O PROBLEMA DE DESIGNAÇÃO DE CAPACIDADES EM REDES DE COMPUTADORES

*Sergio L. Wasserstein**

*Paulo R. Zanjacomo***

Orientador: Carlos Humes

1. Introdução

1.1. Objetivo

Obter um estudo qualitativo do problema de projetar uma rede de computadores, sob aspectos de custo e retardo em comunicações, buscando um algoritmo eficiente para a solução de um problema tipicamente combinatório.

1.2. O problema de designação de capacidades

O problema motivador é o de designação de capacidades (C.A.) que consiste em, definidas a topologia de uma rede e o roteamento das mensagens por ela, minimizar o custo total da instalação dos canais, considerando como variáveis as capacidades dos canais, sob restrição do valor máximo de retardo desejado dentro de um conjunto de capacidades disponíveis; ou analogamente, minimizar o retardo nas capacidades dos canais sujeito a um custo máximo desejado, ou seja:

dados: topologia, fluxos $F = (f_1, f_2, \ldots, f_m)$

minimizar: $D(C)$ (ou $T(C)$)

sujeito a: $\begin{cases} T(C) \leq T_{max} * \gamma(D(C) \leq D_{max}) \\ c_i > f_i \\ c_i \in C \end{cases}$

onde

$T(C)$ é a função custo,

---

* Bolsista de Iniciação Científica do CNPq.

** Bolsista de Iniciação Científica da Fapesp.

$D(C)$ é a função retardo,

γ é o número médio de mensagens na rede,

C é o conjunto de capacidades disponíveis.

1.2.1. Considerações sobre o C.A.

1. Em geral, a função custo é côncava:

$$D(C) = \sum_{i=1}^{m} d_i * c_i^{\alpha}, \quad \alpha \in (0,1)$$

2. Em geral, a função retardo é convexa:

$$T(C) = \sum_{i=1}^{m} \frac{f_i}{c_i - f_i}.$$

3. Apesar do conjunto de capacidades disponíveis ser discreto, pode ser útil resolver o problema contínuo e usá-lo como heurística para o discreto.

4. Pode-se resolver o problema contínuo através de sucessivas linearizações, pois o problema linearizado tem a seguinte fórmula fechada. para achar capacidade ótima (c^*):

$$c_i^* = f_i + \frac{\sum_{i=1}^{m} \sqrt{f_i * d_i}}{\gamma * T_{max}} * \sqrt{\frac{f_i}{d_i}}.$$

2. Resolução do C.A. - Um problema combinatório

2.1. Algoritmo

O algoritmo aqui estudado consiste em se combinar as capacidades disponíveis com os canais da rede de forma viável, preocupando-se com a combinação que gera o menor custo.

O algoritmo básico seria:

- Para cada canal atribua as capacidades disponíveis viáveis.
- Repita.
  1. Combine todos os canais gerando todos os pares (i, j), onde i é alguma capacidade do primeiro canal e j do segundo;

2. elimine aquelas combinações que não conduzem à solução ótima (como?);

3. se não foram geradas todas as combinações, vá para 1.

• Escolha a que gera o menor custo.

Evidentemente um algoritmo de busca exaustiva não consideraria o passo 2 acima.

Estamos interessados em combinar canais de maneira eficiente, ou seja, a melhor política computacionalmente falando e em como eliminar os eficientes nos passos 1 e 2 do algoritmo.

2.2. Definições

Tabela é um conjunto de capacidades viáveis atribuídas a um conjunto de canais, com seus respectivos custos e retardo total.

O *Merge* de duas tabelas T_1 e T_2, denotando $T_1 \oplus T_2$, é o processo de se combinar todos os elementos de T_1 com todos os elementos de T_2, formando uma nova tabela T, i.e.:

$$T = \{e_i + f_j / e_i \in T_1,\ f_j \in T_2\}.$$

O processo de merge de duas tabelas geram conjuntos da forma:

$$T = \{e_i + f_j / e_i \in T_1,\ f_j \in T_2\}, \quad \text{com os } e_i\text{'s fixos.}$$

A este conjunto será dado o nome de *Sub-Tabela*.

A formação de $T = T_1 \oplus T_2$ consiste então em se agrupar e ordenar todas as sub-tabelas geradas no processo de merge.

Dadas duas sub-tabelas s_1 e s_2, o processo de agrupar e ordenar s_1 e s_2 será denominado *sortmerge* e denotado como $s_1 \cup s_2$.

3. Combinação de canais

Dadas duas tabelas associadas a dois conjuntos de canais, estudemos agora a complexidade de fazer o merge e os sortmerge's do processo.

3.1. Sortmerge

Uma estratégia de executar o sortmerge de m sub-tabelas é o assim chamado Divide & Conquer, i.e.:

dadas m sub-tabelas enumere-as $1, 2, 3, \ldots, m$ e junte-as da forma:

$$(\ldots((t_1 \cup t_2) \cup (t_3 \cup t_4)) \cup \ldots \cup (t_{m-1} \cup t_m)) \ldots)$$

Lema 1. *Dadas m sub-tabelas de tamanho n, a melhor política de fazer o* sortmerge *com elas é utilizando a estratégia* D&C.

3.2. Merge.[1]

Lema 2. *Seja $T = \{t_1, t_2, \ldots, t_m\}$ conjunto de tabelas, se $|t_1| \geq |t_2| \geq \ldots \geq |t_m| \geq 2$, então a melhor política para mergeá-las é:*

$$(\ldots(((t_1 \oplus t_2) \oplus t_3) \oplus t_4) \oplus \ldots \oplus t_m)$$

Lema 3. *Sejam T_1 e T_2 tabelas formadas como no lema acima: seja*

$$|t_1| \geq |t_2| \geq \ldots \geq |t_p| \geq 2 \in T_1,$$

$$|t_{p+1}| \geq |t_{p+2}| \geq \ldots |t_k| \geq 2 \in T_2,$$

se $|t_p| < |t_{p+1}|$ *e* $|T_1^* \oplus t_{p+1}| > |T_2^* \oplus t_p|$

onde $T_1^* = (\ldots(((t_1 \oplus t_2) \oplus t_3) \oplus t_4) \oplus \ldots \oplus t_{p-2})$,

$T_2^* = (\ldots(((t_{p+2} \oplus t_{p+3}) \oplus t_{p+4}) \oplus t_4) \oplus \ldots \oplus t_k)$

então

$|(T_1^* \oplus t_{p+1}) \oplus (T_2^* \oplus t_p))| \leq |T_1 \oplus T_2|$

onde $|t \oplus q|$ é o número de comparações no merge de t com q.

---

[1] Considerando o custo de $t_1 \oplus t_2$ como $|t_1| * |t_2| \log_2(\min\{t_1, t_2\})$.

Lema 4. *Sejam* T_1 *e* T_2 *tabelas formadas como no lema acima; seja*

$$|t_1| \geq |t_2| \geq \ldots \geq |t_p| \geq 2 \in T_1,$$

$$|t_{p+1}| \geq |t_{p+2}| \geq \ldots \geq |t_k| \geq 2 \in T_2,$$

se $|t_p| \geq |t_{p+1}|$

então

$$|(T_1 \oplus t_{p+1}) \oplus T_2^*)| \leq |T_1 \oplus T_2|$$

onde $|t \oplus q|$ *é o número de comparações no merge de* t *com* q *e*

$$T_2^* = (\ldots(((t_{p+2} \oplus t_{p+3}) \oplus t_{p+4}) \oplus t_4) \oplus \ldots \oplus t_k)$$

Teorema 1. *O custo computacional de se fazer o merge de* m *tabelas é mínimo se e somente se o processo seguir a disciplina seqüencial, i.e., fazer a cada passo o merge das duas maiores tabelas.*

Dem.: Segue direto dos lemas 2, 3, 4.

4. Considerações Finais

4.1. Eliminação de Ineficientes

4.1.1. Triviais

- Se o retardo de um elemento de uma tabela for maior que o retardo máximo, este elemento é eliminado.
- Uma designação $\bar{c} = (\bar{c}_1, \ldots, \bar{c}_k)$ é ineficiente se existe $c' = (c'_1, \ldots, c'_k)$ tal que

$$\begin{bmatrix} T(c') \\ D(c') \end{bmatrix} \leq \begin{bmatrix} T(\bar{c}) \\ D(\bar{c}) \end{bmatrix}$$

- Designações ineficientes são eliminadas.

4.1.2. Relaxação Contínua

O problema C.A. com o conjunto de restrições disponíveis contínuo pode ser usado como heurística para o problema discreto. A solução do contínuo relaxado é usada como

limite superior na resolução do discreto, assim qualquer eiemento gerado que ultrapasse os limites do contínuo será eliminado.

4.2. Programação Dinâmica

O C.A. pode ser visto como o problema de se encontrar o caminho mais curto em um grafo com peso nas arestas. O algoritmo, então, pode ser visto sob o enfoque de programação dinâmica, onde o nó inicial representa nenhuma capacidade atribuída a nenhum canal e o nó final todas as capacidades designadas a todos os canais de maneira eficiente. e o peso é o custo real de se atribuir determinada capacidade a determinado canal.

Referências

[1] Aho, A.V.; Hopcroft, J.E.; Ullmann, J.D. The Design of Computer Algorithms. Reading, Addison-Wesley, 1974. 470 p.

[2] Bellman, R.E.; Dreyfus, S.R. Applied Dynamic Programming. Princeton University Press, 1962, 363 p.

[3] Bezerra, J.R.M. Sobre Problemas de Otimizaçãom em Redes de Computadores. São Paulo, 1984, 233 p. (M.Sc.Diss.) IME-USP.

[4] Gerla, M. The Design of Store and Forward Computer Networks. Los Angeles, 1973. 220 p. Ph.D. (Thesis) U.C.L.A.

[5] Humes Jr., C. Tópico de Otimização em Redes de Computadores. 1988, 112 p. (Livre Docência) IME-USP.

[6] Kleinrock, L. Queueing Systems. New York, John Wiley, 1975-76, 2 v.