TÉCNICAS PARA ALOCAÇÃO ESTÁTICA DE TAREFAS EM SISTEMAS DISTRIBUÍDOS

H.K. Huang, V.C. Barbosa
Programa de Sistemas, COPPE
Universidade Federal do Rio de Janeiro
Caixa Postal 68511 - 21945 - Rio de Janeiro - RJ

RESUMO

Neste trabalho enfocamos o problema de alocação estática de tarefas em sistemas distribuídos nos quais processadores podem ter diferentes velocidades de processamento e canais de comunicação podem ter diferentes capacidades. A partir da definição de uma função para medir o custo de uma alocação, duas abordagens são propostas. A primeira usa o algoritmo A^* e é semelhante a algumas técnicas já propostas anteriormente. Foram criadas uma nova heurística a partir da função de custo e formas de melhorar o desempenho do método. O segundo método proposto, utiliza a idéia de uma rede neuronal, segundo o trabalho de Hopfield e Tank. Os testes realizados permitem conluir que os dois métodos são comparáveis. A rede neuronal, porém, tem a grande vantagem de poder facilmente ser implementada em paralelo.

ABSTRACT

In this work we address the problem of static task allocation in distributed systems where processors may have different processing speeds and communication channels may have different capacities. From the definitions of a cost function to evaluate an allocation, two approaches are proposed.The first one employs the A^* algorithm, and is similar to some techniques previously proposed. A new heuristic was created from the cost function as well as new alternatives to improve the performance of the method. The second technique utilizes the idea of a neural network, following the work of Hopfield and Tank. We have coducted tests which allow us to conclude that both methods perform comparably. Nevertheless, the inherent distributed appeal of the neural network favors it greatly.

1. Introdução

O rápido desenvolvimento e difusão de processadores de baixo custo e tecnologias de interconexão gerou um grande interesse em sistemas distribuídos para processamento paralelo, conforme atesta a grande multiplicidade desses sistemas no mercado.

Uma das funções mais críticas quanto ao desempenho desses sistemas é a alocação de tarefas a processadores, conhecida como balanceamento estático de carga.

Neste trabalho, nossa abordagem ao pro-

blema é grafo-teórica, no sentido em que o vemos como um problema de isomorfismo de grafos, portanto computacionalmente intratável por métodos exatos [1], mesmo para sistemas com uma estrutura simples como um hipercubo [2].

Para resolver o problema, empregamos dois métodos. O primeiro é o método de busca A^*, com diversas variações quanto a abordagens similares encontradas na literatura. O segundo é a utilização de uma rede neuronal, segundo o modelo de Hopfield e Tank [3].

Ambos os métodos fornecem resultados muito satisfatórios, e de fato comparáveis entre si, tanto em eficiência (em uma implementação centralizada), quanto na qualidade da solução. Todavia, como a rede neuronal oferece a possibilidade inerente de uma implementação distribuída, os resultados obtidos com ela são especialmente encorajadores.

Na Seção 2 apresentamos a descrição do modelo do problema que utilizamos, fazendo uma comparação com trabalhos anteriores. A Seção 3 apresenta as particularidades do algoritmo A^* usado para resolver o problema de alocação de tarefas e detalhes de implementação. A Seção 4 contém a descrição, modelagem e implementação de um algoritmo utilizando o princípio das redes neuronais. Por fim, na Seção 5, apresentamos os resultados obtidos. É feita uma análise desses resultados, estudando o comportamento dos algoritmos e fazendo uma comparação entre as alocações obtidas pelos dois métodos. Considerações finais estão no Seção 6.

2. Alocação de Tarefas em Sistemas Distribuídos

Para que um sistema distribuído tenha um bom desempenho, é necessário que ele seja bastante eficiente no gerenciamento de recursos e tarefas. Um ponto importante é a estratégia de distribuição de carga entre os múltiplos processadores, ou seja, o gerenciamento dos recursos de processamento e comunicação do sistema.

Dois dos principais objetivos de uma política de alocação de tarefas são:

- Evitar a degradação do sistema causada pela sua saturação em consequência do excesso de comunicação entre tarefas alocadas a processadores diferentes.
- Evitar a concentração de muitas tarefas num só processador, sobrecarregando-o. Deve-se procurar uma distribuição uniforme das tarefas.

Estes são objetivos conflitantes, cabendo ao algoritmo a tarefa de tentar satisfazê-los da melhor maneira possível.

O problema de alocação estática de tarefas pode ser formulado como um problema de isomorfismo de grafos, com o objetivo de minimizar uma função de custo.

O primeiro grafo envolvido no problema é o grafo conexo não direcionado $G_P = (N_P, E_P)$, onde N_P é o conjunto dos vértices que representam os processadores e cada arco do conjunto E_P representa um canal de comunicação bidirecional entre dois processadores.

A cada processador p em N_P será associada uma velocidade relativa de processamento representada por s_p. A cada canal de comunicação entre os processadores p e q em E_P, está associada uma capacidade relativa do canal, dada por $c_{(p,q)}$. Supomos que a capacidade é a mesma em ambas as direções, isto é, $c_{(p,q)} = c_{(q,p)}$. Caso p e q não se comuniquem, tem-se $c_{(p,q)} = c_{(q,p)} = 0$. Pode-se notar que s_p/s_q indica quão mais rápido o processador p é em relação ao processador q. O mesmo se aplica aos canais de comunicação. Associada ao grafo G_P existe uma estrutura

de roteamento fixa e conhecida a priori. Denotamos o conjunto de arcos na rota de p e q por $R(p,q)$.

O outro grafo conexo não direcionado envolvido no problema é o grafo de tarefas $G_T = (N_T, E_T)$. Nele os vértices do conjunto N_T representam as tarefas e os arcos do conjunto E_T são as necessidades de comunicação entre tarefas.

A cada tarefa i em N_T está associada uma demanda relativa de processamento, representada por σ_i. A demanda relativa de comunicação entre as tarefas i e j é dada por $\gamma_{(i,j)}$, possivelmente diferente de $\gamma_{(j,i)}$.

Uma alocação de tarefas aos processadores é dada pelo mapeamento $A : N_T \rightarrow N_P$ onde $A(i) = p$ se e somente se a tarefa i está alocada ao processador p.

A carga total de uma alocação devida ao processamento num processador p é dada por

$$PL_p = \frac{1}{\vartheta_p} \sum_{i|A(i)=p} \sigma_i. \quad (1)$$

A carga total de comunicação num canal (p,q) é dada por

$$CL_{(p,q)} = \frac{1}{c_{(p,q)}} \sum_{i \neq j | (p,q) \in R(A(i),A(j))} \gamma_{i,j}. \quad (2)$$

Dada uma estrututra de roteamento $R(p,q)$, o custo de uma alocação A pode ser dado por uma função $H(A)$ [4], composta pela soma de uma parcela de processamento $H_P(A)$ e uma parcela de comunicação $H_C(A)$. Temos então

$$H_C(A) = \sum_{(p,q)} CL_{(p,q)}, \quad (3)$$

e

$$H_P(A) = \sum_p \sum_{i,j|A(i)=A(j)=p} g(\sigma_i, \sigma_j). \quad (4)$$

Aqui g é uma função que encoraja a distribuição de carga entre os processadores.

Sinclair [5] faz com que o termo $H_P(A)$ seja um simples somatório dos custos de cada processador devido ao processamento. Esta função não é boa para o nosso modelo, uma vez que o custo mínimo sempre ocorrerá quando todas as tarefas forem alocadas ao processador de maior velocidade relativa ($H_C(A) = 0$).

Uma outra função de custo é utilizada por Shen e Tsai em [6]. Ela considera o custo da alocação como sendo o custo do processador mais carregado (com o maior valor da função de custo).

$$H(A) = \max_p (PL_p + \sum_{q \neq p} CL_{(p,q)}). \quad (5)$$

Com isso, espera-se que ela não acumule muita carga num só processador, pois isto aumentaria bastante o custo máximo. Ela, entretanto, pode não gerar uma boa distribuição de carga, pois apesar de evitar muita concentração num só processador, não há nada que estimule uma distribuição uniforme de carga entre os outros.

Para encorajar uma distribuição de forma mais uniforme entre processadores, faremos com que a função g da equação (4) seja de tal forma que

$$H_P(A) = \sum_p (PL_p)^2, \quad (6)$$

conforme utilizado em [7]. A função de avaliação de uma alocação do nosso modelo é dada então por

$$H(A) = \sum_p (PL_p)^2 + \sum_{(p,q)} CL_{(p,q)}. \quad (7)$$

Com isso, estimula-se a distribuição de carga mais equilibrada, uma vez que se muitas tarefas ficarem num mesmo processador, o termo "cruzado" (devido ao quadrado do custo de processamento) da parcela de $H(A)$ correspondente ao processamento irá ficar bastante significativo.

3. Alocação com A^*

Em Inteligência Artificial, é muito comum o uso de métodos informados de busca para se

encontrar a solução de um problema. Estes métodos constroem um grafo (árvore de busca) para se encontrar uma solução.

Eles utilizam informações dependentes do problema para se reduzir a árvore de busca. Usualmente, esta informação é colocada sob a forma de uma função de avaliação associada a cada vértice, que geralmente é uma forma de expressar o custo do estado representado pelo vértice. Um dos métodos informados bastante usado é o A^*.

No algoritmo A^*, a função de avaliação é composta por uma função de custo do vértice e por uma heurística. A heurística, usualmente representada por $h(n)$, procura ser uma previsão do custo necessário para ir do vértice n até o vértice-solução, caso este caminho seja percorrido. Com este dado adicional, geralmente se obtém uma busca mais eficiente, com menos vértices gerados na árvore.

A função de avaliação de um vértice n é definida então como:

$$f(n) = g(n) + h(n), \qquad (8)$$

onde $g(n)$ é o custo do vértice e $h(n)$ é a função heurística.

Se um algoritmo encontra sempre a solução ótima caso ela exista, então ele é dito admissível. É mostrado por Nilsson [8], que se $h(n) \leq h^*(n)$ para todos os vértices n, então o algoritmo A^* é admissível, onde $h^*(n)$ é o custo do caminho de menor custo (ótimo) do vértice n até o vértice-solução.

Usaremos o A^* para resolver a alocação estática de tarefas em múltiplos processadores. No nosso modelo, cada vértice n representará uma alocação parcial A_n. A cada nível, uma tarefa é alocada a um processador. A alocação A_n será dada então percorrendo-se desde o vértice n até a raiz, onde cada vértice no caminho terá a informação de qual processador a tarefa correspondente àquele nível, está alocada. A cada um destes vértices está associada uma função de avaliação $f(n)$, dada pela equação (8).

Neste trabalho, a partir do método sugerido por Shen e Tsai [6], fizemos mudanças nas funções de custo e na heurística, introduzindo também alguns novos procedimentos para melhorar o desempenho.

Adotou-se a função de custo dada pela equação (7).

Foi introduzido um parâmetro configurável que permite variar a a distância máxima de alocação de duas tarefas que se comunicam. Representaremos este parâmetro por DV. Isto quer dizer que se esta distância DV for igual a 1, as tarefas que se comunicam entre si só poderão ser alocadas ao mesmo processador ou a processadores vizinhos. Com isso, a partir de uma dada alocação parcial A_n, para se alocar uma nova tarefa i, deve ser certificado que a alocação é válida.

A função heurística adotada, baseada na função de custo (7) é:

$$h(n) = \sum_{i \in TN} \min_{p \in VP_i} \left(\left(\frac{\sigma_i}{s_p} \right)^2 + 2 \sum_{j | A(j)=p} \frac{\sigma_i \sigma_j}{s_p^2} + \sum_{j \in TA} \left(\sum_{(r,s) \in R(p,A(j))} \frac{\gamma(i,j)}{c_{(r,s)}} + \sum_{(r,s) \in R(A(j),p)} \frac{\gamma(j,i)}{c_{(r,s)}} \right) \right). \qquad (9)$$

onde TN é o conjunto das tarefas não alocadas ainda , TA é o conjunto das tarefas já alocadas e VP_i é o conjunto de processadores onde i pode ser alocado (limitação imposta por DV).

Pode-se facilmente ver que esta heurística é admissível, uma vez que $h(n) \leq h^*$. Isto é sempre verdade nesta função porque ao calcularmos o $h(n)$, não estamos levando em consideração a comunicação entre as tarefas ainda não alocadas ($\in TN$). Ele é portanto no máximo igual a h^* (quando nenhuma das tarefas pertencentes a TN se comunicam entre si).

No sentido de se obter uma busca mais em profundidade e consequentemente menos vértices

explorados, é feita inicialmente uma arrumação na ordem em que as tarefas serão alocadas [5]. Esta ordem de alocação é colocada de tal forma que as tarefas que possuem maior demanda de comunicação e de processamento sejam alocadas primeiro. A razão disso está na heurística que é utilizada no algoritmo. Observe que se alocarmos inicialmente as tarefas com pouca demanda (σ_i e $\gamma_{(i,j)}$ pequenos), teremos uma busca em largura (valor de $h(n)$ baixo se comparado ao de $g(n)$). Ao fazer esta arrumação, teremos um $h(n)$ mais próximo do real, levando a uma busca mais eficiente.

Foi adotado um algoritmo de poda para o A^*, uma vez que a árvore de busca gerada se tornou muito grande. Foi implementada uma limitação no tamanho da lista de abertos [9]. Quando esta lista excede o limite máximo, o vértice $n \in LA$ que possuir o maior $f(n)$ é eliminado, onde LA é a lista de abertos. Com isso, nem sempre se poderá garantir que a solução encontrada é ótima.

4. Alocação com Redes Neuronais

A idéia de redes neuronais é baseada no funcionamento de um cérebro, onde um grande número de elementos processadores (neurônios) executam coletivamente uma certa tarefa eficientemente.

Várias proposições do modelo funcional de um cérebro tem sido apresentados. Alguns descrevem os neurônios como sendo elementos lógicos de decisão com dois estados (on - off) que estão organizados numa rede para processar funções booleanas (McCulloch e Pitts [10]). São os chamados neurônios binários. Outros usam um modelo onde os neurônios podem assumir valores contínuos. Hopfield e Tank em [3] propõem um modelo que faz uma analogia com um circuito elétrico. Vamos a seguir fazer uma descrição detalhada deste modelo que usaremos para fazer a alocaçao estática de tarefas.

Seja uma rede neuronal composta de m neurônios $N_1, N_2, \ldots, N_m$. O estado de um neurônio N_i é dado por V_i, onde o seu valor varia de $0 \leq V_i \leq 1$. Considera-se que cada neurônio está conectado a outros $m-1$ neurônios (conexões denominadas de sinapses). O estado V_i é função do potencial P_i do neurônio num dado instante. Esta função obedece a uma sigmóide $f_i(P_i)$ mostrada na figura 1, cuja equação é

$$V_i = f_i(P_i) = \frac{1}{1 + e^{-(P_i - \theta_i)/T_i}}. \qquad (10)$$

Observe que os parâmetros θ_i e T_i podem ser diferentes para cada neurônio dentro da mesma rede. Isto quer dizer que dado um mesmo potencial P_i, os estados V_i dos neurônios podem ser diferentes.

O parâmetro θ_i é o "threshold" do neurônio N_i, conforme mostra a figura 1 e T_i regula a inclinação da curva no ponto $P_i = \theta_i$. Quanto maior o T, menor será esta inclinação.

Já o potencial P_i obedece a seguinte equação diferencial

$$C_i \frac{dP_i}{dt} = \sum_{j \neq i} w_{(i,j)} V_j - \frac{P_i}{R_i} + I_i, \qquad (11)$$

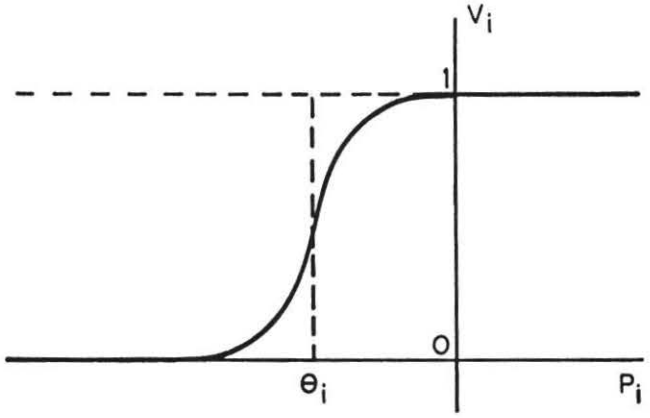
onde I_i é a influência de uma entrada externa ao circuito no neurônio N_i, P_i é o potencial do neurônio N_i, R_i é a "resistência" do neurônio N_i, $w_{(i,j)}$ é a influência (sinapse) do neurônio N_j sobre o neurônio N_i, C_i é a "capacitância" do neurônio N_i e V_j é o estado do neurônio N_j.

Como se pode notar, a variação de P_i, fazendo uma analogia com um circuito elétrico, depende de uma corrente externa ao circuito (I_i), de uma corrente de fuga (P_i/R_i) e de correntes pos-sinápticas ($w_{(i,j)}V_j$) induzidas em N_i por todas as atividades pre-sinápticas dos outros neurônios. Esta influência $w_{(i,j)}$ pode ser excitatória ($w_{(i,j)} > 0$), inibitória ($w_{(i,j)} < 0$) ou nula ($w_{(i,j)} = 0$).

Os parâmetros R_i e C_i determinam as características de constante de tempo de carga e descarga do neurônio, isto é, a "velocidade" com que ele responde a uma entrada. Pode-se dizer que a constante de tempo de um neurônio é dada por R_iC_i.

Para uma rede de neurônios com as conexões sinápticas ($w_{(i,j)}$) escolhidas ao acaso, o seu comportamento (variações de estado com o tempo) pode ser muito complexo, de difícil previsão. Para circuitos simétricos ($w_{(i,j)} = w_{(j,i)}$), entretanto, obtém-se um poderoso teorema sobre o comportamento do sistema [3]. Para o teorema ser válido, além do circuito ser simétrico, as entradas externas (I_i) devem ter variações lentas durante o tempo de estabilização do sistema. A outra exigência é que a relação entre a entrada e a saída do neurônio seja monotônica e limitada. A sigmóide da figura 1 atende a esta condição.

Fig. 1 : Sigmóide - Curva de resposta de um neurônio



O teorema mostra que a quantidade de energia E do sistema decresce durante as mudanças no estado da rede neuronal. Em outras palavras, dado um estado inicial, o sistema procurará minimizar a função de energia E, alcançando um estado onde E é um mínimo local. O sistema se tornará então estável, parando de variar com o tempo. Esta função de energia que é minimizada no sistema é dada por

$$E = -\frac{1}{2}\sum_{i\neq j} w_{(i,j)}V_iV_j - \sum_i I_iV_i + \sum_i \frac{\theta_i}{R_i}V_i. \quad (12)$$

De acordo com esta característica da rede neuronal, nos parece bastante provável que se adaptarmos a função custo de uma alocação à E e ajustarmos convenientemente os diversos parâmetros do sistema, poderemos conseguir que a rede neuronal obtenha uma alocação aceitável rapidamente.

Descrevem-se a seguir o modelo que usamos para representar os grafos de tarefas e processadores e a influência dos parâmetros do problema na rede neuronal.

Seja $|N_T|$ o número de tarefas e $|N_P|$ o número de processadores. O sistema será composto de $|N_T||N_P|$ neurônios arrumados numa matriz $|N_T| \times |N_P|$, onde todos os neurônios estão interconectados entre si. Considera-se que se o neurônio da linha i e coluna p,isto é, $N_{(i,p)}$ possui valor $V_{(i,p)} > V_{li_g}$, a tarefa i está alocada ao processador p ,onde V_{li_g} é o valor mínimo para o qual um neurônio é considerado como ligado.

Deve-se adaptar agora a função custo $H(A)$ dada na equação (7) para a forma da função de energia E para que ela seja minimizada.

Antes de mais nada, devemos assegurar que o estado final seja uma configuração válida, isto é, não haja uma mesma tarefa alocada a mais de um processador. Para isso, todos os neurônios da mesma linha, ($N_{(i,p)}$ e $N_{(i,q)}$), são conectados por uma sinapse com uma influência inibitória grande ($-W$), de forma que o par contribuirá com uma parcela elevada para a energia E no caso de $V_{(i,p)} = V(i,q) = 1$. Isto evita a alocação da tarefa i aos processadores p e q ao mesmo tempo.

A parcela $H_P(A) = (PL_p)^2$ será dividida da seguinte maneira:

- os termos "cruzados" estarão representados nas forças sinápticas entre neurônios

neurônios na mesma coluna ($w_{((i,p),(j,p))}$).

- os demais termos serão as entradas externas ($I_{(i,p)}$).
- Os custos relativos a comunicação serão representados por

$$w_{((i,p),(j,q))} = -\sum_{(r,s)\in R(p,q)} \frac{\gamma(i,j)}{c_{(r,s)}} - \sum_{(r,s)\in R(q,p)} \frac{\gamma(j,i)}{c_{(r,s)}} \quad (13)$$

A garantia de que uma tarefa sempre será alocada a um dos processadores é dada pelo termo

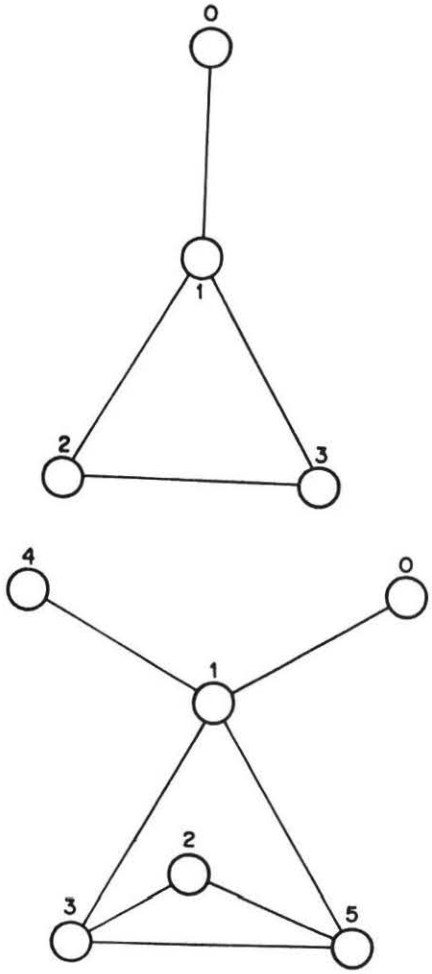
$\sum_i(\theta_i \ / \ R_i)\ V_i$ da equação de energia (12). Observe que quando θ_i é negativo, a contribuição de um neurônio N_i ligado para a energia é negativa, fazendo com que esta decresça. Poderia se esperar com isso, que a energia será mínima quando todos os neurônios da rede estiverem ligados. Porém isto não ocorre devida a sinapse inibitória $-W$ que passará a atuar quando neurônios da mesma linha ficam ligados simultaneamente, elevando a energia da rede.

5. Testes Realizados

Nesta seção, descreveremos algumas experiências realizadas com os algoritmos A^* e rede neuronal, no sentido de fazer uma análise sobre as suas características. Procuramos variar os parâmetros de entrada dos algoritmos com o objetivo de estudar os seus comportamentos. É feito também uma comparação entre as alocações obtidas bem como o tempo gasto para conseguí- las. Dentre todos os testes feitos escolhemos um dos grafos (fig 2) para ilustrar o resultados. Os seus parâmetros estão apresentados na tabela 1.

Para o A^* fizemos testes variando o tamanho da lista de abertos e a distância máxima para alocação de tarefas que se comunicam.

Fig. 2 : Grafos de processadores (G_P) e tarefas (G_T)



Notou-se que para grafos pequenos, é possível se encontrar a solução ótima. Entretanto, de uma maneira geral, é necessário gerar um grande número de vértices para se obter uma solução ótima, mesmo para grafos relativamente modestos. Porém, para alocações subótimas obtidas limitando a lista de abertos, o tempo requerido é relativamente pequeno.

Dependendo dos grafos de processadores e tarefas e seus parâmetros, a limitação da distância máxima de alocação pode não piorar a qualidade

Tabela 1 : Parâmetros dos grafos G_T e G_P

$i \backslash j$		0	1	2	3	4	5
	σ_i	$\gamma_{(i,j)}$					
0	5	-	8	-	-	-	-
1	4	8	-	-	5	7	10
2	1	-	-	-	9	-	15
3	20	-	5	9	-	-	4
4	15	-	7	-	-	-	-
5	10	-	10	15	4	-	-

$p \backslash q$		0	1	2	3
	s_p	$c_{(p,q)}$			
0	24	-	100	-	-
1	16	100	-	100	50
2	32	-	100	-	50
3	24	-	50	50	-

da alocação obtida conseguindo reduzir o número de vértices gerados (tempo de execução). Porém, na maioria dos casos, esta limitação resulta numa pior alocação.

Procuramos avaliar o comportamento do algoritmo da rede neuronal,variando os parâmetros $R_{(i,p)}$, $C_{(i,p)}$, $\theta_{(i,p)}$ e $T_{(i,p)}$. Eles se mostraram muito interdependentes entre si, gerando uma família de curvas. Esses resultados são apresentados com mais detalhes em [11].

Comparando os resultados obtidos pelo A^* e pela rede neuronal (tabela 2), notamos que a relação entre os custos das alocações e os tempos para obtê-las são bastante dependentes dos grafos e seus parâmetros.

Comparando as alocações obtidas pela rede neuronal às obtidas pelo A* com a lista de abertos igual a dois em termos de custo e tempo gasto, pode-se dizer que de uma forma geral:

- para grafos pequenos o desempenho do algoritmo A* é melhor.
- para grafos médios (conjuntos 4 e 6), os dois são comparáveis.
- para grafos grandes a rede neuronal é melhor.

Tabela 2 : Melhores alocações do A* e R. N.

Parâmetros do A*:

DV	LA
2	20

Parâmetros da rede neuronal:

Δt	ε	$\theta_{(i,p)}$	$T_{(i,p)}$	$C_{(i,p)}$	$R_{(i,p)}$
0.01	0.001	-2	0.04	20	0.01

Tarefa	*Processador*	
	*A**	*R.N.*
0	1	3
1	0	1
2	2	1
3	2	2
4	1	2
5	2	1
Custo	3.393	4.720
Tmp(s)	2	0.4

6. Considerações Finais

Com o algoritmo A^* é possível se achar a solução ótima. Porém com grafos maiores, o tempo e recursos computacionais que são necessários para se encontrar a solução ótima o tornam inviável. Entretanto, para se encontrar alocações próximas à ótima, o A^* se mostrou bastante aceitável, encontrando uma alocação num tempo comparável ao algoritmo da rede neuronal.

Com relação à alocação de tarefas pela rede neuronal, pode-se notar que seu funcionamento depende intimamente da escolha apropriada dos parâmetros $R_{(i,p)}$, $C_{(i,p)}$, $\theta_{(i,p)}$ e $T_{(i,p)}$ que são selecionados. Acreditamos que algumas relações entre os parâmetros existem, como no caso de $R_{(i,p)}$, $\theta_{(i,p)}$ e W conforme descrito na Seção 5, que podem ajudar na escolha apropriada dos parâmetros. Porém deve ser necessário ainda se fazer alguns ajustes que dependem dos parâmetros dos grafos de processadores e tarefas para se conseguir o melhor desempenho possível.

Deve-se lembrar também que o algo-

ritmo da rede neuronal tem características ideais para processamento paralelo, o que o torna bastante "atrativo" para a resolução do problema de alocação de tarefas em sistemas distribuídos.

Referências

(1) GAREY, M.R., e JOHNSON, D.S., "Computers and Intractability: A Guide to the Theory of *NP*-Completeness", Freeman, New York, NY, 1979.

(2) KRUMME, D.W., VENKATARAMAN, K.N., e CYBENKO, G., "Hypercube Embedding is *NP*-Complete," *Hypercube Multiprocessors 1986*, pp. 148–157 em M. T. Heath (Ed.), SIAM, Philadelphia, PA, 1986.

(3) HOPFIELD, J.J., e TANK, D.W., "Computing with Neural Circuits: a Model", *Science*, vol 233, pp 625-633, ago, 1986.

(4) FOX, G.C., e OTTO, S.W., "Concurrent Computation and the Theory of Complex Systems", *Hypercube Multiprocessors 1986*, SIAM, Philadelphia, PA, 1986.

(5) SINCLAIR, J.B., "Efficient Computational of Optimal Assignments for Distributed Tasks", *Journal of Parallel and Distributed Computing* 4, pp 342-362, 1987.

(6) SHEN, C.C., e TSAI, W.H., "A Graph Matching Approach to Optimal Task Assignment in Distributed Computation System Using a Minimax Criterion", *IEEE Transactions on Computers*, vol c-34, n 3, pp 197-203, mar, 1985.

(7) FOX, G.C., KOLAWA, A., e WILLIAMS, R., "The Implementation of a Dynamic Load Balancer", pp. 114-121 em M. T. Heath (Ed.), *Hypercube Multiprocessors 1987*, SIAM, Philadelphia, PA, 1987.

(8) NILSSON, N.J., "Principles of Artificial Intelligence", Tioga Publishing Co, Palo Alto, 1980.

(9) PEARL, J., "Heuristics - Intelligent Search Strategies for Computer Problem Solving", Addison-Wesley Publishing Company, 1984.

(10) McCULLOCH, W.S., e PITTS, W., *Bull Math. Biophys. 5*, 115, 1943.

(11) HUANG, H.K., "Técnicas para Alocação Estática de Tarefas em Sistemas Distribuídos", *Tese de Mestrado em Engenharia de Sistemas*, COPPE - Sistemas, UFRJ, jun, 1988.