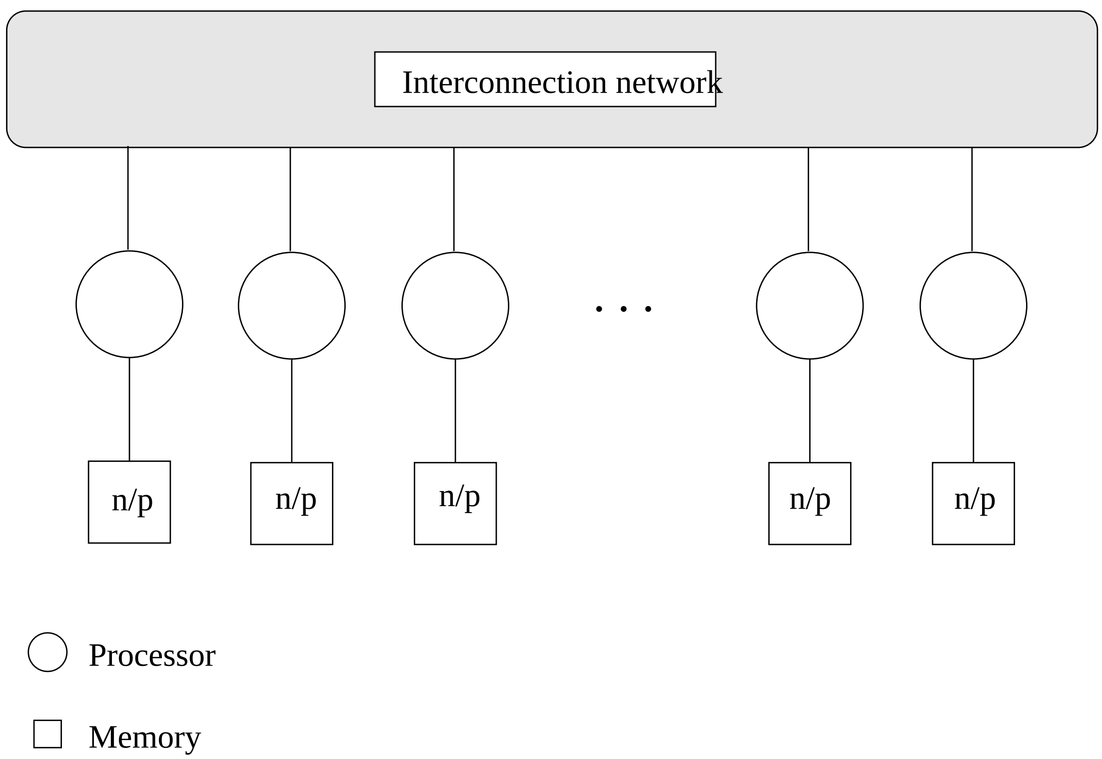
Efficient 1-D and 2-D Parallel Pattern Matching with Scaling

H. Mongelli and S. W. Song

26 de abril de 2004

CGM - Coarse-Grained Multicomputer



n: input size

p: p processors, $p << n$

Each processor with $O(n/p)$ memory

A CGM Algorithm

A CGM algorithm consists of:

A small number of alternating

- **Computing round**:

  each processor computes independently

- **Communication round**:

  each proc. may send n/p data

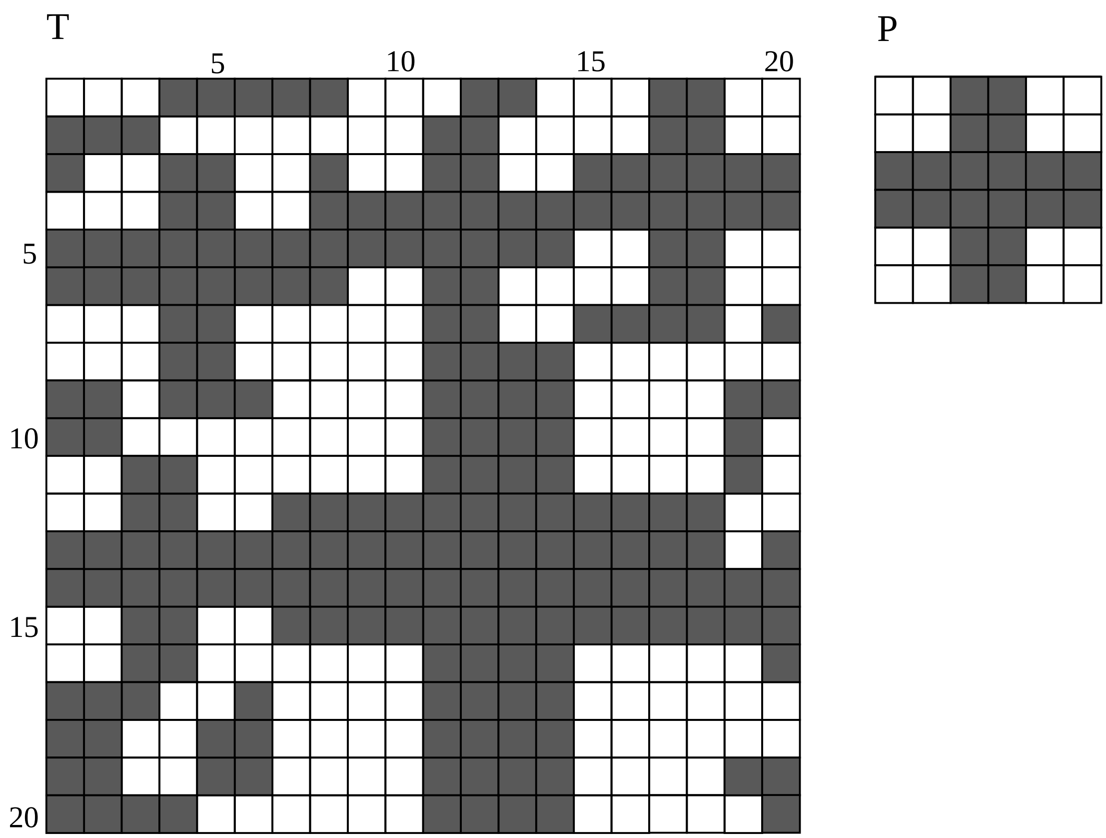
  and receive n/p data

GOAL: Minimize number of rounds

Objetivos

- Desenvolver um algoritmo CGM eficiente para o problema de busca bidimensional de padrões com escala.
- Obtivemos também algoritmos CGM eficientes para os problemas de busca unidimensional com e sem escala e bidimensional com escala.
- Estes algoritmos têm tempo de computação local linear e utilizam 1 rodada de comunicação.
- Nos algoritmos procuramos minimizar a quantidade de dados trocados nesta rodada de comunicação.

2-D Pattern Matching with scaling



- Given $N \times N$ Text T and $m \times m$ Pattern P
- Obtain all positions of T with k-occurrences of P, for all $k = 1, \ldots \lfloor N/m \rfloor$

Related Works

- Sequential algorithm: Amir, Landau, Vishkin 1992 - Linear time on the input size.
- Ferragna and Luccio 1999 - consider parallel string searching (without scaling).
- There are no known parallel algorithms for 2-D matching without and with scaling.

Aplicação

- A principal aplicação do algoritmo de busca bidimensional com escala é na busca de padrões em imagens onde podem existir alterações nas dimensões do padrão no texto a ser procurado.
- Estas alterações podem ser causadas por causa da distância da fonte de captação da imagem.

Publicações

- Mongelli, H. and Song, S. W. A range minima parallel algorithm for coarse grained multicomputers. IPPS'99/Irregular'99 - 6th. Intern. Workshop on Solving Irregularly Structured Problems in Parallel. Lect. Notes in Comp. Sc., Vol. 1586, José Rolim et al. (eds), Springer-Verlag. San Juan, Puerto Rico, April 12 - 16, 1999, pp. 1075 -1084.
- Mongelli, H. and Song, S. W. Parallel Range Minima on Coarse Grained Multicomputers. Intern. Journal of Foundations of Comp. Sc. Vol.10, No. 4, December 1999, pp. 375 - 389.

- Mongelli, H. and Song, S.W. Parallel String Matching with Scaling. Proc. 2001 Intern. Conf. on Parallel and Distributed Processing Techniques and Applications, Vol. 2, June 25-28, 2001, pp. 605-609.
- Mongelli, H. and Song, S. W. Parallel Pattern Matching with Scaling. Parallel Processing Letters. 2001. To appear.
- Mongelli, H. and Song, S. W. Efficient Two-Dimensional Parallel Pattern Matching with Scaling. *Proceedings 13th Intern. Conf. on Parallel and Distributed Computing and Systems.* August 21 - 24, 2001. To appear.

Publicações Relacionadas

- Problema do Mínimo Intervalar (Mongelli e Song, 1999, *International Journal of Foundations of Computer Science* e *Lecture Notes in Computer Science*)
- Problema de Buscas Múltiplas em Cadeias (Ferragina e Luccio, 1999, *Algorithmica*)
- Reconhecimento de Padrões Geométricos (Boxer, Miller e Rau-Chaplin, 1999, *Journal of Parallel and Distributed Computing*)
- Busca Unidimensional de Padrões com Escala Real (Amir, Butman e Lewenstein, 1999, *Information Processing Letters*)

Busca Unidimensional sem Escala

- Considere um texto T, de comprimento n, e um padrão P, de comprimento m. O *problema de busca de padrões unidimensional* consiste em determinar-se todas as posições em T onde o padrão P ocorre.
- O algoritmo mais conhecido para este problema e que leva tempo linear é o algoritmo KMP.
- No caso com escala, estamos interessados em determinar todas as ocorrências do padrão no texto, para todas as escalas inteiras k, com $1 \leq k \leq \lfloor n/m \rfloor$. Esse case será visto mais tarde.

Algoritmo KMP (Knuth Morris Pratt)

- Etapa 1: no pré-processamento ou análise do padrão uma função π é construída;
- Etapa 2: na análise do texto, as ocorrências de P em T são obtidas

Dado o padrão $P = \rho_1 \ldots \rho_m$

P_k denota o prefixo $\rho_1 \ldots \rho_k$, de k caracteres, do padrão P.

Se A e B são duas cadeias de caracteres, então $A \sqsupset B$ significa A é um sufixo de B.

Com esta notação, o problema de busca unidimensional de padrões é encontrar todas as posições s em T, $1 \leq s \leq n - m + 1$, tais que $P \sqsupset T_{s+m-1}$.

Algoritmo KMP - Análise do Padrão

Pré-processar o vetor P, comparando-o consigo mesmo, para construir uma função π, denominada *função prefixo*.

Esta função irá direcionar a busca das possíveis ocorrências do padrão no texto. Dado um padrão $P = \rho_1 \ldots \rho_m$, a função π para o padrão P é dada por:

$$\pi : \{1, 2, \ldots, m\} \rightarrow \{0, 1, \ldots, m-1\}$$

$$\pi[i] = \max\{k : k < i \text{ e } P_k \sqsupset P_i\}.$$

Isto é, $\pi[i]$ é o comprimento do maior prefixo de P que é um sufixo próprio de P_i.

Exemplo:

i	1	2	3	4	5	6	7	8	9	10
P_i	a	b	a	b	a	b	a	b	c	a
$\pi[i]$	0	0	1	2	3	4	5	6	0	1

Algoritmo para calcular função π

Os valores da função π serão armazenados em um vetor denominado também de π.

Entrada: o padrão P.
Saída: o vetor π de m posições.

1. $m \leftarrow comprimento(P)$
2. $\pi[1] \leftarrow 0$
3. $k \leftarrow 0$
4. **para** $q \leftarrow 2$ **até** m **faça**
5.   **enquanto** $k > 0$ **e** $P[k+1] \neq P[q]$ **faça**
6. $k \leftarrow \pi[k]$
7.   **se** $P[k+1] = P[q]$
8. $k \leftarrow k+1$
9. $\pi[q] \leftarrow k$
10. **devolva** π

Este procedimento leva tempo $O(m)$.

Busca 1-D sem Escala CGM

- Considere uma cadeia texto de comprimento N e uma cadeia padrão de comprimento m, $m \leq N/p$.
- O algoritmo CGM tem tempo local $O(N/p)$ e utiliza apenas uma rodada de comunicação em que são trocados $O(m)$ dados.
- Dois processadores rotulados i e j são ditos vizinhos se seus identificadors diferem de uma unidade.
- Dado um processador i, seu *vizinho esquerdo* é o processador $i-1$, para $1 \leq i \leq p-1$, e seu *vizinho direito* é o processador $i+1$, para $0 \leq i \leq p-2$.

Passos dos Algoritmos

1. Cada processador executa um algoritmo de busca de padrões seqüencial para os dados locais, armazenando as posições de ocorrência em uma lista R. (Tempo: $O(n/p)$)

2. Cada processador determina as *posições candidatas* na fronteira direita, armazenando-as em uma lista C_D. (Tempo: $O(m)$)

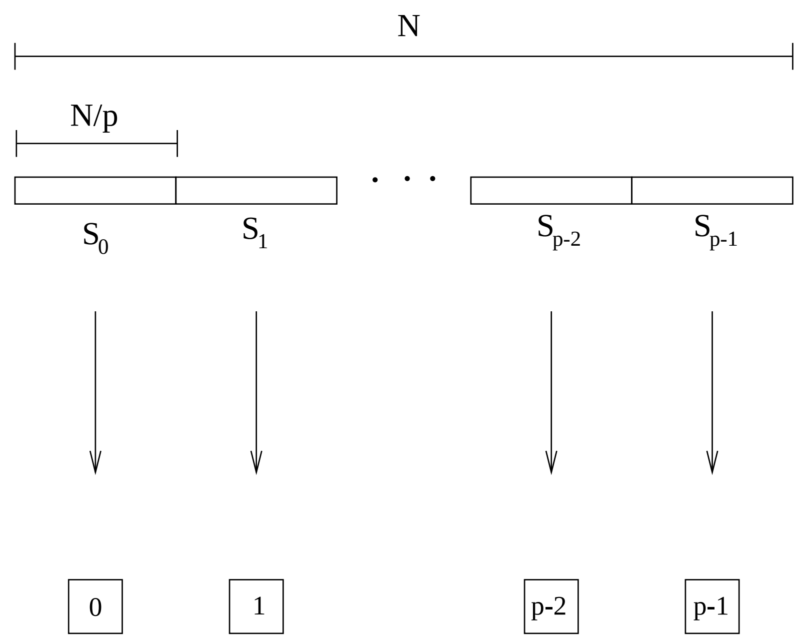
3. Cada processador obtém as *posições de confirmação* na fronteira esquerda, armazenando-as em uma lista C_E. (Tempo: $O(m)$)

4. A lista C_E é enviada para o vizinho esquerdo. (Comunicação: 1 rodada; Tamanho da Mensagem: $\leq O(m)$)

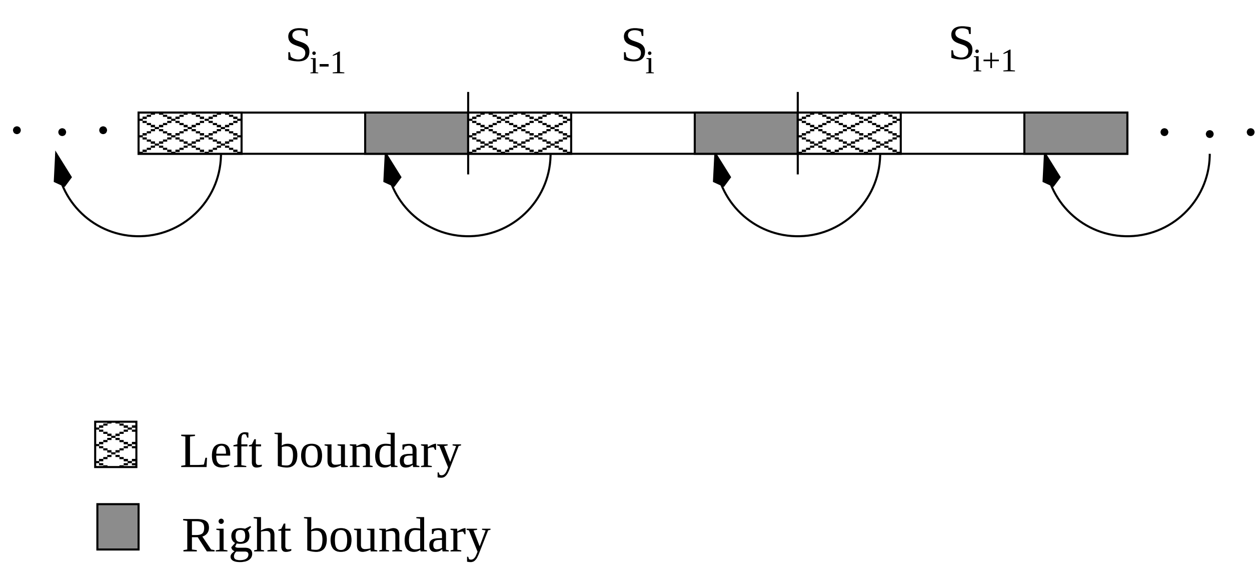
5. Cada processador combina C_D(local) com C_E(recebida) inserindo em R as posições de C_D que são ocorrências globais.

Distribuição de Dados e Comunicação

- **Distribuição de Dados**



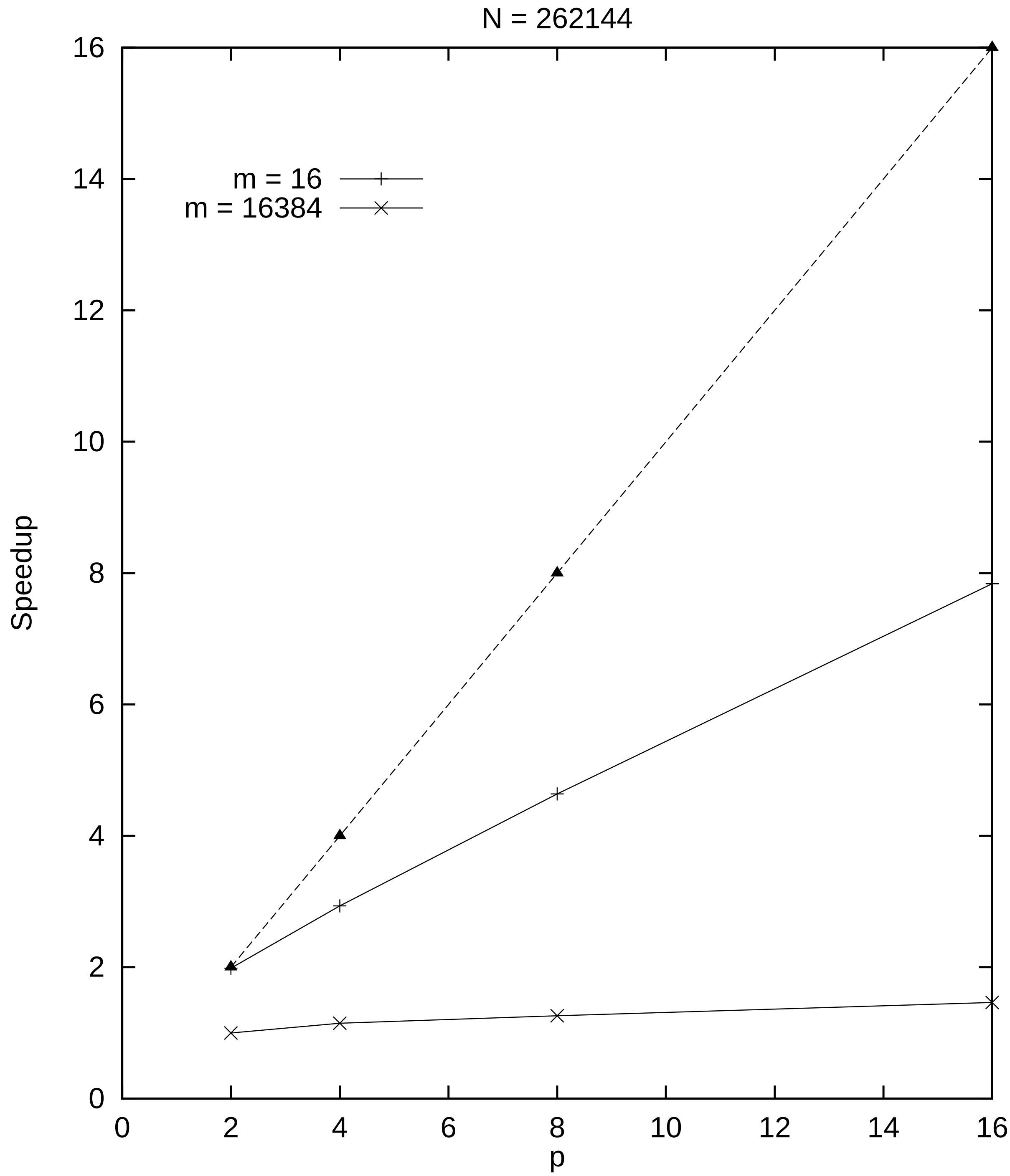
- **Comunicação**



Implementação

- Linguagem C
- *Interface PowerPVM*
- Máquina *Parsytec PowerXplorer*
  - 16 nós formados por um par *PowerPC 601 – Transputer T805*
  - 32 Mbytes de memória local
  - Grade bidimensional
  - Nós divididos em 4 partições de 4 nós

Speedup – Caso sem Escala



Busca Unidimensional com Escala

- Considere um texto T, de comprimento n, e um padrão P, de comprimento m. O *problema de busca de padrões unidimensional* consiste em determinar-se todas as posições em T onde o padrão P ocorre.

- No caso com escala, estamos interessados em determinar todas as ocorrências do padrão no texto, para todas as escalas inteiras k, com $1 \leq k \leq \lfloor n/m \rfloor$.

- Para este problema utilizamos as *representações fatoradas* das cadeias texto e padrão.

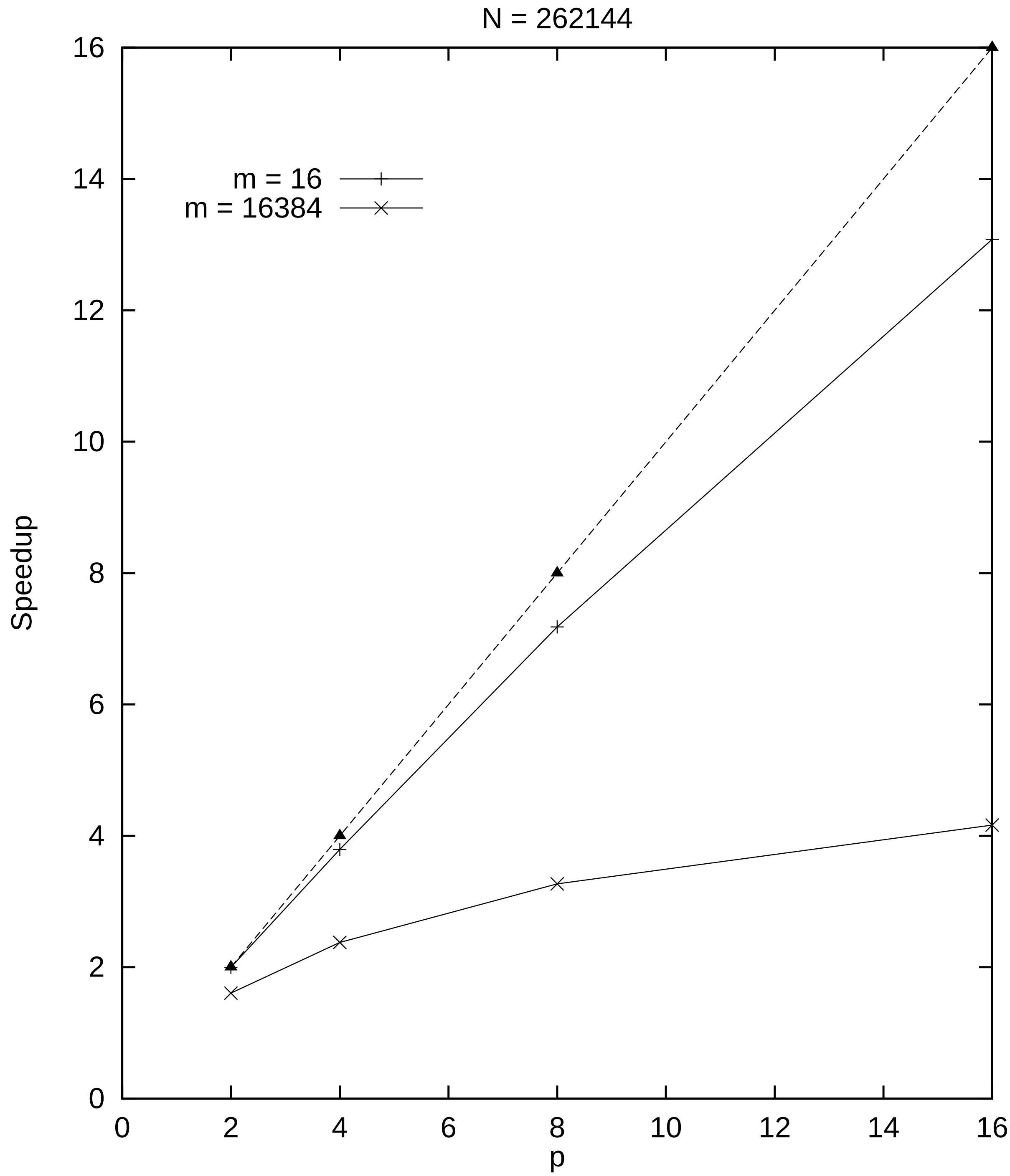
- Seja $S = \sigma_1 \sigma_2 \cdots \sigma_n$ uma cadeia em um alfabeto Σ. A *representação fatorada* da cadeia S é a cadeia $S' = \sigma_1'^{t_1} \sigma_2'^{t_2} \cdots \sigma_{\hat{n}}'^{t_{\hat{n}}}$ tal que:
  1. $\sigma_i' \neq \sigma_{i+1}'$, for $1 \leq i < \hat{n}$; e
  2. S pode ser descrita como uma concatenação do símbolo σ_1' repetido t_1 vezes, do símbolo σ_2' repetido t_2 vezes, $\ldots$, e do símbolo $\sigma_{\hat{n}}'$ repetido $t_{\hat{n}}$ vezes.

- Denotemos por $S^{\Sigma} = \sigma'_1 \sigma'_2 \cdots \sigma'_{\hat{n}}$, a *parte de símbolos* de S e por $S^{\#}$ o vetor de números inteiros $t_1, t_2, \ldots, t_{\hat{n}}$, a *parte de expoentes* of S.

- Nós também construímos as *cadeias quociente* $T' = \{t_2/t_1, t_3/t_2, \ldots, t_{\hat{n}}/t_{\hat{n}-1}\}$, de $\hat{n}-1$ elementos, onde $t_i \in T^{\#}$ e $P' = \{p_3/p_2, p_4/p_3, \ldots, p_{\hat{m}-1}/p_{\hat{m}-2}\}$, de $\hat{m}-3$ elementos, onde $p_i \in P^{\#}$

- O algorimto para o caso com escala utiliza as cadeias T^{Σ}, $T^{\#}$ e T', e P^{Σ}, $P^{\#}$ e P', e o algoritmo KMP nelas.

Speedup – Caso com Escala



Busca Bidimensional sem Escala

Usa os conceitos de *árvore digital de busca*, *árvore de sufixo* e LCA *lowest common ancestor*.

Árvore digital de busca ou *trie* T associada a um conjunto de cadeias distintas $C_1, C_2, \ldots, C_m$. Nenhum C_i é um prefixo de algum C_j, $i \neq j$.

A árvore digital de busca T é uma árvore com raiz com m nós terminais tal que:

1. Cada aresta de T está rotulada com um símbolo do alfabeto Σ, e está orientada no sentido raiz para nós terminais.

2. Quaisquer duas arestas que saem de um mesmo nó têm rótulos diferentes.

3. Cada folha u corresponde a uma e somente uma cadeia C_i, de forma que a concatenação dos rótulos das arestas do caminho da raiz até u seja C_i.

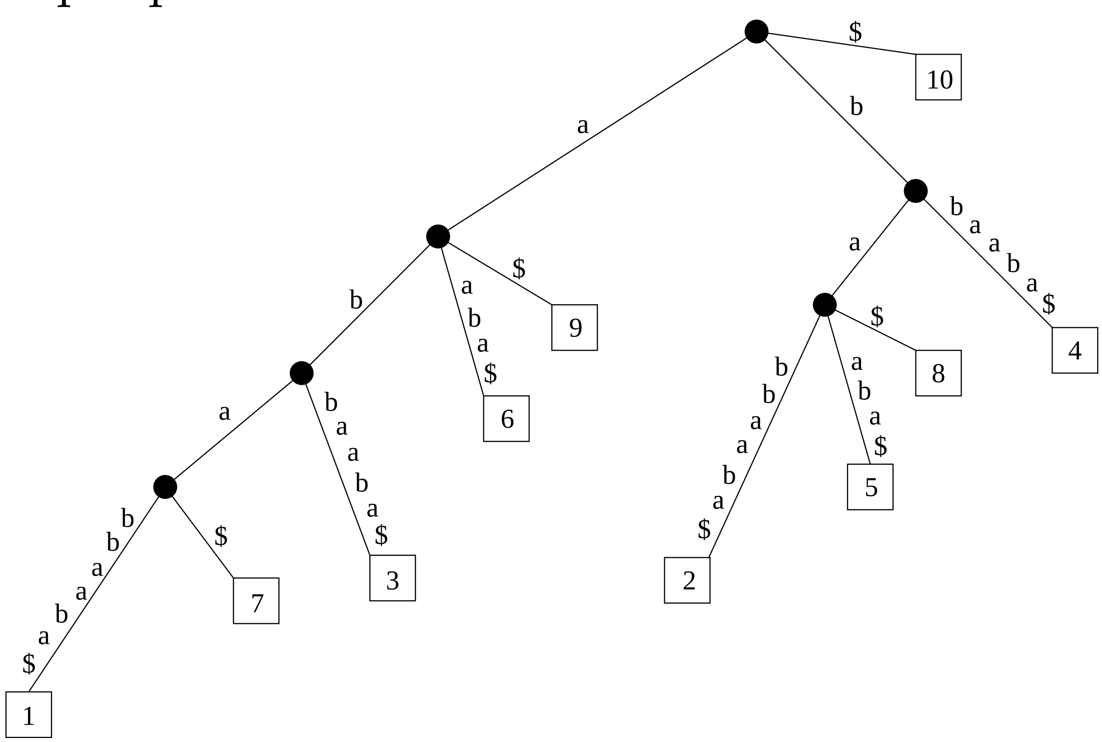
Nosso interesse é construir tal árvore para todos os sufixos de uma cadeia S dada.

Árvore de Sufixos

Uma árvore de sufixos T representando uma cadeia S obedece às seguintes restrições:

1. Uma aresta de T pode representar qualquer subcadeia não-vazia de S.
2. Cada nó não terminal de T, exceto a raiz, deve ter no mínimo dois arcos filhos.
3. As cadeias representadas por nós irmãos de T devem começar com caracteres diferentes.

Exemplo para a cadeia *ababbaaba*\$.



Construção em tempo linear Weiner 1973, McCreight 1976.

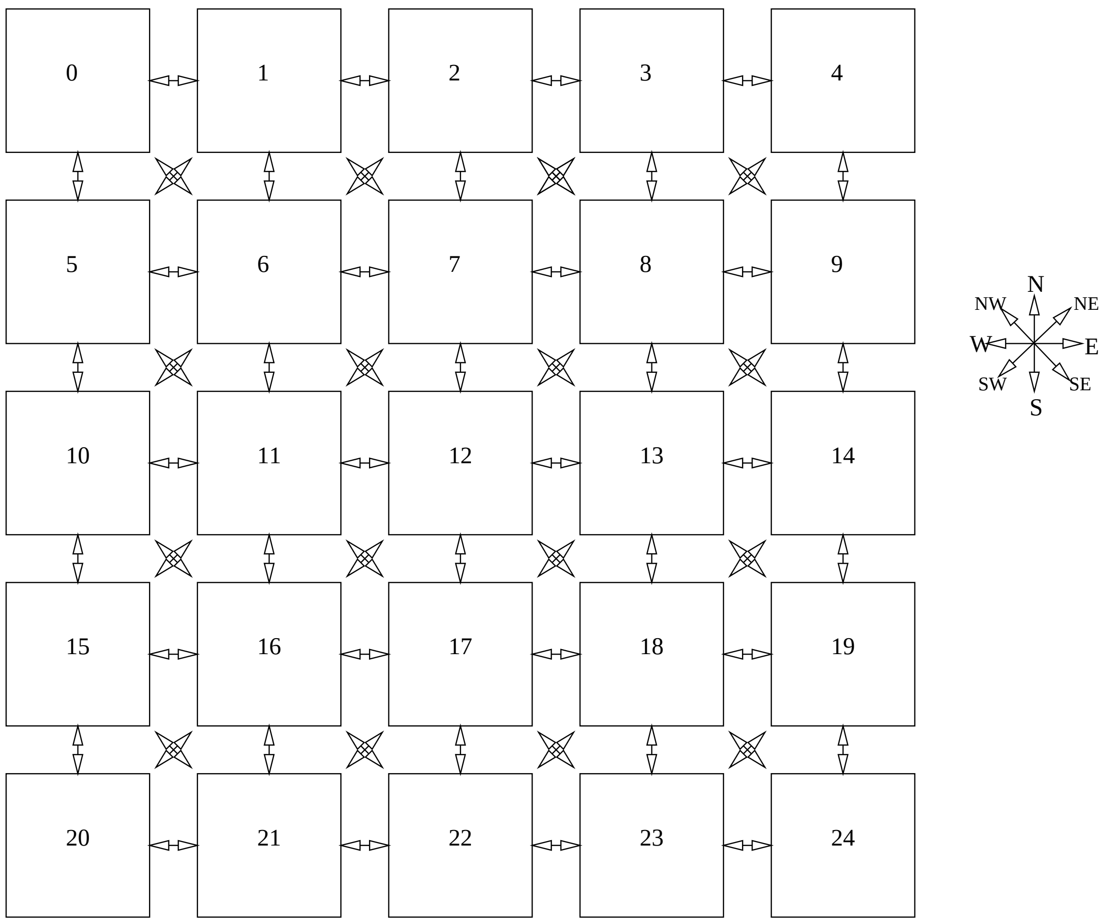
Busca Bidimensional sem Escala

- A idéia deste algoritmo é aplicar, para cada coluna da matriz texto, uma busca unidimensional.
- Em cada coluna j do texto, comparam-se as linhas do padrão com as sublinhas do texto de comprimento m que começam neste coluna.
- Para isto, as linhas do padrão e as sublinhas do texto de comprimento m são considerados como sendo símbolos.
- O tempo de um algoritmo descrito desta forma é $O(n^2 + m^2)$, para matrizes texto de ordem n e padrão de ordem m.
- Este tempo é obtido ao garantirmos que a comparação entre uma linha do padrão e uma sublinha do texto seja feita em tempo constante, para isto fazemos um pré-processamento da entrada.

- Este pré-processamento consiste em:
  - Construir uma cadeia C formada pela concatenação de todas as linhas de T e de todas as linhas de P.
  - Construir a árvore sufixa ST de C.
  - Processar esta árvore para consultas LCA.
- Com isto, a comparação entre uma linha de P e uma sublinha de T consiste de uma consulta LCA na árvore ST, que é feita em tempo constante.
- A análise do padrão é a mesma do caso unidimensional considerando cada linha do padrão como sendo um símbolo.

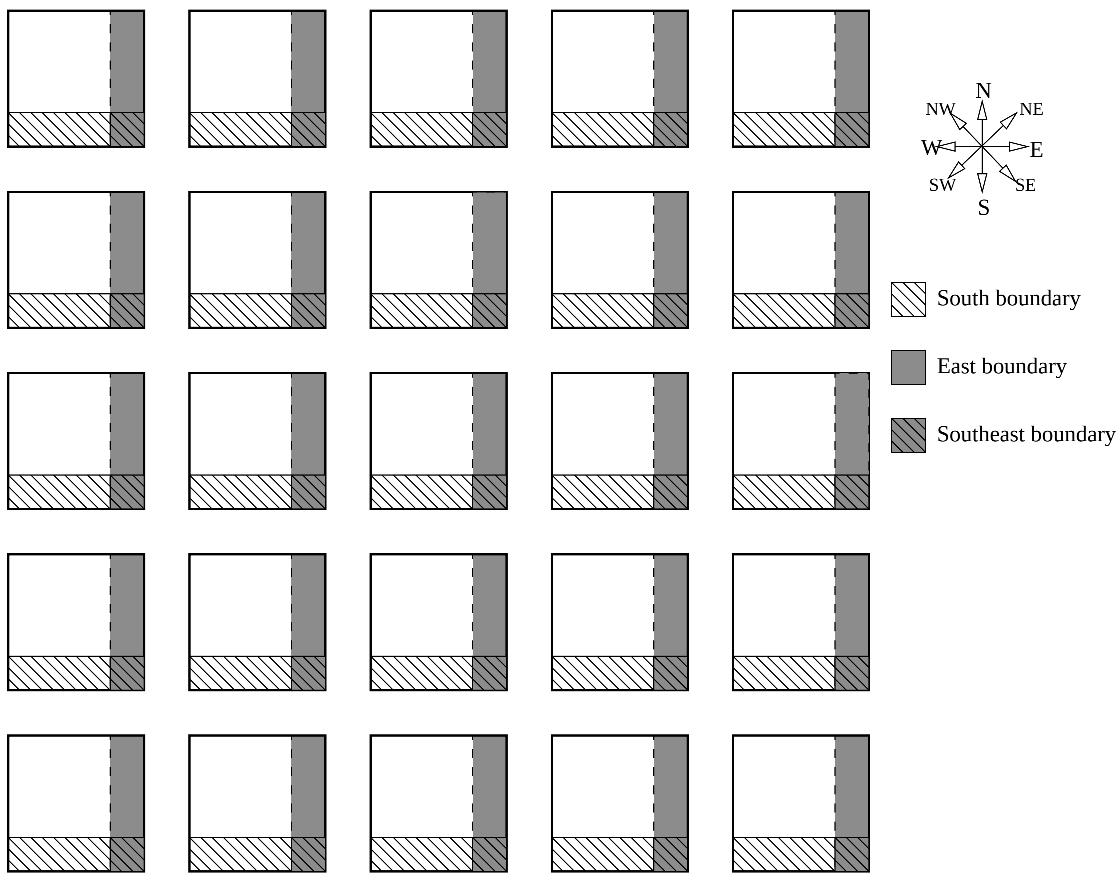
O Algoritmo CGM

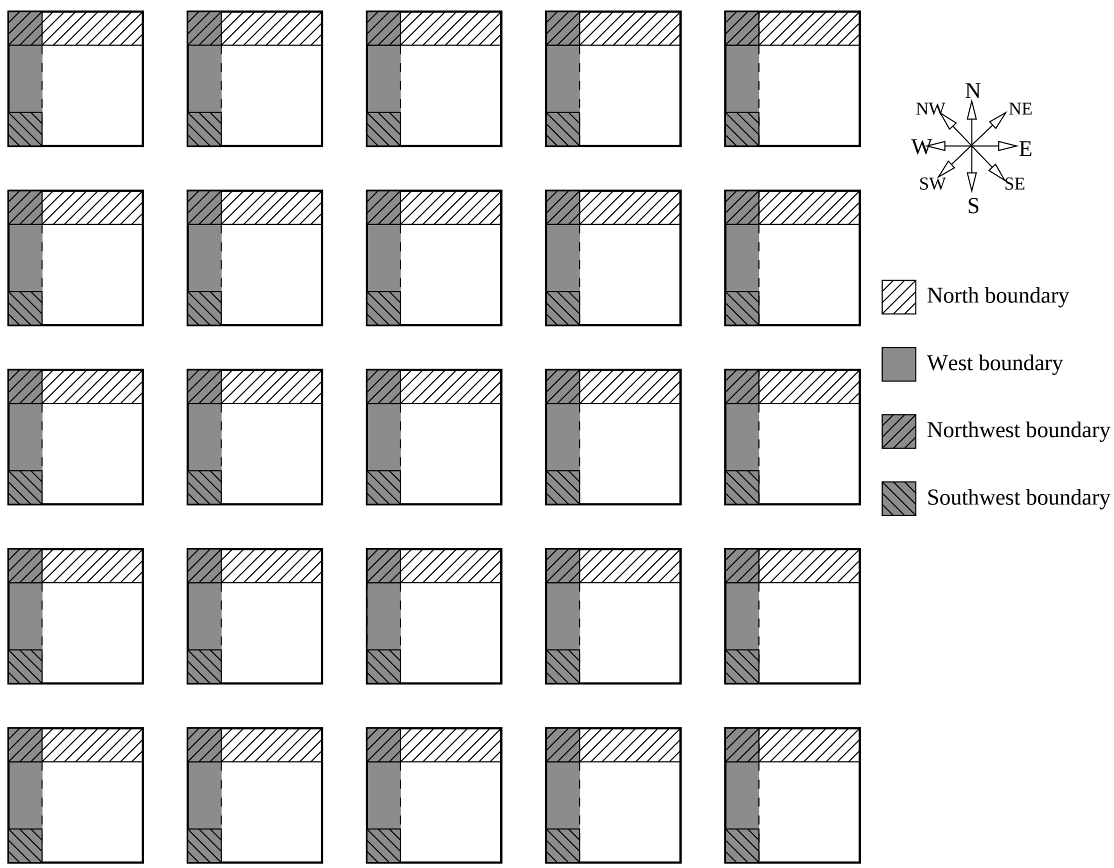
- Distribuição dos dados.



Cada processador armazena uma submatriz de ordem $n = N/\sqrt{p}$.

- Fronteiras


N
NE
E
SE
S
SW
W
NW
South boundary
East boundary
Southeast boundary


N
NE
E
SE
S
SW
W
NW
North boundary
West boundary
Northwest boundary
Southwest boundary

O Algoritmo

- Cada processador inicialmente executa o algoritmo seqüencial em seus dados locais, armazenando as posições de ocorrência em uma lista R.

- Cada processador, conforme seu índice, obtém *posições candidatas* nas fronteiras S e/ou E e/ou SE, armazenando estas posições em uma lista.

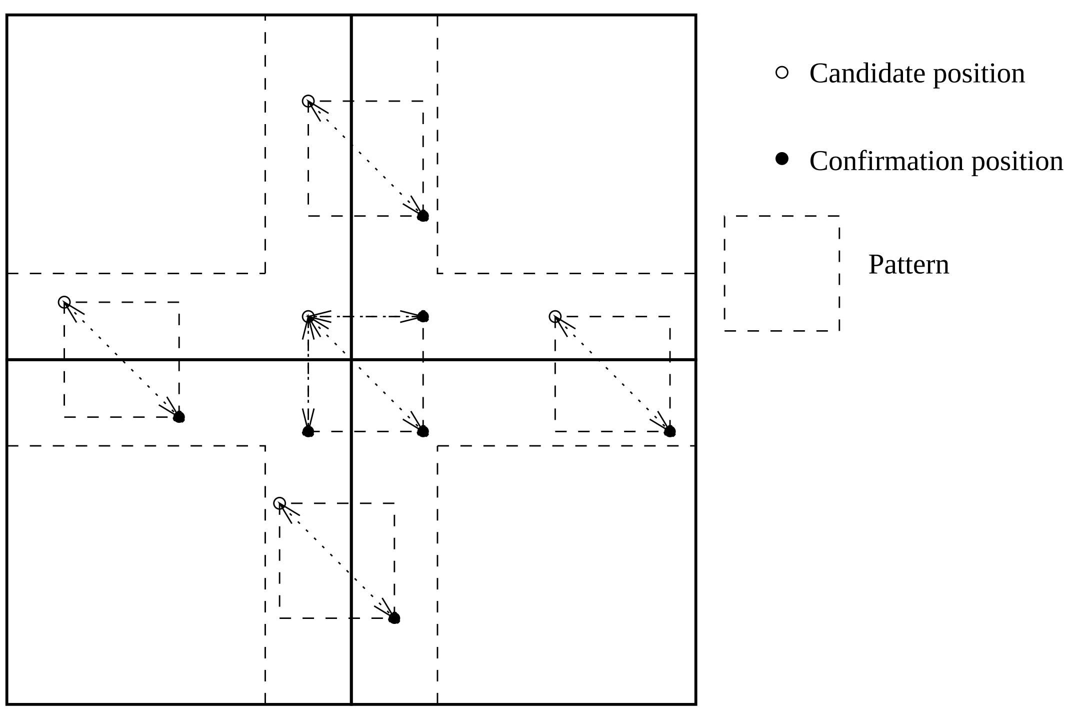
- Cada processador, conforme seu índice, obtém *posições de confirmação* nas fronteiras O e/ou N e/ou NO e/ou SO, armazenando-as em uma lista.

- As posições de confirmação são enviadas para os respectivos vizinhos.

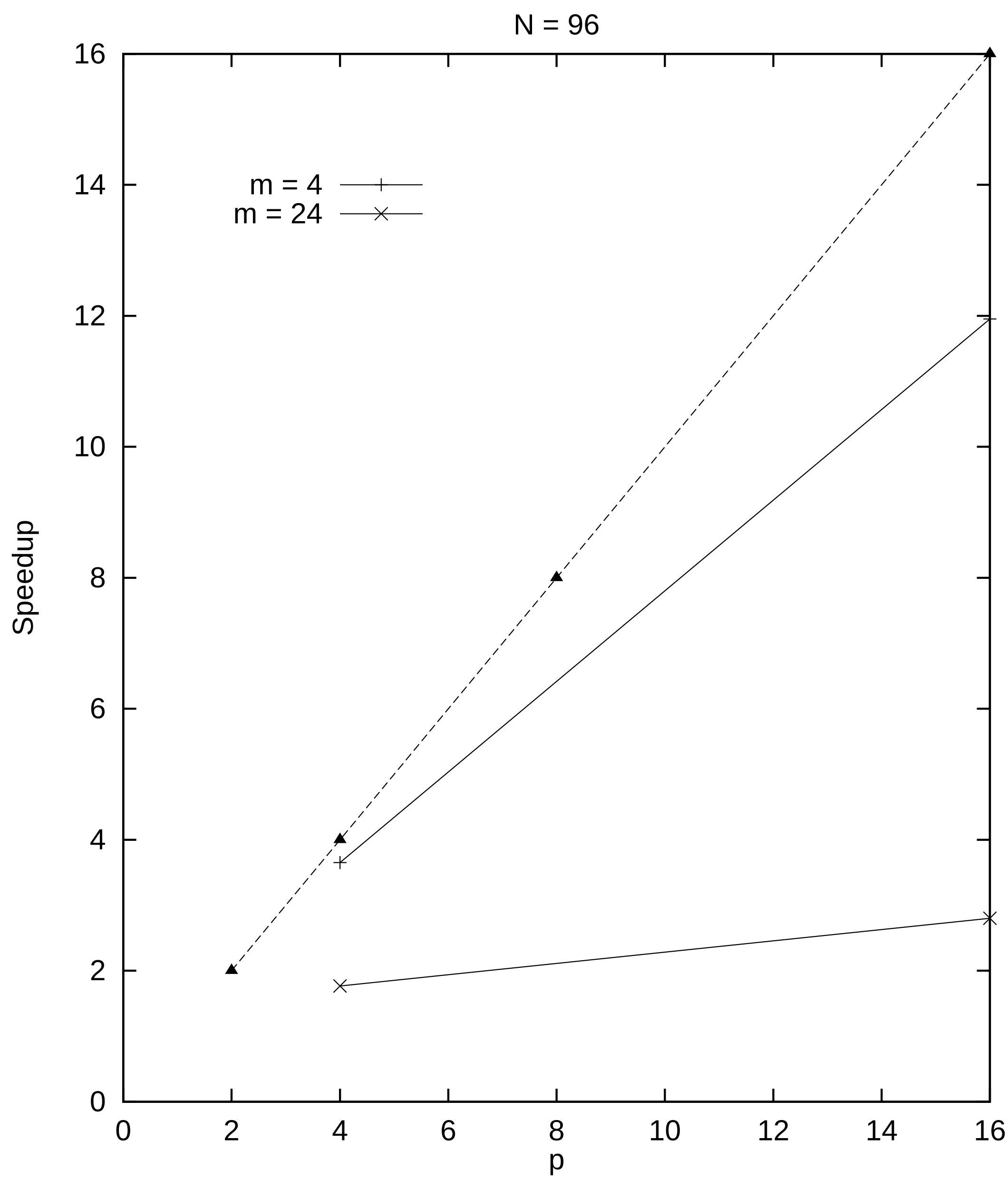
- Localmente, as posições candidatas e as de confirmação recebidas são combinadas através de um procedimento *Combina* e inseridas na lista R.

- Este algoritmo é executado em tempo local $O(n^2) = O(N^2/p)$, consome memória $O(n^2)$, utiliza uma rodada de comunicação em que são enviados no máximo $O(n^2)$ dados.

- A relação entre as posições candidatas locais e as de confirmação recebidas é dada pela figura:



Speedup



Busca Bidimensional com Escala

- Este problema consiste em obter todas as posições de T onde existam k-ocorrências de P, para todo $k = 1, \ldots \lfloor N/m \rfloor$, onde T tem ordem N e P tem ordem m, $m \leq N$.
- O algoritmo seqüencial tem tempo $O(N^2)$.
- A linearidade do algoritmo é obtida:
  - reduzindo-se, para cada escala k, o número de colunas nas quais pode existir o início de uma k-ocorrência e
  - dentro destas colunas reduz-se a quantidade de posições onde pode existir uma k-ocorrência.

O Algoritmo Seqüencial

- Seja $k > 0$ um inteiro.
  $T' = T[i_1, \ldots, i_2; j_1, \ldots, j_1 + km - 1]$ é um **k-bloco** na posição i_1 de uma coluna j_1 de T se:
  1. todas as linhas de T' são iguais;
  2. nenhuma linha pode ser adicionada a T' sem corromper a condição 1.
- A **altura** de tais k-blocos é $i_2 - i_1 + 1$.
- Se um k-bloco está contido em uma k-ocorrência então sua altura deve ser no mínimo k.

- Entrada:
  - a representação fatorada FT da matriz texto T e
  - a representação fatorada FP da matriz padrão P.
- As principais etapas do algoritmo são:
  1. Construção das estruturas de dados (cadeia C e árvore sufixa).
  2. Análise do Padrão.
  3. Análise do Texto.
- As k-ocorrências são determinadas, para cada escala k, separadamente.

Construção das Estruturas de Dados

- Constrói-se uma cadeia C obtida pela concatenação de todas as linhas de FT seguidas pelas $\lfloor N/m \rfloor$ seguintes cadeias:
  1. uma concatenação das linhas de FP.
  2. para cada k, $2 \leq k \leq \lfloor N/m \rfloor$, uma concatenação das linhas de P, onde cada linha é escalada a k, e dada em sua representação fatorada.

Análise do Padrão

- Na análise do padrão obtemos uma cadeia R formada pela concatenação das linhas fatoradas $FP_1, FP_2, \ldots, FP_m$.
- Comprimimos esta cadeia R, obtendo a representação fatorada FR.
- Consideramos a cadeia $\hat{F}R$ obtida da seqüência FR desconsiderando-se o primeiro e o último elementos.
- Para esta cadeia, obtemos o vetor π.

Análise do Texto

- Conforme os dados do padrão temos dois casos a considerar:

  Caso A. Em quaisquer $m/2$ linhas sucessivas do padrão existe pelo menos uma linha de comprimento fatorado maior do que 1.

  Caso B. Caso contrário. Isto é, existem pelo menos $m/2$ linhas consecutivas do padrão com comprimento fatorado 1.

Caso A

- Para este caso, os principais passos são:
  - Determinação das listas de finais de k-blocos.
  - Pré-processamento adicional.
  - Busca do padrão no texto.

Listas de Finais de k-Blocos

- Para cada escala k e cada coluna potência c, obtemos as colunas j onde podem existir k-ocorrências.
- Em cada uma destas colunas j construímos listas I_j^k de intervalos que contenham finais de k-blocos.
- Para cada lista I_j^k de intervalos, determinamos uma lista Q_j^k de finais de k-blocos.

Busca do Padrão no Texto

- Para cada escala k, utilizamos as listas Q_j^k de finais de bloco para percorrer as colunas, comparando-as com o padrão e assim poder determinar as k-ocorrências de $\hat{F}R$.
- Os k-blocos antes e depois destas k-ocorrências são verificadas quanto à extensão a k-ocorrências de FR no texto.

Caso B

- Os principais passos para este caso são:
  1. Para uma linha P_i de P convenientemente determinada, obter as k-ocorrências de P_i em todas as linhas de T. (Neste caso temos uma busca unidimensional com escala de P_i em cada linha de T.)
  2. Algumas posições de k-ocorrência de P_i são descartadas, conforme os subcasos.
  3. Para cada posição de k-ocorrência de P_i, verificamos se existe um posição de ocorrência de P em T.

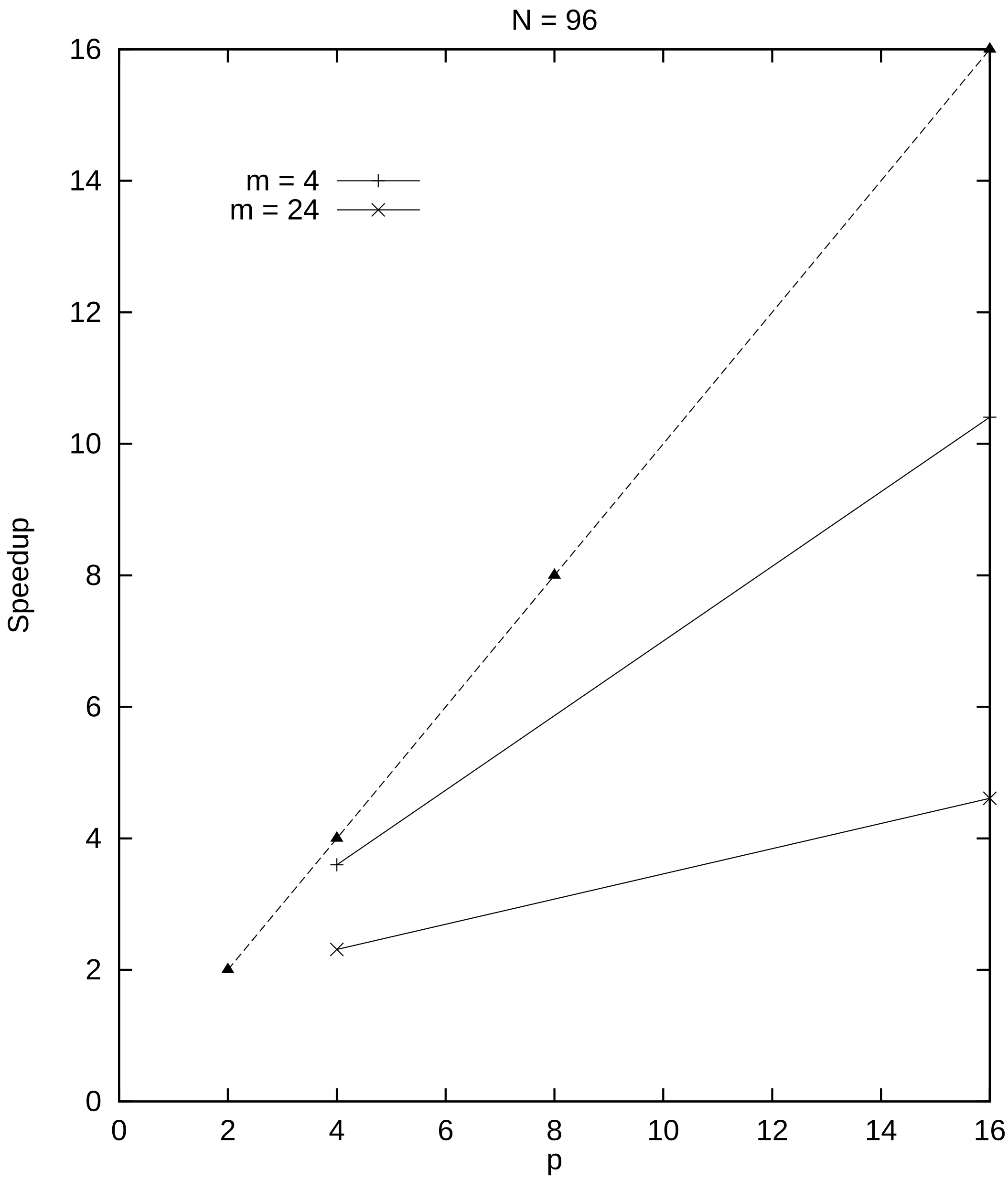
O Algoritmo CGM

- A distribuição dos dados é a mesma dos casos sem escala.
- Para uma matriz texto de ordem N e uma matriz padrão de ordem m, este algoritmo:
  - é executado em tempo local $O(N^2/p)$, consumindo memória $O(N^2/p)$,
  - utiliza 1 rodada de comunicação onde são enviados $O(N^2/p)$ dados.

Algoritmo

- Cada processador obtém as k-ocorrências dos dados armazenados localmente, guardando em uma lista R as posições e o valor de k.
- Cada processador, conforme seu índice, determina as posições de confirmação nas fronteiras N e/ou NO e/ou O e/ou SO.
- Estas posições são enviadas para os respectivos vizinhos juntamente com algumas informações adicionais.
- Cada processador, conforme seu índice, determina as posições candidatas nas fronteiras E e/ou S e/ou SE.
- As posições de confirmação recebidas e as candidatas localmente calculadas são combinadas e as posições candidatas confirmadas são inseridas juntamente com os respectivos valores de k na lista R.

Speedup



Conclusões

- Apresentamos algoritmos seqüenciais de tempo linear para todos os problemas e um sublinear para a busca bidimensional sem escala.
- Os algoritmos CGM propostos tem tempo local linear e utilizam 1 rodada de comunicação. A memória consumida e a quantidade de dados comunicados obedecem as restrições do modelo.
- Os resultados experimentais mostraram *speedups* consideráveis para os problemas.
- Obtivemos ainda um algoritmo CGM inédito para o problema de mínimo intervalar.
- Alguns trabalhos futuros podem ser desenvolvidos na implementação dos algoritmos, podendo obter *speedups* ainda melhores.

- Os *speedups* podem ser melhorados considerando-se que os dados utilizados nos testes correspondiam aos casos envolvendo a maior quantidade possível de dados na comunicação.
- Um algoritmo para o caso unidimensional com escala real foi recentemente publicado, não existindo ainda algoritmos paralelos para este caso.