos PDDA e PDDPA são idênticos. Conceitualmente pode-se dizer que a PDDPA tende à PDDA quando a quantidade de processadores disponíveis tende à quantidade de nós da árvore de cenários.

Para este algoritmo é necessário definir como os nós serão distribuídos entre os processadores, pois isso pode impactar na dinâmica do algoritmo. Neste trabalho, os nós foram agrupados em subárvores e estas divididas entre os processadores.

6.0 - RESULTADOS

Para avaliar as técnicas apresentadas foram construídos quatro estudos de caso com diferentes tamanhos e formatos de árvore de cenários. Assim o tempo computacional, *speedup* e eficiência foram medidos ao se resolver o problema utilizando diferentes quantidades processadores e os algoritmos:

- PDD paralela tradicional
- PDDA
- PDDPA

A análise e resultados obtidos estão expostos na sequencia.

6.1 Estudos de caso

Foi considerado o problema de planejamento hidrotérmico para parte do Sistema Interligado Nacional (SIN). No modelo são representadas 84 usinas hidráulicas organizadas em cascatas, onde 44 delas com capacidade de regularização, ou seja, a operação do reservatório impacta os problemas dos estágios seguintes, logo seus volumes iniciais são variáveis de estado para o problema. Também foram adicionadas ao modelo 46 usinas térmicas. Para avaliar a sensibilidade dos algoritmos com o tamanho e formato da árvore de cenários, quatro árvores diferentes foram construídas, conforme descrito nas Tabelas 1, 2, 3 e 4:

Tabela 1 – Estudo de caso 1: árvore de cenários

Número de nós:	127						
Número de estágios:	1	2	3	4	5	6	7
Cenários por estágio:	1	2	2	2	2	2	2

Tabela 2 – Estudo de caso 2: árvore de cenários

Número de nós:	781				
Número de estágios:	1	2	3	4	5
Cenários por estágio:	1	5	5	5	5

Tabela 3 – Estudo de caso 3: árvore de cenários

Número de nós:	306						
Número de estágios:	1	2	3	4	5	6	7
Cenários por estágio:	1	1	1	1	1	1	300

Tabela 4 – Estudo de caso 4: árvore de cenários

Número de nós:	221											
Número de estágios:	1	2	3	4	5	6	7	8	9	10	11	12
Cenários por estágio:	1	20	1	1	1	1	1	1	1	1	1	1

Alguns dos aspectos principais do modelo resolvido são:

- As usinas térmicas e hidráulicas foram representadas de forma individualizada.
- As afluências nas usinas, que representam as variáveis estocásticas do problema foram sinteticamente geradas por um modelo autorregressivo periódico (GEVAZP (11)), cujos parâmetros foram calculados com base em registros históricos.
- Foram considerados estágios mensais, exceto no estudo de caso 3, onde os estágios 1 ao 6 são semanais de forma a emular os estudos oficiais que definem os preços no sistema brasileiro.
- A curva de carga foi representada em três patamares: pesado, médio e leve, para cada estágio.
- Uma função de custo futuro multivariada foi considerada no final do horizonte de estudo, acoplando o modelo de médio prazo com o modelo de longo prazo, como apresentado em (12).
- A geração das usinas hidráulicas é calculada através de uma função linear por partes considerando o volume armazenado, a vazão turbinada e a vazão vertida da usina (13).

Cada subproblema de um nó na árvore de cenários tem cerca de 1200 variáveis e 2500 restrições, além dos cortes que são construídos e adicionados como restrições ao longo das iterações ou *steps*.

6.2 Características de hardware e software

Os três algoritmos avaliados foram implementados em linguagem Fortran e C++, em ambiente Linux. O paralelismo implementado é adequado para memória distribuída e passeado em passagem de mensagens, o padrão *Message Passing Interface* (MPI) foi usado para implementar os algoritmos paralelos. Os estudos de caso foram executados em cluster com vários nós contendo dois processadores AMD com seis cores cada, a memória RAM total de cada nó é de 96GBytes.

6.3 Resultados do estudo de caso 1

A Tabela 5 mostra o tempo de execução para resolver o problema do estudo de caso 1 para os três métodos variando a quantidade de processadores.

Tabela 5 – Estudo de caso 1: Tempo em segundos até a convergência de cada método.

	Número de Processadores										
Método	**1**	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
PDDPA	1897	337	249	177	110	85	84	73	65	71	70
PDDA	3563	408	302	175	147	135	103	104	101	93	89
PDD	1845	655	517	347	358	317	321	321	326	323	327

Analisando estes resultados podemos ressaltar os seguintes pontos:

- Enquanto o algoritmo da PDD monoprocessado leva em torno de 30 minutos para convergir, este tempo cai para cerca de 11 minutos quando utilizando 12 processadores e para cerca de 5 minutos com 60

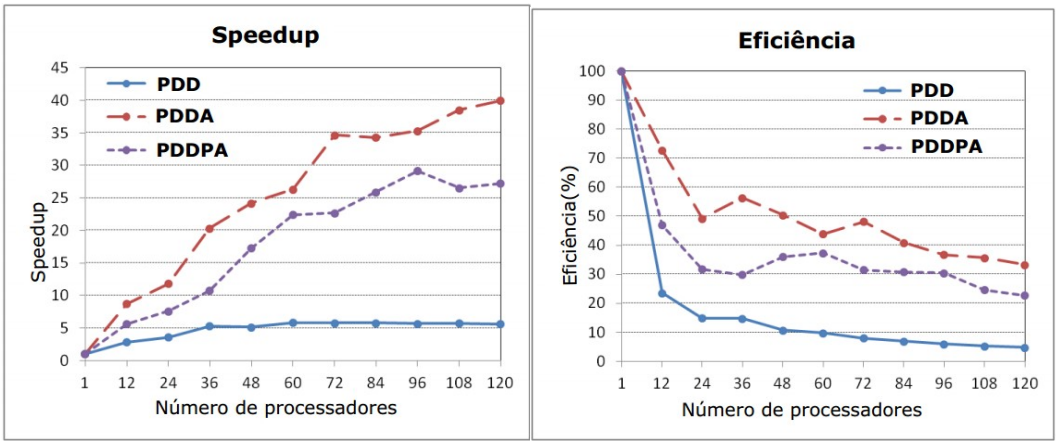
processadores. Embora observada redução considerável do tempo, vemos que ela não cresce linearmente com o aumento do número de processadores.

- Ainda no algoritmo da PDD ,pode-se perceber que há uma saturação, ou seja, não há mais redução considerável do tempo para uma quantidade de processadores maior que 60. Isso acontece devido ao formato da árvore de cenários, que tem 64 nós folha (cenários no último estágio), assim a PDD tradicional é capaz de paralelizar utilizando até no máximo 64 processadores.
- O tempo de execução sequencial da PDDA foi o dobro do tempo da PDD. No entanto ao aumentar a quantidade de processadores para 12, a PDDA já consegue um tempo de execução menor do que a PDD. Essa melhora é ainda mais evidente ao aumentar a quantidade de processadores. Com 120 processadores a PDDA leva um minuto e meio para finalizar sua execução. Isso evidencia que o algoritmo da PDDA é mais adaptável para ambientes multi-processados.
- No caso da PDDPA, o tempo sequencial ficou compatível com o da PDD e o tempo paralelo ficou melhor do que a PDDA, assim mostramos que o algoritmo parcialmente assíncrono é capaz de se adaptar ao ambiente em que está sendo executado obtendo sempre um bom desempenho.

Uma maneira de medir o desempenho de um algoritmo em paralelo é através do *speedup*, que é definido como a razão entre o tempo de execução usando um processador e o tempo usando N processadores ($speedup(N) = t_1/t_N$). O speedup ideal ocore quando $t_N = t_1/N$, ou $speedup(N) = N$. A efeciência ou performace paralela de um algoritmo ($\eta_{\%}$) é definido como o *speedup* dividido pelo número de processadores:

$$\eta_{\%}(N) = 100 \times \frac{t_1}{N \times t_N}$$

A Figura.3 mostra o *speedup* e eficiência do estudo de caso 1. Pode-se observar que ambos os algoritmos PDDA e PDDPA têm melhores valores de *speedup* e eficiência comparados a PDD, para qualquer quantidade de processadores. Como a taxa de convergência da PDDPA melhora quando a quantidade de processadores diminui (algoritmo adaptável), o *speedup* é pior que o método da PDDA, mesmo a PDDPA obtendo melhores tempos computacionais.



(a) (b)

Figura 3 – (a) *Speedup* e (b) Eficiência para o caso de estudo 1.

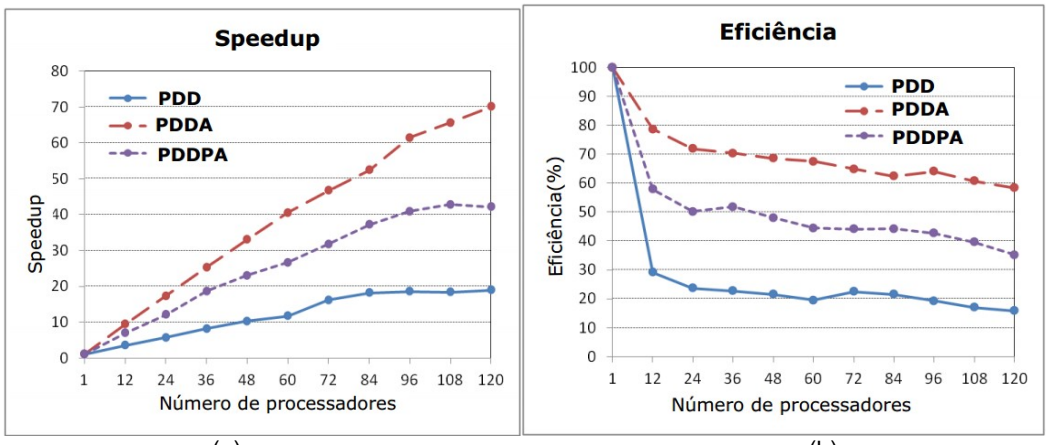
6.4 Resultados do estudo de caso 2

A Tabela 6 mostra o tempo de execução para resolver o problema do estudo de caso 2 para os três métodos variando a quantidade de processadores.

Tabela 6 – Estudo de caso 2: Tempo em segundos até a convergência de cada método.

	Número de Processadores										
Método	**1**	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
PDDPA	9944	1432	825	533	431	373	313	268	243	233	236
PDDA	19042	2017	1102	752	578	470	408	363	310	290	272
PDD	9536	2727	1676	1164	925	817	590	526	514	519	504

A Figura 4 mostra os resultados de *speedup* e eficiência para este caso. Enquanto a PDD tradicional tem eficiência entre 30 e 15%, a PDDA tem eficiência entre 80 e 60% ao se variar a quantidade de processadores, evidenciando a melhor adequação do método proposto ao ambiente paralelo.



(a) (b)

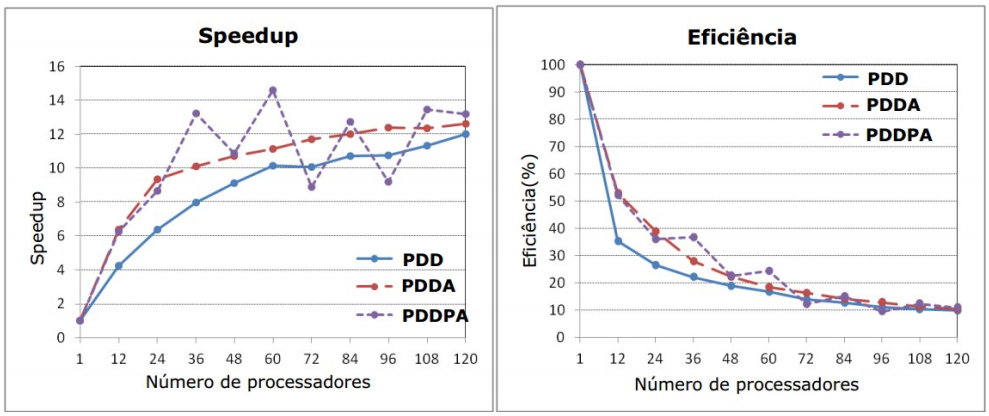
Figura 4 – (a) *Speedup* e (b) Eficiência para o caso de estudo 2

6.5 Resultados do estudo de caso 3

A Tabela 7 mostra o tempo de execução para resolver o problema do estudo de caso 3 para os três métodos, variando a quantidade de processadores. Neste caso, a PDDA não teve desempenho tão bom quanto os outros métodos, enquanto que a PDDPA e a PDD tiveram tempos compatíveis em todas as rodadas. O motivo da PDDA ter obtido tempos altos, para este tipo de árvore de cenários, foi a quantidade de *steps* necessários para convergência, o que fez com que o problema ficasse, a cada *step*, mais custoso e logo demandando mais tempo. A PDDPA por sua vez conseguiu melhorar a taxa de convergência o que proporcionou tempos menores. Ainda assim, observa na Figura 5 que as eficiências dos três métodos foram próximas, ou seja, o ganho de eficiência observado na PDDA e PDDPA em relação a PDD nos casos anteriores não foi observada neste caso.

Tabela 7 – Estudo de caso 3: Tempo em segundos até a convergência de cada método.

	Número de Processadores										
Método	**1**	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
PDDPA	6266	998	724	473	576	429	706	492	682	466	475
PDDA	13075	2053	1397	1293	1222	1175	1116	1088	1054	1060	1037
PDD	5132	1211	804	643	562	506	509	478	476	452	428



(a) (b)

Figura 5 – (a) *Speedup* e (b) Eficiência para o caso de estudo 3.

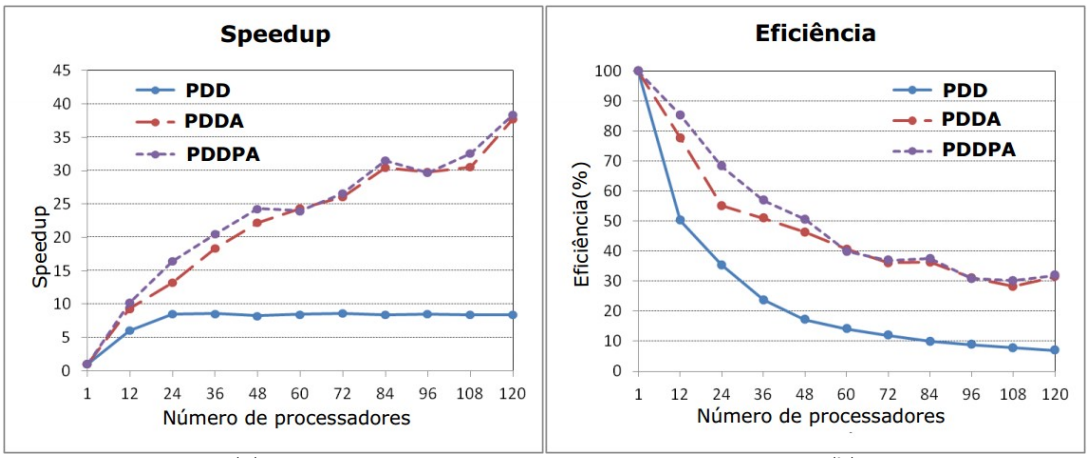
6.6 Resultados do estudo de caso 4

A Tabela 8 mostra o tempo de execução para resolver o problema do estudo de caso 4 para os três métodos variando a quantidade de processadores.

Tabela 8 – Estudo de caso 4: Tempo em segundos até a convergência de cada método.

	Número de Processadores										
Método	**1**	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
PDDPA	6566	641	400	320	270	274	247	208	221	202	171
PDDA	10652	1143	806	581	480	437	409	350	357	348	282
PDD	6179	1024	725	722	748	730	715	735	725	736	735

Neste caso, a árvore de cenários permite somente o uso de até 20 processadores no caso da PDD, o que pode ser observado no tempo de processamento que se satura em cerca de 700 segundos, a partir de 24 processadores. Enquanto isso os métodos da PDDA e PDDPA conseguem fazer uso de quantidades maiores de processadores, reduzindo o tempo até cerca de 200 segundos. A Figura 6 mostra os dados de *speedup* e eficiência para o caso.



(a) (b)

Figura 6 – (a) *Speedup* e (b) Eficiência para o caso de estudo 4.

7.0 - CONCLUSÕES

O problema de otimização do despacho de grandes sistemas hidrotérmicos é um desafio tanto do ponto de vista de aspectos de modelagem quanto de resolução. Quanto mais complexo o modelo, mais precisa é a representação de diversos aspectos do sistema, porem isso causa grandes dificuldades no processo de resolução do problema. Métodos de decomposição são comumente utilizados para problemas de grandes dimensões, dado que são capazes de encontrar uma solução global pra o problema resolvendo subproblemas menores de forma iterativa.

A programação dinâmica dual (PDD) é capaz de resolver problemas estocásticos multiestágio e é largamente utilizado em problemas de planejamento hidrotérmico. Este trabalho propõe dois algoritmos, baseados no método da PDD, melhor adaptáveis em ambientes multiprocessados, visando obter uma solução para o problema com menos tempo.

O primeiro algoritmo é a PDD Assíncrona (PDDA) que visa desacoplar os subproblemas da árvore de cenários resolvendo-os de forma independente e simultânea. Os resultados mostraram que a PDDA monoprocessada obtém tempo piores do que a PDD, no entanto, ao se utilizar de uma quantidade, grande ou pequena, de processadores, os resultados de tempo podem ser bem melhores do que o algoritmo paralelo tradicional da PDD. Também observou-se que este método obtêm os melhores resultados em termos de *speedup* e eficiência paralela.

O segundo algoritmo é uma adaptação do primeiro, quando a quantidade de processadores disponíveis é limitada. A PDD Parcialmente Assíncrona (PDDPA) tem o papel de se adaptar e resolver o problema da melhor forma, de acordo com os recursos disponíveis. Os resultados mostraram que este método obtém os melhores tempos de processamento, para qualquer quantidade de processadores.

Finalmente, ressalta-se a importância de se trabalhar com algoritmos que tenham bom desempenho paralelo e escalabilidade, o que pode permitir compensar aumento de complexidade da modelagem com aumento de recursos computacionais e sem prejuízos associados ao tempo de resolução do problema.

8.0 - REFERÊNCIAS BIBLIOGRÁFICAS

(1) Fortunato, L. A. M., Neto, T. A. A., Albuquerque, J. C. R., Pereira, M. V. F., “Introdução ao planejamento da expansão e operação de sistemas de produção de energia elétrica”, Universidade Federal Fluminense (1990).

(2) Birge, J. R. “Decomposition and partitioning methods for multistage stochastic linear programs”. Operations Research, 33(5):989-1007 (1985).

(3) R. T. Rockafellar and R. J.-B. Wets, “Scenarios and policy aggregation in optimization under certainty,” Mathematics of Operations Research, vol. 16, no. 1, pp. 119–147, 1991.

(4) N. Growe-Kuska, K. C. Kiwiel, M. P. Nowak, W. Romisch, and I. Wegner, “Power management in a hydro-thermal system under uncertainty by lagrangian relaxation,” Humboldt-Universitat, Tech. Rep., 1999

(5) J. R. Birge, C. J. Donohue, D. F. Holmes, and O. G. Svintsitski, “A parallel implementation of the nested

decomposition algorithm for multistage stochastic linear programs," Mathematical Programming, vol. 75, no. 2, pp. 327–352, 1996.

(6) M. A. H. Dempster and R. T. Thompson, "Parallelization and aggregation of nested Benders decomposition," Annals of Operations Research, vol. 81, no. 0, pp. 163–188, 1998.

(7) J. Linderoth and S. Wright, "Decomposition algorithms for stochastic programming on a computational grid," Computational Optimization and Applications, vol. 24, no. 2, pp. 207–250, 2003.

(8) Moritsch, H. W., Pflug, G. C., Siomak, M., 2001, "Asynchronous nested optimization algorithms and their parallel implementation", Wuhan University Journal of Natural Sciences, v. 6, n. 1 (Mar), pp. 560– 567. ISSN: 1993-4998. doi: 10.1007/BF03160302

(9) T. N. Santos, A. L. Diniz, and C. L. T. Borges, "A new nested Benders decomposition strategy for parallel processing applied to the hydrothermal scheduling problem," IEEE Transactions on Smart Grid, vol. 8, no. 3, pp. 1504–1512, May 2017.

(10) Diniz, A. D., Costa, F. S., Macira, M. E., Santos, T. N., Brandão, L. C., Cabral, R. N. (2018). "Short/Mid-Term Hydrothermal Dispatch and Spot Pricing for Large-Scale Systems-the Case of Brazil". *20th Power Systems Computation Conference* 1-7. 10.23919/PSCC.2018.8442897.

(11) Jardim, D.L.D.D., Maceira, M. E. P., Falcao, D. M., 2001, "Stochastic streamflow model for hydroelectric systems using clustering techniques". In: 2001 IEEE Porto Power Tech Proceedings (Cat. No.01EX502), v. 3, p. 6. doi: 10.1109/PTC.2001.964916

(12) Maceira, M.E.P., Duarte, V.S., Penna, D.D.J., Moraes, L.A.M., Melo, A.C.G., "Ten years of application of stochastic dual dynamic Programming in official and agent studies in Brazil –Description of the NEWAVE program", *16th Power Systems Computation Conference - PSCC*, Glasgow, SCO (2008).

(13) Diniz, A.L., Maceira, M.E.P., "A four-dimensional model of hydro generation for the short-term hydrothermal dispatch problem considering head and spillage effects", *IEEE Trans. Power Syst.,* v. 23, n.3, pp. 1298-1308 (2008).

9.0 - DADOS BIOGRÁFICOS

Lílian C. Brandão: possui graduação em Engenharia Elétrica (2014) pela UFMG, MG, Brasil. Diploma de engenheiro nível master (2013) pela *Ecole Supérieure d'Ingé*nieurs en Electronique et Electrotechnique, Paris, França. Mestrado em Engenharia de Sistemas e Computação (2018) pela UFRJ / COPPE, RJ, Brasil. É pesquisadora no CEPEL - Centro de Pesquisas de Energia Elétrica, do grupo ELETROBRÁS, desde 2014.

André L. Diniz: possui graduação em Engenharia Civil (1997), mestrado em Engenharia de Transportes (2000) e doutorado em Otimização pelo Programa de Engenharia de Sistemas e Computação (2007), todos pela UFRJ / COPPE. Em 2014 realizou um pós-doutorado no Weierstrass Institute for Applied Analysis and Stochastics. É pesquisador e chefe do Departamento de Otimização Energética e Meio Ambiente (DEA) do CEPEL - Centro de Pesquisas de Energia Elétrica, do grupo ELETROBRÁS, e professor adjunto da UERJ - Universidade do Estado do Rio de Janeiro, no departamento de Estatística. Tem grande experiência na área de programação matemática e otimização, especialmente aplicada em problemas de planejamento e operação de Sistemas de Energia Elétrica.