ALGORITMO DE PROPAGAÇÃO DE CRENÇA GENERALIZADO PARALELO - UMA APLICAÇÃO EM PROCESSAMENTO DE IMAGENS

Robson Lorbieski, Reginaldo Zara, Josué P. Castro, Rodolfo Lorbieski, André Luiz Brun, e-mail: robson.lorbieski@gmail.com

Universidade Estadual do Oeste do Paraná/Centro de Ciências Exatas e Tecnológicas – Cascavel – PR

Palavras-chave: Propagação de Crenças, Programação Paralela, Restauração de Imagens.

Resumo:

Este trabalho utilizou o algoritmo Generalizado de Propagação de Crença (Generalized Belief Propagation - GBP) como ferramenta para restauração de imagens danificadas, sendo este implementado de forma seqüencial e paralela. O trabalho apresenta um estudo comparativo entre as versões seqüencial e paralela. Para a versão paralela foram realizados testes em um cluster, utilizando-se o MPI como interface padrão de comunicação. O algoritmo GBP envolve inferência estatística e o cálculo de probabilidades marginais requerendo grande processamento de dados. Neste algoritmo, que foi realizado de forma iterativa, o valor da variável aleatória associada a um nodo do grafo, é calculado combinando valores observados com mensagens trocadas entre nodos vizinhos. Verificou-se uma redução de tempo de processamento do algoritmo proporcional ao número de processadores alocados à sua execução, mostrando assim que o algoritmo é fortemente paralelizável, principalmente em situações em que o número de pixels em cada unidade de processamento não seja muito pequeno ao ponto de gerar overhead pela troca de mensagens.

Introdução

Algoritmos de trocas de mensagens, por serem iterativos, em geral consomem muito tempo de processamento. O algoritmo GBP, que é um algoritmo que já possui a comprovação de um bom resultado quanto à convergência quando implementado de forma sequencial, pode acarretar em gastos de tempo de processamento consideráveis. Dessa forma, a proposta de uma implementação paralela visou encontrar uma forma de reduzir o tempo total de convergência do processo.

O algoritmo Generalizado de Propagação de Crença (Generalized Belief Propagation - GBP), proposto por Yedidia e colaboradores, é uma evolução do algoritmo de Propagação de Crença (Belief Propagation - BP), proposto por Pearl e colaboradores.

Ambos os algoritmos de propagação de crença, BP e GBP, são utilizados para investigar inferência estatística em modelos gráficos. No algoritmo BP, o valor da variável aleatória de cada vértice de determinado

grafo, é calculado através da combinação de valores observados, chamados de evidência local, com os valores das mensagens que o vértice recebe de seus vértices (nós) vizinhos. O GBP por ser uma evolução do BP, incorpora o mesmo conceito de inferência de vértices, além disso, introduz o conceito de regiões, onde cada região recebe mensagens de regiões vizinhas.

Por ser relativamente novo e pouco conhecido, muitas das propriedades de convergência e critérios de aplicabilidade a diferentes tipos de problemas do algoritmo GBP ainda estão em estudo, dispondo assim pouca literatura especializada, sendo esta encontrada dispersa em diferentes artigos, o que acaba dificultando sua divulgação, bem como o reconhecimento do algoritmo como um método útil na solução de problemas que envolvam inferência estatística.

O algoritmo GBP consiste basicamente em cinco conceitos: Evidência, Interação, Troca de Mensagens, Crença e Regiões de vértices. Uma evidência é um fato ocorrido ou alguma informação disponível à respeito de uma hipótese. É utilizada na inferência de uma outra informação. Neste trabalho, os valores das evidências locais usam informações referentes aos estados (cores) possíveis de cada pixel. A interação é o fato que ocorre quando dois vértices interagem entre si.

A mensagem que um vértice envia ao outro vértice é denotada por $m_j^i(x_j)$, e pode ser descrita como uma mensagem que um vértice i envia para um vértice j, informando sobre qual estado o vértice j deveria estar. Essa definição pode ser aplicada também a regiões de vértices. A mensagem $m_j^i(x_j)$ é um vetor com dimensão de x_j, onde cada elemento corresponde ao valor de probabilidade que *i*, o emissor da mensagem, "acredita" que *j*, o receptor da mensagem, terá em um estado futuro. A Figura 1 exibe o funcionamento do processo de troca de mensagens entre os vértices.

O conceito de Crença (belief) é dado pelo valor que um vértice *i* associado a um grafo, "acredita" que possa assumir um valor x_i, denota-se $b_i(x_i)$, ou seja, a crença de que a variável aleatória X_i associada ao vértice *i*, seja x_i. No algoritmo BP a crença para um vértice i é proporcional ao produto das evidências locais do vértice com todas as mensagens que chegam ao vértice *i*.

A principal diferença entre algoritmo GBP e algoritmo BP está na idéia de conjuntos ou regiões de vértices que enviam mensagens para outras regiões de vértices. Cada região recebe mensagens de regiões externas a ela, sendo que os vértices que compartilham a mesma aresta de determinada região compartilham também a crença daquela aresta. Conforme mostra a Figura 2, o vértice *k* recebe mensagens dos vértices abaixo dele e a sua esquerda, compartilhando assim, com *i*, o valor que recebe da região da esquerda e com *l* o valor que recebe da região de baixo.

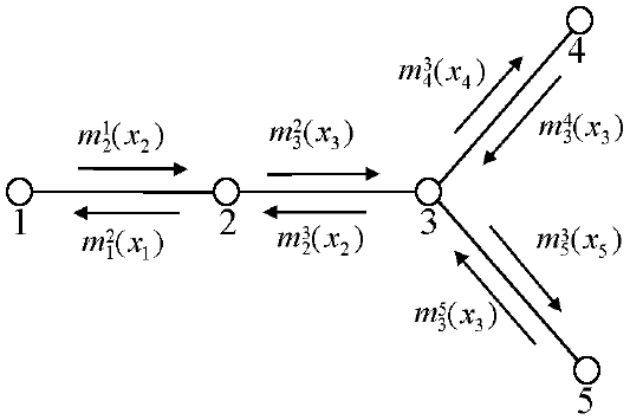

Figura 1. Troca de mensagens entre vértices.

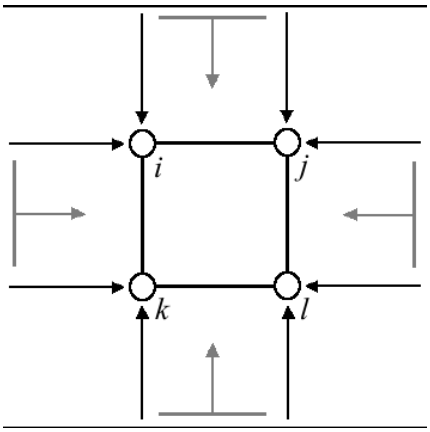

Figura 2. Ilustração do cálculo de crença de uma região formada por uma plaqueta de quatro vértices.

Uma aplicação do GBP sequencial, em uma linguagem interpretada, para restauração de imagens foi desenvolvida por Zara. Tal implementação confirmou a aplicação do GBP como método de melhoria em imagens com presença de ruídos.

Ruídos em imagens digitalizadas são pequenas características que atrapalham uma boa visualização da imagem, podendo ser originadas na aquisição da imagem, por exemplo, devido a características do sensor que capturou a imagem, ou ainda devido a diversos aspectos, como movimento de objetos em cena, fatores de iluminação, entre outros. Sendo necessário a aplicação de técnicas para a remoção dos ruídos, uma das técnicas possíveis é a utilização de métodos de filtragem.

Um ruído bastante conhecido é o Ruído Sal e Pimenta, também chamado de ruído Chuvisco. Ele se caracteriza pelo aparecimento de pixels de cor clara em partes escuras da imagem e pontos de cor escura em partes claras da imagem. Geralmente ocorre devido a problemas de transmissão de dados.

Materiais e Métodos

Inicialmente foram agrupadas várias imagens, em tons de cinza, isentas da presença ruídos. Em seguida todas foram poluídas com ruído do tipo sal e pimenta visando testar a eficiência do método quanto à qualidade da remoção dos ruídos. A intensidade de poluição pelo ruído foi dada por uma probabilidade definida pelo projetista.

A complexidade deste algoritmo, segundo Bosa, é $O(pc^4)$, onde p representa o número de pixels que compõe a imagem e c a quantidade de cores. A escolha de imagens com 256 tons de cinza se deve ao fato da simplificação do processo, visto que os três canais (RGB), componentes das cores, têm a mesma intensidade. Caso fosse trabalhado com imagens coloridas, cada canal deveria ser tratado separadamente. Além disso, se fossem utilizadas mais cores, o custo em relação ao tempo para a obtenção dos resultados seria demasiadamente longo devido à ordem de execução do algoritmo.

Para a medição da qualidade das imagens obtidas foi utilizada a métrica do Erro Quadrático Médio (MSE – Mean Square Error). Tal critério compara o valor da imagem original, antes de ser poluída com o ruído, com a imagem final, onde houve a incidência e remoção dos ruídos, de forma que, quanto menor o valor obtido pelo MSE, melhor a qualidade da imagem final.

Para o desenvolvimento do trabalho cada pixel foi associado como sendo um vértice e toda imagem como sendo o grafo, dessa forma, a inferência dos pixels toma como base os estados dos pixels vizinhos e as mensagens trocadas entre as diferentes regiões da imagem, usando como evidência local as cores possíveis em cada pixel e as interações entre pares.

Por tratar-se de um algoritmo de troca de mensagens, as bordas externas da imagem não recebem mensagens de pixels vizinhos que estariam externos à imagem. Para contornar este problema foi estipulado um valor padrão para troca de mensagens destes pixels externos.

A paralelização de um algoritmo se justifica quando este requer um processamento muito grande para ser usado de forma seqüencial, no caso deste trabalho, a paralelização do algoritmo se justifica quando a imagem for grande e/ou principalmente quando houver muitas cores (estados possíveis) na imagem.

Para trabalhar com a simulação paralela foi utilizado um cluster do tipo Beowulf para processar os dados, através do padrão de interface para a troca de mensagens em máquinas paralela MPI. Os clusters Beowulf são resultados de um projeto iniciado pela NASA (National Aeronautics and Space Administration) em 1994 no centro de pesquisas CESDIS (Center of Excellence in Space Data and Information Sciences), como parte do projecto Earth and Space Sciences, cujo objetivo primário consistia em determinar a aplicabilidade de arquiteturas de processamento paralelo a problemas do domínio das ciências espaciais, para o tratamento de dados recolhidos por satélite, a preços acessíveis. A designação Beowulf se da pelo nome de um

herói lendário da literatura épica medieval inglesa, que foi escolhido pelos criadores desta arquitetura.

MPI é um padrão de interface para a troca de mensagens em máquinas paralelas com memória distribuída. Uma aplicação MPI é constituída por um ou mais processos que se comunicam através de funções para o envio e recebimento de mensagens entre os processos.

Para a distribuição da imagem para cada nodo do cluster, fez-se necessário a partição da imagem, a qual se viu facilitada pelo fato de se conhecer previamente o tamanho total da mesma e também por não haver bordas irregulares.

Na implementação deste trabalho, no que se refere à distribuição imagem entre os nodos do cluster, foi utilizada uma divisão vertical da imagem, na qual cada processo fica responsável por uma parte dessa divisão ou faixa, sendo o tamanho de cada faixa definido como a parte inteira da divisão da largura da imagem pelo número de processos, onde o último processo recebeu, além desse valor, o resto dessa divisão, caso haja. Entretanto, a paralelização criou novas sub-imagens que, consequentemente, geraram mais bordas extras. Para resolver este problema criou-se uma largura (definida por número de pixels) extra-borda, no qual pode-se ter três casos específicos para a distribuição desta largura extra, como pode ser visto na Figura 3:

• faixa da extremidade esquerda que além da área de sua faixa, terá atribuído um espaço equivalente ao tamanho desta largura extra, sendo este tamanho extra alocado à direita desta faixa;

• faixa da extremidade direita que além da área de sua faixa, terá atribuído um espaço equivalente ao tamanho desta largura extra, sendo este tamanho extra alocado à esquerda desta faixa;

• faixas internas que além da área de sua respectiva faixa, terá atribuído um espaço equivalente ao tamanho desta largura extra, sendo este tamanho extra alocado à esquerda e a direita de cada faixa.

LVC	1	2	3	4		5		6	7	8		9		10	11	12		13		14	15	16		17		18	19	20	21
1	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
2	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
3	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
4	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
5	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
6	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
7	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
8	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
9	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
10	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
11	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
12	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
13	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
14	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
15	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
16	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
17	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
18	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
19	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
20	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
21	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
22	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
23	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
24	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4
25	0	0	0	0	1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	4

Figura 3. Exemplo de imagem com partições verticais.

Esse procedimento de distribuição extra entre as faixas evita a troca de mensagens entre as bordas, sendo necessário apenas o envio das sub-imagens processadas para a centralização (junção) dos dados. Ressalta-se que o tamanho da matriz a ser enviada ao processo responsável pela junção é igual ao tamanho de cada faixa, ou seja, nesse processo os valores extra-borda são ignorados, já que sua função é possibilitar a inferência dos vértices das bordas das faixas.

O algoritmo paralelo assemelha-se muito ao algoritmo seqüencial, onde cada processo trabalha de forma semelhante a esse algoritmo, diferenciando-se apenas na parte da escolha da crença, visto que cada processo envia sua sub-imagem ao processo que fará a junção. Apesar do uso desse valor extra em cada faixa implicar em cálculos redundantes, a não obrigatoriedade de transmissão de dados relativos à borda, assim como a não necessidade do tratamento dessas mesmas sub-imagens, acaba compensando o custo, além do que, caso não houvesse essa redundância seria necessário uma sincronização entre os processos, já que caso os valores requeridos por uma determinada borda ainda não estejam calculados, seria necessário aguardar os cálculos dos mesmos. Conclui-se então que o não uso de bordas compartilhadas acabaria aumentando a complexidade do desenvolvimento além de possibilitar um aumento no overhead das comunicações entre os nodos do cluster.

Para cada conjunto de testes deste trabalho foram utilizados 15 iterações até o término da execução do mesmo.

Os conjuntos de testes foram submetidos a um cluster composto de 16 máquinas Pentium 4 HT com freqüência de 2,99 GHz. Nos testes do algoritmo seqüencial executados foi utilizado um nodo do cluster para a execução do algoritmo (para que o processo seqüencial trabalhasse em uma máquina com a mesma configuração que os testes paralelos).

Resultados e Discussão

As imagens da Figura 4 têm dimensões de 195×126 pixels, sendo compostas por 15 tons de cinza cada uma, tendo um custo relativamente alto por tratar-se de uma imagem com tamanho considerável e pelo número de cores. Foi aplicado um ruído $\eta = 0.10$, ou seja, 10% de ruído sobre a imagem original. A Tabela 1 mostra o tempo de execução de cada iteração para o algoritmo seqüencial e para o algoritmo paralelo com diversos processos, ao executar os cálculos do conjunto de testes da Figura 4. Nessa tabela pode-se ver que houve vantagem em usar esse conjunto de testes de forma paralela, já que houve um aumento da velocidade proporcional ao número de processos utilizados, com exceção da execução seqüencial do algoritmo.

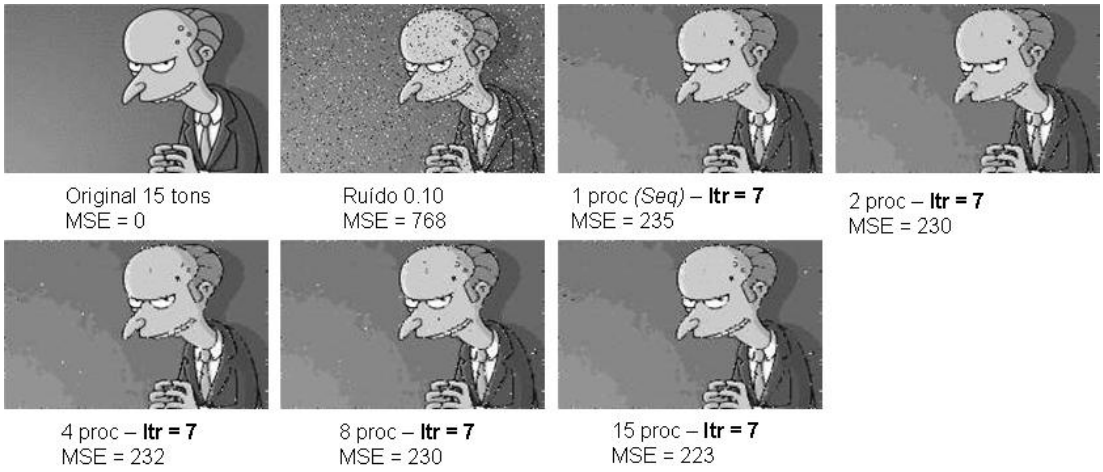

Figura 4. Seqüência de figuras da 7ª iteração, para diferentes números de processos.

O gráfico das Figuras 5 e 6 também mostram o comportamento do algoritmo e seus múltiplos processos em função do tempo. A Figura 5 mostra o comportamento do algoritmo em função do tempo para convergência do resultado, nela pode-se observar que houve considerável discrepância de comportamento, onde o método seqüencial foi bastante inferior às abordagens paralelas.

Tabela 1 – Tempo de execução de cada iteração da Figura 4.

Iteração	1 prc (*Seq*)	2 prcs	4 prcs	8 prcs	15 prcs
0	757s	76s	46s	31s	21s
1	757s	73s	38s	21s	13s
2	767s	74s	39s	24s	13s
3	868s	74s	39s	21s	14s
4	945s	74s	39s	21s	13s
5	1058s	79s	39s	21s	13s
6	1265s	78s	42s	22s	14s
7	1366s	82s	43s	23s	14s
8	1465s	85s	46s	24s	15s
9	1575s	95s	50s	27s	18s
10	1698s	110s	59s	29s	20s
11	1787s	122s	68s	35s	23s
12	1821s	129s	74s	38s	26s
13	1903s	137s	79s	42s	26s
14	1950s	144s	84s	44s	28s
Total Real	340m52s	27m52s	15m9s	8m12s	5m16s

Na Figura 6 são exibidos apenas as execuções com processos paralelos, visando verificar a análise do comportamento temporal do algoritmo. Pode-se observar que não houve um overhead e que há a certa linearidade em relação ao número de processos utilizados, ou seja, a utilização de muitos processos se faz necessária quando a imagem exigir muito processamento, tendo assim um comportamento desejado.

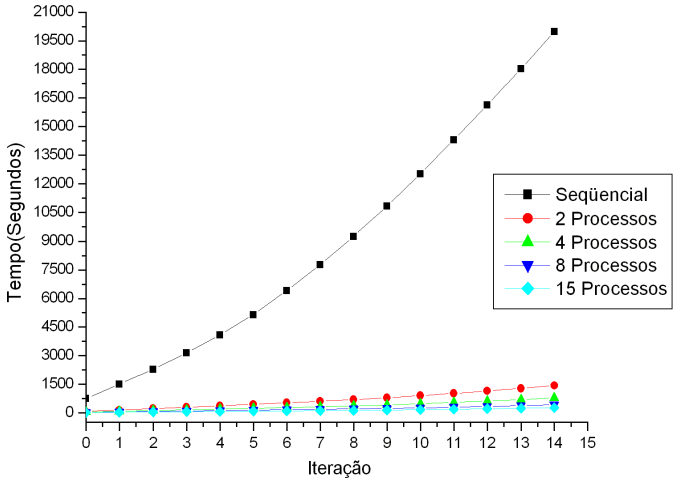

Figura 5 –Tempo de execução das imagens da Figura 4.

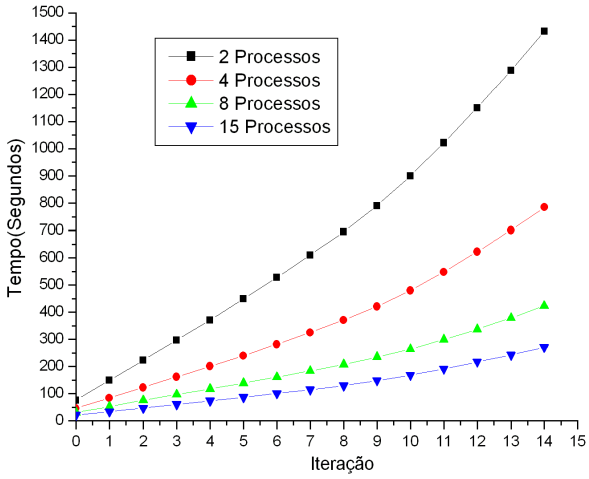

Figura 6 - Tempo de execução das imagens da Figura 4 considerando somente os processos paralelos.

Verificou-se o algoritmo seqüencial não manteve a mesma proporcionalidade de tempo em relação às execuções paralelas, visto que todos os cálculos, tanto os das iterações quanto os relativos à construção da rede, são concentrados em um único processo de forma seqüencial, ao contrário do algoritmo paralelo, no qual não somente o cálculo das iterações, mas também todos os outros anteriores necessários para o algoritmo também são divididos, havendo apenas a junção para a escolha da crença, junção essa que ocorre somente no fim de cada iteração, o que leva o algoritmo paralelo a ter uma vantagem muito maior frente ao seqüencial, mesmo que somente sejam usados dois processos para a execução

paralela, já que com o uso de mais processos há, entre outras coisas, uma redução de carga na memória de cada máquina.

O algoritmo paralelo manteve o mesmo comportamento do algoritmo seqüencial com respeito à convergência dos dados, inclusive obtendo os melhores resultados nas mesmas iterações e com valores de MSE parecidos, sendo que a implementação paralela também manteve o mesmo padrão de evolução do tempo de execução, caso a imagem requeresse grande processamento, com ganhos de velocidade proporcionais ao número de processos, desde que esse número seja menor ou igual ao de processadores físicos no cluster.

Conclusões

O estudo realizado nesse trabalho teve como objetivo a diminuição de tempo para a obtenção de resultados que já se sabiam aceitáveis no que se referia a convergência. Duas soluções foram propostas para esse trabalho: a implementação seqüencial compilada de um algoritmo baseada em um algoritmo seqüencial interpretado já existente e a implementação paralela do mesmo algoritmo.

Nos testes realizados com a implementação paralela observou-se uma redução de tempo de processamento do algoritmo proporcional ao número de processadores alocados à sua execução. Verificou-se também uma relação estreita entre o desempenho do algoritmo e a largura das faixas distribuídas a cada processador, quando esta largura for pequena ocorrerá um aumento de overhead no sistema, acarretando uma queda no desempenho.

Comparando-se o desempenho da implementação seqüencial com o da implementação paralela, observou-se que o desempenho paralelo foi bastante superior ao seqüencial, mesmo em casos de overhead. A discrepância entre as abordagens é ainda mais acentuada em casos em que as imagens de teste apresentavam tamanho maior.

Sugere-se para trabalhos futuros a realização de testes de exaustão visando determinar qual a relação entre largura de faixa e número de processadores alocados. Pode-se também utilizar o algoritmo para a remoção de outros tipos de ruídos, por exemplo, a remoção de ruídos de áudio digital.

Referências

Bosa, J.L. Algoritmo Generalizado de Propagação de Crença – Monografia de Graduação, Universidade Estadual do Oeste do Paraná, 2006.

Velho, L.; Gomes, J. *Computação Gráfica: Imagem*, Ed: IMPA. 1994; Vol.1, 421.

Filho, V. J. M. F.; Ignácio, A. A. V. *Mpi: Uma ferramenta para implementação paralela*, E-book. 2002.
Lorbieski, R. Algoritmo de propagação de crença generalizado paralelo - Uma aplicação em processamento de imagens – Monografia de Graduação, Universidade Estadual do Oeste do Paraná, 2007.
Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Ed: Morgan Kaufmann Publishers. 1988.
Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Ed: Morgan Kaufmann Publishers. 1988.
Yedidia, J. S.; Freeman, W. T.; Weiss, Y. Bethe free energy, kikuchi approximations, and belief propagation algorithms. *Mearl:* Mitsubishi Electric Research Laboratory, 2001.
Yedidia, J. S.; Freeman, W. T.; Weiss, Y *Generalized belief propagation*. *Mearl:* Mitsubishi Electric Research Laboratory, 2000.
Yedidia, J. S.; Freeman, W. T.; Weiss, Y *Understanding belief propagation and its generalizations*. *Mearl:* Mitsubishi Electric Research Laboratory, 2002.
R. A. Zara; J. L. Bosa. Aplicação do Algoritmo Generalizado de Propagação de Crenças à Restauração de Imagens In Encontro Paranaense de Computação, Cascavel, Paraná, 2007.

.