MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO DE PESQUISAS ESPACIAIS

AUTORIZAÇÃO PARA PUBLICAÇÃO
AUTHORIZATION FOR PUBLICATION

PALAVRAS CHAVES/KEY WORDS

AUTORES / AUTHORS

PROCESSAMENTO PARALELO
OTIMIZAÇÃO IRRESTRITA
PROGRAMAÇÃO NÃO-LINEAR

AUTORIZADA POR/AUTHORIZED BY

Marco Antonio Raupp
Diretor Geral

AUTOR RESPONSAVEL
RESPONSIBLE AUTHOR

Luiz Antonio N. Lorena

DISTRIBUIÇÃO/DISTRIBUTION

- [ ] INTERNA / INTERNAL
- [x] EXTERNA / EXTERNAL
- [ ] RESTRITA / RESTRICTED

REVISADA POR / REVISED BY

(Vide Obs.*)

CDU/UDC

519.863

DATA / DATE

August 1987

PUBLICAÇÃO Nº
PUBLICATION NO

INPE-4298-PRE/1169

ORIGEM
ORIGIN

LAC

PROJETO
PROJECT

POPES

Nº DE PAG.
NO OF PAGES

23

ULTIMA PAG.
LAST PAGE

18

VERSÃO
VERSION

Nº DE MAPAS
NO OF MAPS

TÍTULO/TITLE

PROCESSAMENTO PARALELO EM PROGRAMAÇÃO NÃO-LINEAR

AUTORES/AUTHORSHIP

Ana Clara da Mota
Luiz Antonio Nogueira Lorena

RESUMO - NOTAS / ABSTRACT - NOTES

O aparecimento de computadores paralelos representou um desafio para a criação de novos algoritmos e/ou adaptações de algoritmos existentes em Análise Numérica e Otimização. Particularmente em Programação Não-Linear, várias propostas para otimização irrestrita utilizando idéias de paralelismo, vêm sendo divulgadas recentemente. O objetivo deste trabalho é apresentar uma revisão de métodos de otimização irrestrita usando processamento paralelo. Basicamente são mostrados métodos de direções conjugadas e métodos tipo Newton. Também apresentam-se as principais arquiteturas de computadores paralelos com suas possíveis vantagens e desvantagens em uma aplicação a problemas de otimização irrestrita.

OBSERVAÇÕES / REMARKS

Trabalho apresentado no XVI Colóquio Brasileiro de Matemática, 19-25 de julho, IMPA, Rio de Janeiro-RJ.

*Dispensado da Revisão Técnica e Revisão de Linguagem.

ABSTRACT

The appearance of parallel computers is a challange for Numerical Analysis and Optimization. In this work we make a survey of parallel unconstrained optimization methods. Conjugate directions and Newton methods are presented for some of the main architectures of parallel computers.

SUMÁRIO

PROCESSAMENTO PARALELO EM PROGRAMAÇAO NÃO-LINEAR

Ana Clara da Mota

Luiz Antonio Nogueira Lorena

Ministèrio da Ciência e Tecnologia - MCT

Instituto de Pesquisas Espaciais - INPE

Caixa Postal 515 - 12201 - São Josè dos Campos - SP

RESUMO

O aparecimento de computadores paralelos representou um desafio para a criação de novos algoritmos e/ou adaptações de algoritmos existentes em Anàlise Numèrica e Otimização. Particularmente em Programação Não-Linear, vàrias propostas para otimização irrestrita utilizando idèias de paralelismo, vêm sendo divulgadas recentemente.

O objetivo deste trabalho è apresentar uma revisão de mètodos de otimização irrestrita usando processamento paralelo, e tambèm levantar novas possibilidades de pesquisa na àrea. Basicamente são mostrados mètodos de direções conjugadas e mètodos tipo Newton. Tambèm apresentam-se as principais arquiteturas de computadores paralelos com suas possiveis vantagens e desvantagens em uma aplicação a problemas de otimização irrestrita.

1 - INTRODUÇAO:

O aparecimento de computadores com processamento paralelo representou um desafio para a criação de novos algoritmos , e/ou adapatação de algoritmos existentes em Anàlise Numèrica e Otimização. Em Programação Não-linear, em particular, vàrias propostas têm sido divulgadas recentemente, algumas com adaptações de algoritmos com atè vinte anos de idade.

Neste trabalho apresenta-se uma revisão de mètodos de otimização irrestrita usando processamento paralelo, mostrando basicamente idèias que surgiram de adaptações de

mètodos tipo Newton e mètodos de direções conjugadas que não usam informações de gradientes.

No capítulo 2 faz-se uma revisão das principais classes de arquiteturas de computadores paralelos, destacando suas potencialidades na aplicação em otimização irrestrita. Também são apresentadas medidas de desempenho dos computadores paralelos em relação aos sequenciais.

No capítulo 3, apresentam-se os mètodos tipo Newton, onde a maior parte das pesquisas estão direcionadas ao processo de busca em linha (unidimensional) do mètodo, embora algumas pesquisas estejam centradas na idèia òbvia de calcular os gradientes e matrizes Hessianas usando avaliações paralelas da função objetivo do problema. Outras idèias buscam paralelizar os sistemas de equações lineares resultantes do cálculo das direções de busca.

A seguir são vistos mètodos de direções conjugadas em que as pesquisas estão centralizadas em explorar vàrias direções simultâneas de busca em paralelo.

É importante notar que serão revisados apenas mètodos de otimização irrestrita locais, embora os mètodos globais sejam uma àrea muito favorável para o paralelismo. Note-se também que a maior parte dos mètodos globais executam otimizações locais e portanto podem potencialmente usar os resultados anteriores. Outra àrea importante è a de otimização com restrições, onde também são executadas otimizações locais.

2 - ARQUITÉTURA DOS COMPUTADORES PARALELOS:

Flynn (1972) apresentou um dos primeiros e mais amplos esquemas de classificação dos computadores paralelos, baseando-se na multiplicidade de instruções e fluxo de dados que a màquina comporta. Na classificação de Flynn são definidas quatro classes de màquinas: SISD, MISD, SIMD, e MIMD; que são descritas a seguir:

a) SISD (Single Instruction Stream, Single Data Stream):

Esta classe corresponde à arquitetura convencional de computador sequencial. Todas as instruções são executadas

sequencialmente, sendo que cada instrução é executada em um dado instante, sobre um determinado conjunto de dados.

b) MISD (Multiple Instruction Stream, Single Data Stream):

Esta classe corresponde à máquina onde determinado número de processadores executa, simultaneamente, diferentes instruções sobre o mesmo conjunto de dados. Pouca atenção lhe é atribuída e não existem realizações práticas.

c) SIMD (Single Instruction Stream, Multiple Data Stream):

Nesta classe a mesma instrução é executada simultaneamente sobre vários conjuntos de dados. Um determinado número de elementos de processamento (cada um com sua própria armazenagem local) é coordenado por uma única unidade de controle. Em um dado instante, cada processador está executando a mesma instrução em um conjunto diferente de dados.

Uma máquina SIMD consiste de:

- uma unidade de controle (UC);
- N processadores (EPs);
- uma ligação em rede.

A unidade de controle busca e decodifica a instrução para todos os processadores, e estes a executarão ao mesmo tempo. Existe, portanto, um único fluxo de instrução.

O computador SIMD possui uma única memória acessível à UC e todos EPs. Mas, geralmente ela é decomposta (como mostra a figura 1) para permitir acessos simultâneos à cada EP. Sua estrutura de ligação emite dados da UC para todos EPs, ou vice-versa, e também permite a troca direta de dados entre EPs. As ligações para o sistema de I/O (entrada/saída) e ao computador hospedeiro, são fornecidas normalmente.

O primeiro computador a usar esse tipo de arquitetura foi o ILLIAC IV, com 64 processadores organizados em uma matriz 8x8 (veja Ribeiro, 1985). Outros exemplos: ICL-DAP (Distributed Array Processor) e o Burroughs BSP (Scientific Processor)

d) MIMD (Multiple Instruction Stream, Multiple Data Stream)

Esta classe corresponde ao modelo mais geral de arquitetura paralela. São também chamados multiprocessadores, porque geralmente, cada processador atua em um conjunto diferente de dados. Os computadores MIMD são caracterizados pela realização de operações simultâneas, feitas com múltiplos fluxos de instruções.

Neste tipo de arquitetura, existem n componentes de processamento, cada qual possuindo um decodificador de instrução e uma unidade lógica aritmética inteiramente independente dos outros processadores, de forma que os processadores podem decodificar e executar instruções simultaneamente. Neste caso, cada processador executa um programa.

Um computador MIMD consiste de m módulos de memórias, onde cada processador pode ter acesso independente. Chaves de cruzamento permitem qualquer dos n processadores acessar uma das m memórias, e assim nxm cruzamentos possuem potencial de conexão, permitindo acesso paralelo (ver figura 2). Nesta classe encontra-se o C.mmp (ver Rodrigue, 1982).

Esta foi uma descrição rápida das arquiteturas de computadores paralelos segundo Flynn. Os algoritmos mostrados no Capítulo 3, em geral admitem máquinas SIMD e algumas vezes MIMD, como será visto adiante. Existe ainda outro tipo de arquitetura que geralmente não é considerada na classificação de Flynn, esta é a arquitetura do computador "pipelined" que é descrita a seguir.

O computador "pipelined" permite uma redução direta em tempo de processamento na fase de operações aritméticas. Consiste de uma organização relacionada de processadores, como mostra a figura 3. Neste esboço, cada um dos passos possui uma unidade separada dentro da CPU desempenhando a execução da instrução (por exemplo, busca da instrução, determina o tipo de instrução, localiza os dados, ...).

As execuções de instruções neste tipo de arquitetura efetuam-se da seguinte forma:

A primeira unidade busca a instrução inicial. Enquanto a segunda unidade inicia a decodificação da instrução inicial, a primeira unidade permanece ocupada buscando a segunda instrução. A terceira unidade começa a examinar a instrução inicial observando se existe a necessidade de dados da memória, enquanto a segunda unidade

está decodificando a segunda instrução e a primeira unidade está buscando a próxima instrução.

Computadores vetoriais formam uma subclasse dos computadores "pipelined" e usam essa técnica para agilizar operações aritméticas com vetores; sendo assim são indicados a problemas grandes, onde a parte de operações com vetores é predominante, tais como a resolução de grandes sistemas de equações lineares e em muitos algoritmos usados para a solução de sistemas de equações diferenciais parciais. Exemplos de máquinas vetoriais são o CRAY-1 e o CYBER-205. O IBM 360/195 e o CDC STAR usam sofisticadas formas de "pipelined" (ver Tanenbaum, 1984).

Algumas universidades americanas têm usado redes locais de computadores em experimentos de processamento paralelo, tais como o projeto Crystal da Universidade de Wisconsin e o projeto ENCOMP da Universidade do Colorado (ver Schnabel, 1986). Tais redes consistem de múltiplos computadores com grande proximidade física, e uma rápida rede de comunicação, embora substancialmente mais lenta que a comunicação entre processadores em um computador individual. Alguns dos algoritmos a serem apresentados poderiam ser testados em redes desse tipo, naturalmente com alguma dificuldade na comunicação e maior tempo de processamento do que em máquinas SIMD e MIMD.

Um problema que ocorre quando se pensa em utilizar alguma das máquinas descritas, é o da determinação de uma medida de desempenho da máquina. A seguir descreve-se algumas possibilidades para máquinas tipo SIMD e MIMD:

- Para sistemas SIMD:

O desempenho do algoritmo paralelo para arquiteturas SIMD é medido, em geral, em relação a um sistema sequencial. A razão de desempenho ϕ é dada por: ϕ = tempo de processamento em um sistema sequencial pelo tempo de processamento em um sistema paralelo (SIMD). Por exemplo, seja a divisão de uma função objetivo em um número de tarefas idênticas, como no problema

$$\min \quad f(x) = \sum_{K=1}^{M} S_K (x),$$

onde cada $S_K(x)$ é um cálculo idêntico utilizando dados diferentes. Tem-se portanto um caso óbvio onde (se $M \leq p$ = número de processadores) uma máquina SIMD poderá ser usada (Patel, 1982).

- Para sistemas MIMD:

Neste caso o desempenho poderá ser medido em termos do fator de aceleração, definido como a razão do tempo de processamento em um processador pelo tempo de processamento em p processadores. Se $\theta(p)$ è o tempo de processamento usando p processadores, então o fator de aceleração è igual a: $\theta(1)/\theta(p)$. Deveria-se supor que esta razão seja p, mas isto não ocorre em geral, e dois são os fatores que contribuem para uma perda:

(a) falta de sincronização dos processadores,

(b) impossibilidade de acesso simultâneo ao conjunto de dados globais.

A perda por falta de sincronização è ilustrada pela situação de um problema em que, quando tenta-se um solução por um algoritmo sequencial, o tempo gasto na rotina de avaliação da função è cem vezes o tempo gasto na parte de otimização do algoritmo. Um exemplo particular disso (Patel, 1982) è o caso da otimização do consumo de combustível de uma aeronave, sujeito a restrições de ruído e especificações de desempenho. Para paralelizar tal problema, observam-se os aspectos paralelos na rotina de avaliação da função. Cada tarefa paralela pode tomar um tempo diferente para ser completada e o valor da função objetivo não è calculado enquanto todas as tarefas executadas em paralelo não são completadas. Esta è a perda de sincronização que pode ser minimizada garantindo que as tarefas paralelas são aproximadamente iguais em duração (ver figura 4).

A outra perda no fator aceleração è devido ao acesso de muitos processadores no conjunto de dados globais em um instante particular de tempo. Em qualquer instante, somente um processador pode ter acesso a um dado global; assim, enquanto este está acessando, os outros processadores estão esperando. Pode-se minimizar o tempo de espera garantindo que o conjunto de dados globais è reduzido ao mínimo.

3 - ALGORITMOS PARA OTIMIZAÇÃO IRRESTRITA:

Existe uma variedade de algoritmos para encontrar um mínimo local de uma função continuamente diferenciável, sem restrições. (ver Fletcher, 1980; Gill, Murray & Wright, 1981). Neste capítulo serão revistos alguns algoritmos adaptados ao processamento paralelo. Serão apresentados algoritmos baseados em dois tipos de métodos, os tipo Newton e os de direções conjugadas.

3.1 - METODOS TIPO NEWTON:

Uma iteração k do método de Newton para minimizar uma função não-linear f no espaço R^n, pode ser resumido nos seguintes passos: ($g(x^k)$ = gradiente em x^k, $G(x^k)$ = Hessiana em x^k)

Passo 1 - geração da direção de busca

$$G(x^k).s^k = - g(x^k);$$

Passo 2 - exploração da direção de busca encontrando um mínimo local da função

$$\phi_k(\lambda) = f(x^k + \lambda.s^k),$$

em relação a $\lambda_k \geq 0$;

Passo 3 - fazer $x^{k+1} = x^k + \lambda_k.s^k$
Teste de otimalidade: se x^{k+1} e x^k estão "suficientemente" próximos, o processo termina.

O método de Newton sequencial; é excelente para problemas pequenos ($n \leq 30$), sendo sua principal desvantagem o fato de que o usuário deve fornecer as derivadas primeiras e segundas de f no processo de otimização. Várias rotinas de otimização proporcionam facilidades para diferenciação numérica (ver Gill, Murray & Wright, 1981). Neste caso pode-se usar uma máquina SIMD em avaliações paralelas da função no cálculo do gradiente e da matriz Hessiana, como uma aplicação imediata do paralelismo.

A seguir descreve-se duas idéias que referem-se a mudanças no passo 2 do esquema de algoritmo tipo Newton, isto é, no processo de busca em linha. A primeira idéia é um processo controlado de cercar um mínimo local de ϕ usando a avaliação paralela da função em vários pontos na linha gerada pela direção s^k. A segunda idéia propõe fazer uma busca bi-dimensional nas direções de Newton (passo 1) e a direção de descida mais rápida (- g), como veremos a seguir.

3.1.1 - BUSCA UNIDIMENSIONAL:

A busca pode ser realizada em dois passos (Lootsma, 1986) quando o paralelismo è introduzido. No primeiro passo, procura-se cercar um mínimo local de ϕ , calcula-se o valor da função num "grid" de pontos suficientemente grande (que pode ser gerado, pois $\lambda = 1$ è em geral uma boa aproximação para o mínimo na linha). Como resultado obtèm-se um intervalo que contèm o mínimo (como na figura 5). No passo de interpolação posterior (ver figura 6). toma-se outro "grid" de pontos no intervalo que contèm o mínimo e calcula-se o valor da função objetivo simultaneamente nos pontos do "grid". Usa-se o mínimo de uma função interpolação de alta ordem (polinomial, racional, ou alguma outra) para aproximar o mínimo na linha (os pontos do "grid" devem ser suficientemente espaçados para uma interpolação adequada). Lootsma experimentou essas idèias em estudos comparativos com otimização sequencial, chegando a conclusão que o número de processadores não será proibitivo para essa abordagem (em geral, dez serão suficientes). Deve ser ainda experimentado para outras variações do mètodo de Newton.

3.1.2 - BUSCA BI-DIMENSIONAL:

As idèias de busca bi-dimensional foram testadas por Dixon & Patel (Dixon & Patel, 1982) usando o mètodo de Newton amortecido, cuja variação se dá no passo 1 do algoritmo tipo Newton, isto è, no cálculo da direção de busca.

No mètodo de Newton amortecido, gera-se a direção de busca $s^k(\mu)$ da iteração k resolvendo-se o sistema:

$$(G(x^k) + \mu.I)s^k = - g(x^k),$$

onde μ è um escalar positivo, e I è a matriz identidade de ordem n.

O ponto $x^k + s^k(\mu)$ minimiza a aproximação quadrática local de f na esfera definida com centro em x^k e raio $\| s^k(\mu)\|$. A esfera, conhecida como região de validade, è expandida e contraída de um modo controlado. Compara-se uma redução real e uma prevista de f; se a previsão è pobre (excelente), o parâmetro μ è aumentado (diminuido) para ajustar a esfera, e $s^k(\mu)$ è recalculado; senão, aceita-se $s^k(\mu)$ (ver Fletcher, 1980).

A direção $s^k(\varnothing)$ pode também ser vista como um compromisso entre as direções de Newton $n^k = s^k(0)$ e a de descida mais rápida $d^k = -g(x^k)$, pois

$$s^k(0) = -G^{-1}(x^k).g(x^k),$$

e

$$s^k(\varnothing) = -\varnothing^{-1}.g(x^k) \quad \text{(para valores grandes de } \varnothing\text{)}.$$

Tal compromisso é conveniente no caso de mal-condicionamento de $G(x^k)$. Nesse caso, os contornos da aproximação quadrática local de f são elipsóides estreitas e longas, de forma que n^k pode ser quase ortogonal à d^k (ver figura 7). Um lento progresso seria feito ao longo de n^k nesse caso.

Baseando-se nessas idéias, Patel (Patel, 1982) usou paralelismo em uma busca bi-dimensional na variedade linear gerada por n^k e d^k. Buscam-se aproximações a_k e b_k para um mínimo da função T_k definida por:

$$T(a, b) = f(x^k + an^k + bd^k)$$

Quando o número de processadores é grande, esse passo pode ser executado via avaliações simultâneas da função em um "grid" de pontos pré-estabelecido. O ponto do "grid" com o menor valor da função é tomado como uma aproximação do mínimo planar. Patel realizou uma série de experiencias numéricas com funções testes de dimensão 64 (n = 64) no ICL-DAP com 4096 processadores paralelos, comparando vários métodos de busca: a busca unidimensional em n^k, a busca bi-dimensional em n^k e d^k, e também busca quatro-dimensional na variedade linear gerada por n^k, d^k, $x^k - x^{k-1}$ e $x^{k-1} - x^{k-2}$. Porém os resultados obtidos no sistema ICL-DAP não foram conclusivos sobre a preferência de um tipo de busca em relação a outra. Uma análise matemática com essas idéias ainda está pendente.

Schnabel (Schnabel, 1986) retornando à idéia de cálculo do gradiente de f usando avaliações em paralelo da função em um dado ponto, apresentou três estratégias para uso de paralelismo na rede local ENCONP. Como Schnabel observou, em geral $\lambda = 1$ é uma boa aproximação para o mínimo em linha em um processo de busca unidimensional, isto é, muitas vezes na primeira tentativa aceita-se o ponto na linha. Assim um modelo típico de avaliações da função é: f, f, g, f, g, f, g, f, f, f, g, f, g, f, f, g, f, g (onde f, f significa que

o ponto não foi aceito na primeira tentativa, e g è uma avaliação do gradiente em paralelo). Assim, as três estratègias resumem-se em: (n = dimensão do problema, p = número de processadores)

Estratègia 1 - quando o gradiente no ponto teste è necessàrio, ele è calculado por diferenças finitas com n avaliações da função em paralelo ($\lceil n/p \rceil$ passos concorrentes de avaliação da função). Cada avaliação da função no ponto teste $f(x^k + \lambda.s)$ è executada por um processador e os restantes dos processadores não são usados.

Estratègia 2 - em cada instante que a função è avaliada no ponto teste, executam-se $\min\{p-1, n\}$ avaliações em paralelo da função para calcular o gradiente de f por diferenças finitas. Se $p<n+1$ e o ponto teste è aceito então calculam-se em paralelo as $n+1-p$ avaliações restantes para o càlculo do gradiente ($\lceil n+1-p/p \rceil$ passos concorrentes de avaliação da função).

Estratègia 3 - em cada instante que a função è avaliada no ponto teste, executam-se as n avaliações em paralelo da função para calcular o gradiente por diferenças finitas ($\lceil (n+1)/p \rceil$ passos concorrentes da avaliação da função).

A Estratègia 1 è idêntica a um algoritmo sequencial, exceto que o gradiente por diferenças finitas è calculado em paralelo. Quando somente o valor da função è calculado (isto è, não aceita-se o ponto teste), alguns processadores ficarão ociosos. Nas Estratègias 2 e 3 existem càlculos paralelos do gradiente em qualquer avaliação da função, o que eventualmente pode representar càlculo de informação que não será usada. mas que pode representar um uso adequado dos processadores disponiveis. Entretanto, para qualquer problema particular todos os três algoritmos produzirão a mesma sequencia de iterações que o algoritmo sequencial.

Outras experiências computacionais em torno de mètodos tipo Newton apareceram em Pierre (Pierre, 1973) e Mukai (Mukai, 1981).

3.2 - METODOS DE DIREÇÕES CONJUGADAS:

Os mètodos de direções conjugadas sempre foram sujeito a muitas pesquisas, por causa de seu potencial para resolver problemas grandes, com atè centenas de variàveis. Serà revisto aqui basicamente, um mètodo de direções conjugadas que não usa informações de gradiente (Chazan & Miranker, 1970).

A conjugação de dois vetores não-nulos, s e t, em relação a uma matriz definida positiva G(x) é a propriedade de que: $s^tG(x)t = 0$. Um método de direções conjugadas é aquele que gera tais direções quando aplicado a uma função quadrática de Hessiana G(x), isto é

$$f(x) = 1/2\ x^tG(x)x - b^tx.$$

Os seguintes teoremas proporcionam a base para os métodos em questão.

Teorema 1: Dados k vetores $s^1, s^2, \ldots, s^k$, mutuamente conjugados. Seja x^1 um ponto arbitrário, e seja x^{i+1} o ponto mínimo de f na direção s^i originando em x^i $(i=1,\ldots,k)$, então x^{k+1} minimiza f sobre a variedade linear que passa por x^1 gerada pelo conjunto de direções de busca $\{s^1, \ldots, s^k\}$.

Teorema 2: Seja L uma variedade linear em R^n, sejam M^1 e M^2 variedades lineares paralelas em R^n obtidas por "shifts" de L, e sejam x^1 e x^2 mínimos restritos de f(x) em M^1 e M^2 respectivamente; então, $x^1 - x^2$ é conjugado a qualquer $s \in L$.

O teorema 1 explica porque direções conjugadas são interessantes (minimiza-se f(x) sobre variedades lineares de dimensões constantemente crescentes), e o teorema 2 fornece um mecanismo para gerar direções conjugadas (pela minimização de f(x) sobre espaços paralelos geometricamente).

Um método conhecido, utilizando esses resultados é o método de direções conjugadas de Powell, resumido a seguir:

Dados em R^n um ponto p e m direções linearmente independentes $v^1, v^2, \ldots, v^m$;

Passo 1: Iniciando em p, fazer m minimizações unidimensionais, em sequencia, nas direções $v^1, v^2, \ldots, v^m$, respectivamente. Este processo produz uma trajetória poligonal que termina no ponto q.

Passo 2: Fazer mais uma minimização unidimensional, iniciando em q na direção $v^{m+1}=q - p$. Seja r o resultado dessa última minimização. Esta é uma iteração do algoritmo. Para a próxima iteração, atualiza-se p e $v^1, v^2, \ldots, v^m$, como segue: $r \longrightarrow p$ e $v^{i+1} \longrightarrow v^i$, $i=1,\ldots,m$ (ver Avriel, 1975).

Chazan & Miranker (Chazan & Miranker, 1970) modificaram o algoritmo de Powell para uma versão paralela, em que as buscas do passo 1 são substituidas por buscas paralelas, como segue:

Dados x^0 e um conjunto de n vetores linearmente independentes (múltiplos dos vetores unitários respectivos) $\xi^1, \xi^2, \ldots, \xi^n$;

Iteração 1 : Fazer $x^i = x^{i-1} + \xi^i$, $i=1, \ldots, n$, sem buscas lineares; toma-se $s = x^1 - x^0$ e executam-se simultaneamente n buscas lineares ao longo de s nos pontos $x^1, x^2, \ldots, x^n$. Essas buscas não são necessariamente tarefas idênticas e produzem os pontos $y^0, y^1, \ldots, y^{n-1}$.

Iteração 2: Gera-se o ponto y^n arbitrariamente por $y^n = y^{n-1} + \xi^1$, faz-se $t = y^1 - y^0$, e executam-se n buscas lineares ao longo de t nos pontos $y^1, y^2, \ldots, y^n$. Pelo teorema 2, as direções s e t são conjugadas, assim, os pontos minimos $z^0, z^1, \ldots, z^{n-1}$ produzidos na segunda iteração são minimos planares (minimizam f sobre espaços gerados por s e t).

Iteração 3: Gera-se um ponto de partida extra fazendo $z^n = z^{n-1} + \xi^2$. A direção de busca $u = z^1 - z^0$, conjugada de s e t, è usada para n buscas lineares de $z^1, z^2, \ldots, z^n$, que produzem os pontos minimos $v^0, v^1, \ldots, v^{n-1}$, respectivamente.

A figura 9 mostra um exemplo tridimensional do método de Chazan & Miranker, onde v^0 è o minimo da função quadràtica f. Para minimizar uma função não-quadràtica, continua-se o processo. Assim, na iteração 4 inicia-se com a geração de $v^3 = v^2 + \xi^3$. Estes deslocamentos são introduzidos para gerar arbitrariamente pontos de partida para os casos não-quadràticos. Os deslocamentos são empregados em ordem cíclica, uma por iteração, e são reduzidos na vizinhança de um minimo, para evitar passos grandes. Termina-se o processo iterativo, logo que os pontos finais de duas iterações sucessivas são "muito pròximos".

4 - CONCLUSOES:

Esta introdução ao processamento paralelo em Programação Não-linear mostrou parte do que existe nesta nova àrea de pesquisa, revisando-se somente o caso de

otimização local irrestrita. Schnabel (Schnabel, 1986) apresentou alguma pesquisa em métodos globais paralelos; e métodos restritos usando paralelismo são parcialmente introduzidos em Lootsma (Lootsma, 1986)

Os métodos apresentados são adaptações para máquinas SIMD e/ou MIMD de métodos existentes em otimização sequencial. Pode-se prever no futuro a criação de métodos dirigidos especialmente para máquinas paralelas, bem como a continuidade na pesquisa de adaptações dos métodos existentes.

A melhor contribuição para a área, talvez seja uma perfeita análise matemática dos algoritmos, não descartando porém os aspectos de implementação e comparação com os algoritmos sequenciais.

As várias questões abertas existentes, provavelmente levarão a muitos estudos comparativos não só no campo de otimização irrestrita, mas também em áreas correlatas.

5 - BIBLIOGRAFIA:

Avriel, M. *Nonlinear programming: analysis and methods* - Prentice-Hall Inc., Englewood Cliffs, 1976.

Chazan, D. & Miranker, W.L. *A nongradient and parallel algorithm for unconstrained minimization* - SIAM Journal on Control 8(2):207-217, 1970.

Dixon, L.C.W. & Patel, K.D. *The place of parallel computation in numerical optimization: IV paralel algorithms for nonlinear optimization* - NOC Technical Report 125, The Hatfield Polytechnic, U.K., February 1982.

Fletcher, R. *Practical methods of optimization, vol. 1: Unconstrained optimization* - John Wiley & Sons., New York, 1980.

Flynn, M.J. *Some computer organization and their effectiveness* - IEEE Transactions on Computers C-21:948-960, 1972.

Gill, P.E. & Murray, W. & Wright, M.H. *Practical optimization* - Academic Press, London, 1981.

Lootsma, F.A. *Parallel algorithms for unconstrained and constrained nonlinear optimization* - Report 86-15, Delft University of Technology, The Netherlands, 1986.

Mukai, H. *Parallel algorithms for solving systems of nonlinear equations* - Computers and Mathematics with Applications 7:235-250:1981.

Patel, K.D. *Parallel computation and numerical optimization* - NOC Technical Report 129, The Hatfield Polytechnic, October 1982.

Pierre, D.A. *A nongradient minimization algorithm having parallel structure, with implementation for an array processor* - Computational and Electrical Engineering 1:3-21, 1973.

Ribeiro, C.C. *Parallel computer models and combinatorial algorithms* - Monograph GSM-12, Catholic University of Rio de Janeiro, June 1985.

Rodrigue, G. *Parallel computations* - Academic Press, New York, 1982.

Schnabel, R.B. *Concurrent function evaluations in local and global optimization* - CS-CU-345-86, Department of Computer Science, University of Colorado, U.S.A., October 1986.

Tanenbaum, A.S. *Computer systems organization* - Prentice-Hall Inc., Englewood Cliffs, 1984.

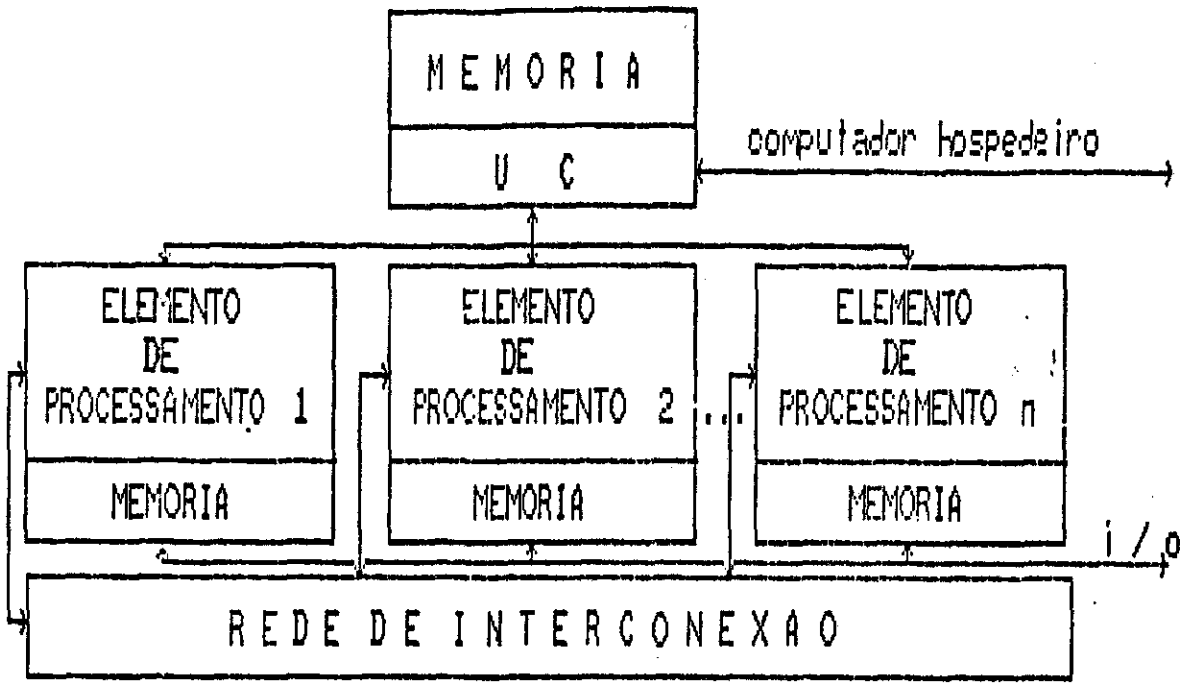


Figura 1: Arquitetura SIMD

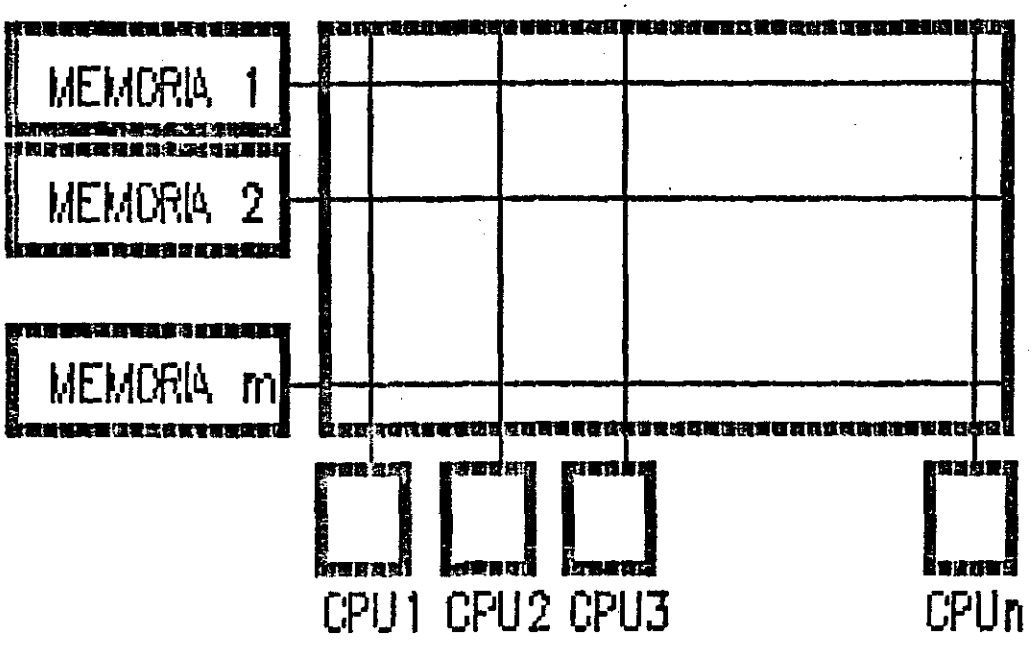


Figura 2: Diagrama de uma arquitetura MIMD

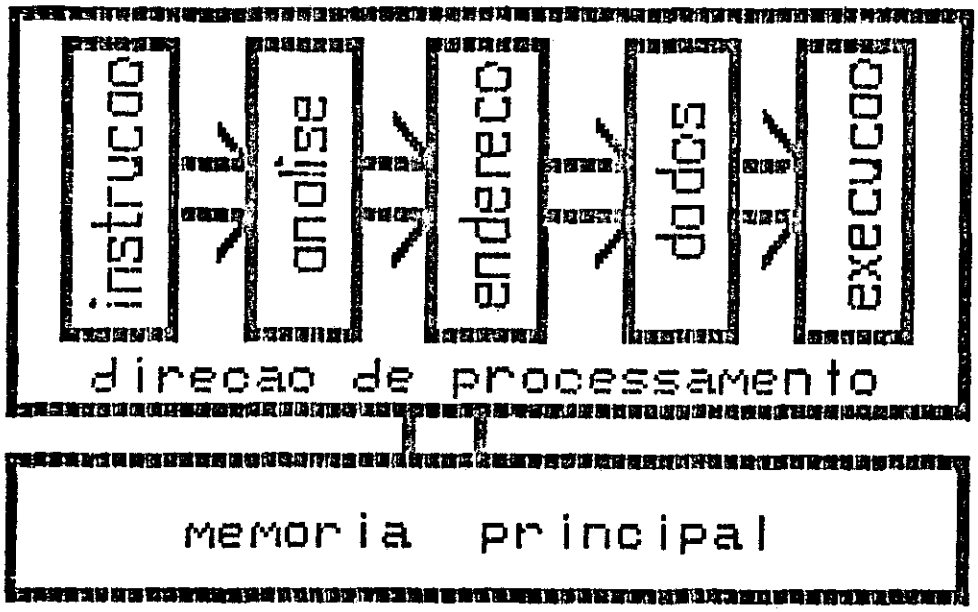
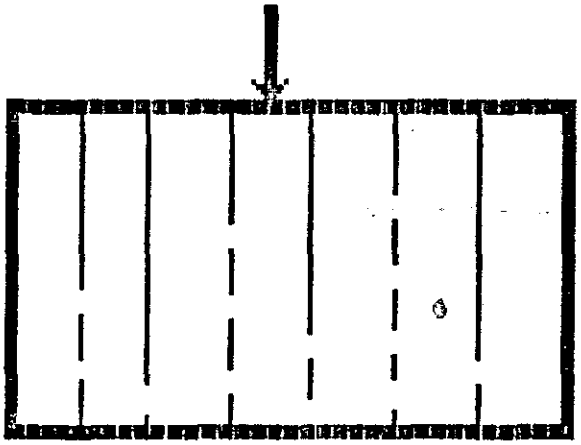


Figura 3: "pipelined"



processador trabalha
processador espera

figura 4:MIMD(sincronizacao)

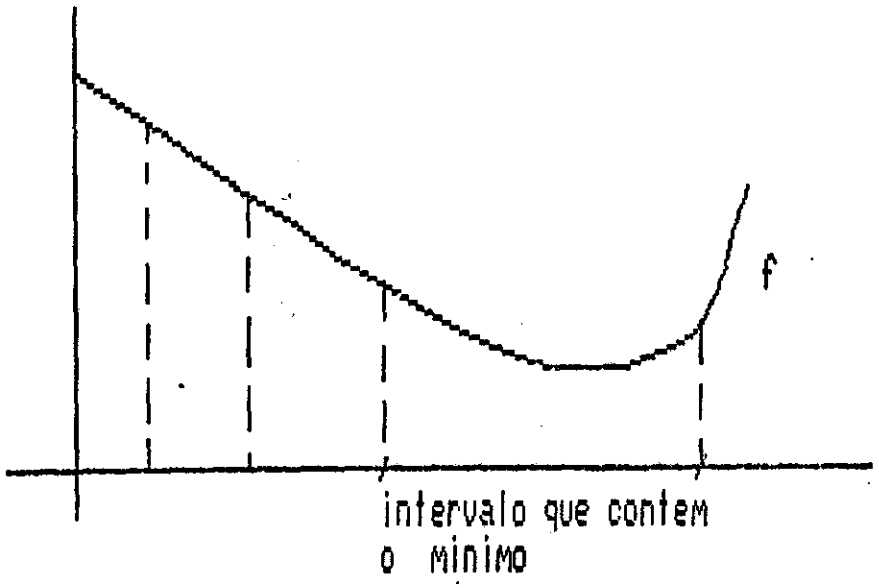


Figura 5: Enquadramento do passo usando um "grid" crescente de pontos

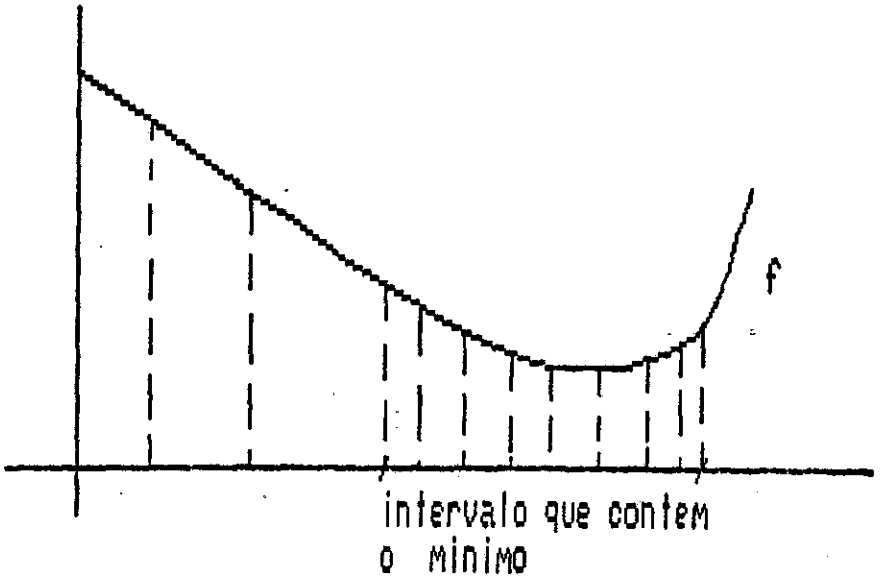


Figura 6: Um "grid" fino de pontos no intervalo que contem o minimo (passo de interpolacao)

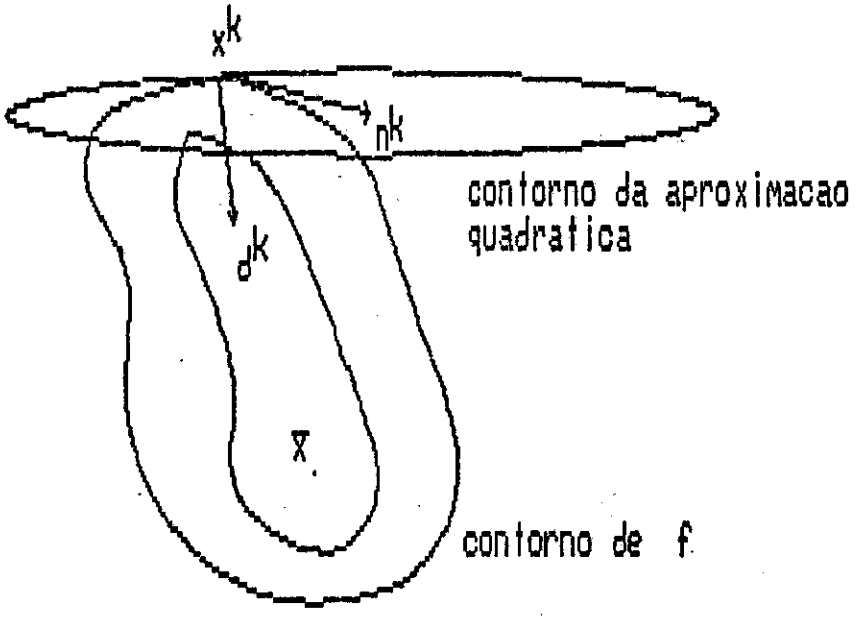


Figura 7: A direcao de Newton e a direcao de descida mais rapida em x^k sao quase ortogonais

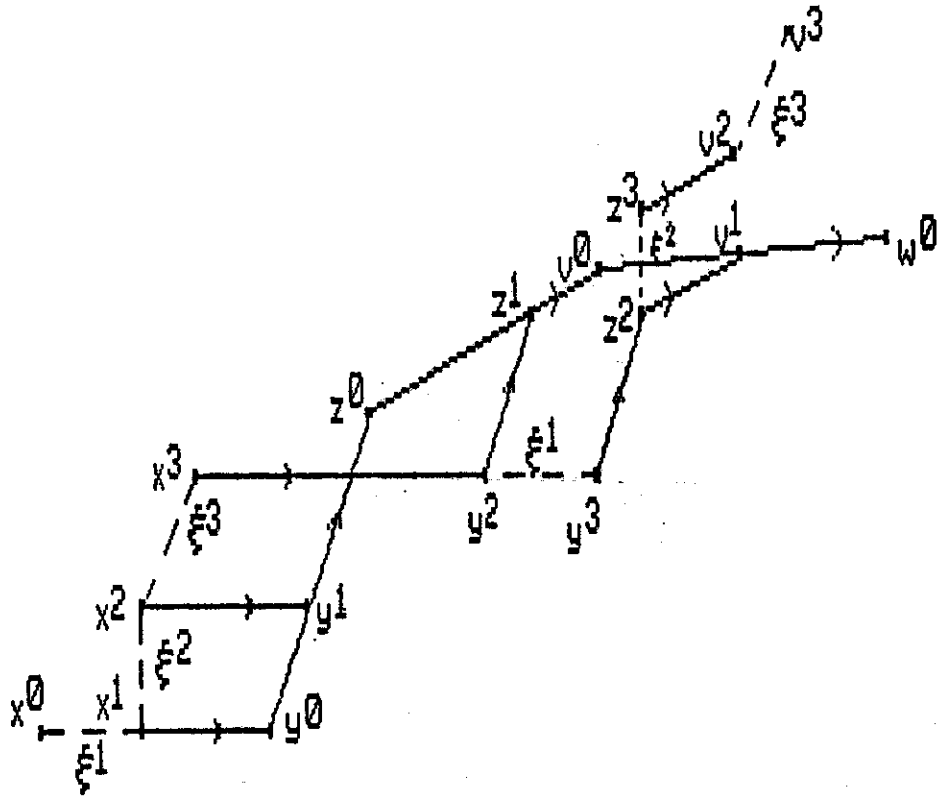


Figura 8: Metodo de Chazan & Miranker em tres dimensoes

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO DE PESQUISAS ESPACIAIS

PROPOSTA PARA PUBLICAÇÃO

☐ DISSERTAÇÃO
☐ TESE
☐ RELATÓRIO
☐ OUTROS

IDENTIFICAÇÃO

TÍTULO: PROCESSAMENTO PARALELO EM PROGRAMAÇÃO NÃO-LINEAR

AUTOR(ES): Ana Clara da Rocha
Luiz Antonio Nogueira Lorena

ORIENTADOR:

CO-ORIENTADOR:

DISS. OU TESE

LIMITE: __/__/__ | DEFESA: __/__/__ | CURSO: | ORGÃO:

DIVULGAÇÃO: ☒ EXTERNA ☐ INTERNA ☐ RESTRITA

EVENTO/MEIO: ☒ CONGRESSO ☐ REVISTA ☐ OUTROS

REV. TÉCNICA

NOME DO REVISOR: *

RECEBIDO: __/__/__ | DEVOLVIDO: __/__/__ | ASSINATURA:

APROVAÇÃO

NOME DO RESPONSÁVEL: Leon Sinay

APROVADO: ☒ SIM ☐ NÃO | DATA: 10/07/87 | ASSINATURA: LEON SINAY
Chefe do Laboratório Associado de Computação e Matemática Aplicada

REV. LINGUAGEM

Nº: | PRIOR.: | RECEBIDO: __/__/__ | NOME DO REVISOR:

PÁG.: | DEVOLVIDO: __/__/__ | ASSINATURA:

DATILOGRAFIA

OS AUTORES DEVEM MENCIONAR NO VERSO INSTRUÇÕES ESPECÍFICAS, ANEXANDO NORMAS, SE HOUVER

RECEBIDO: 9/8/87 | DEVOLVIDO: 10/8/87 | NOME DA DATILÓGRAFA:

Nº DA PUBLICAÇÃO: 4398-PRE/1169 PÁG.:
CÓPIAS: Nº DISCO: LOCAL:

AUTORIZO A PUBLICAÇÃO
☐ SIM
☐ NÃO __/__/__

DIRETOR

OBSERVAÇÕES E NOTAS

Trabalho a ser apresentado no XVI° Colóquio Brasileiro de Matemática. IMPA - Rio de Janeiro - 19 a 25 julho.

* Solicito dispensa de Rev. Técnica e Rev. de Linguagem.

p/ LEON SINAY
Chefe do Laboratório Associado de Computação e Matemática Aplicada

APÓS A PUBLICAÇÃO, ESTA FICHA SERÁ ARQUIVADA JUNTAMENTE COM A VERSÃO ORIGINAL

INPE 106