Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
1º Semestre de 2013

Trabalho Prático 4 - Paralelização do Problema de Alocação de Canais em redes de Telefonia Celular

Esse trabalho prático tem como objetivo a implementação de uma solução paralelizada para o problema de alocação de canais em redes de telefonia celular, cujo objetivo é minimizar o número de canais necessários.

Problema

A empresa *previousTel* precisou expandir o número de ERBs na cidade de Belo Horizonte, devido a alta demanda ocasionada pela falência de um concorrente local. A *previousTel* ganhou um processo de licitação da *Anatel* relativa a licença de uso dos canais antes alocados para este concorrente. Você como programador da *previousTel* terá que utilizar de recursos de Programação Paralela para otimizar o algoritmo proposto para alocar o menor número de canais possíveis.

O problema de alocação dos canais continua sendo, dado um conjunto de *ERB*s com um determinado raio de atuação, determinar o menor número de licenças de canais que devem ser comprados, de modo que *ERB*s que possuem áreas de cobertura comum (possibilidade de interferência) não sejam alocadas para funcionarem no mesmo canal. Será utilizada a versão ótima do TP2 para a implementação da paralelização.

A ideia geral de paralelizar o seu algoritmo, consiste em dado um conjunto de possíveis canais, será disparada uma *thread* para cada canal, começando do menor para o maior número. Desse modo, deverá haver comunicação entre as *threads*, pois várias poderão retornar respostas satisfatórias. A *thread* que retornar resposta positiva com o menor número de canais para alocar as ERBs conterá a solução para o problema, conforme ilustrado na Figura 1.

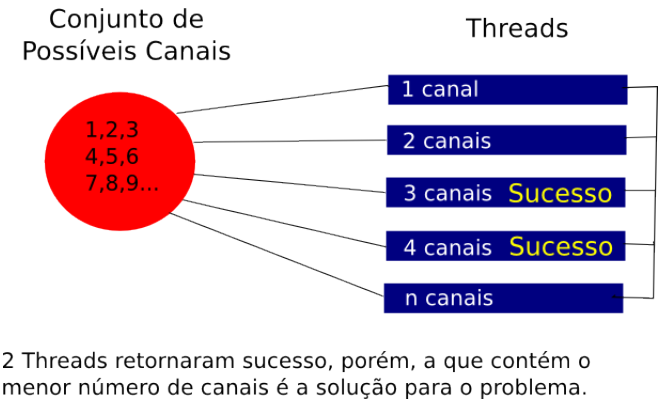

Figura 1: Exemplo de Paralelização de Alocação de Canais

Entrada e Saída

O programa deverá solucionar múltiplas instâncias do problema em uma única execução. Serão passados na entrada de dados informações para a construção do grafo que representará a rede de telefonia celular da cidade de Belo Horizonte e Região Metropolitana. A saída será dada pelo número de componentes conectados (lembre-se que um vértice isolado também conta como um componente), bem como o número mínimo de canais utilizados para cobrir a alocação de todas antenas. A entrada será lida de um arquivo e o resultado do programa deve ser impresso em outro arquivo de saída. Ambos arquivos devem ser passados por parâmetro na chamada do executável:

./tp4 input.txt output.txt

O arquivo de entrada possui um inteiro N na primeira linha onde N é o número de instâncias a serem simuladas. Em seguida, as N instâncias são definidas da seguinte forma. A primeira linha possui um inteiro M indicando o número de *ERB*s existentes na cidade ($1 \leq M \leq 400$). As linhas seguintes serão listadas da seguinte forma ($X => 0$, $Y >= 0$, $R >= 0$), onde X e Y são as coordenadas e R o raio de cada antena. As *ERB*s serão representadas por pares de inteiros distintos (x,y) representando as coordenadas e um raio (r). Considere que não há o caso em que toda área de atuação de uma antena seja englobada por uma outra (i.e. queremos evitar *ERB*s redundantes). Lembre-se que circunferência, gerada pela atuação da antena, também é considerada dentro da área de atuação (i.e. Circunferências que se tocam são passíveis de interferência).

Para cada instância, deve ser impresso no arquivo de saída, o número de componentes conectados do grafo gerado e o número canais utilizados para ativar todas as *ERB*s de forma que não haja possibilidade de interferência.

Exemplo

A seguir temos um exemplo de funcionamento do programa:

Entrada:

```
1
8
16 39 4
11 26 5
22 30 4
19 23 4
17 11 4
24 15 3
42 42 5
43 32 6
```

Saída:

```
2
2
```

A figura 2 ilustra o exemplo, mostrando o grafo gerado.

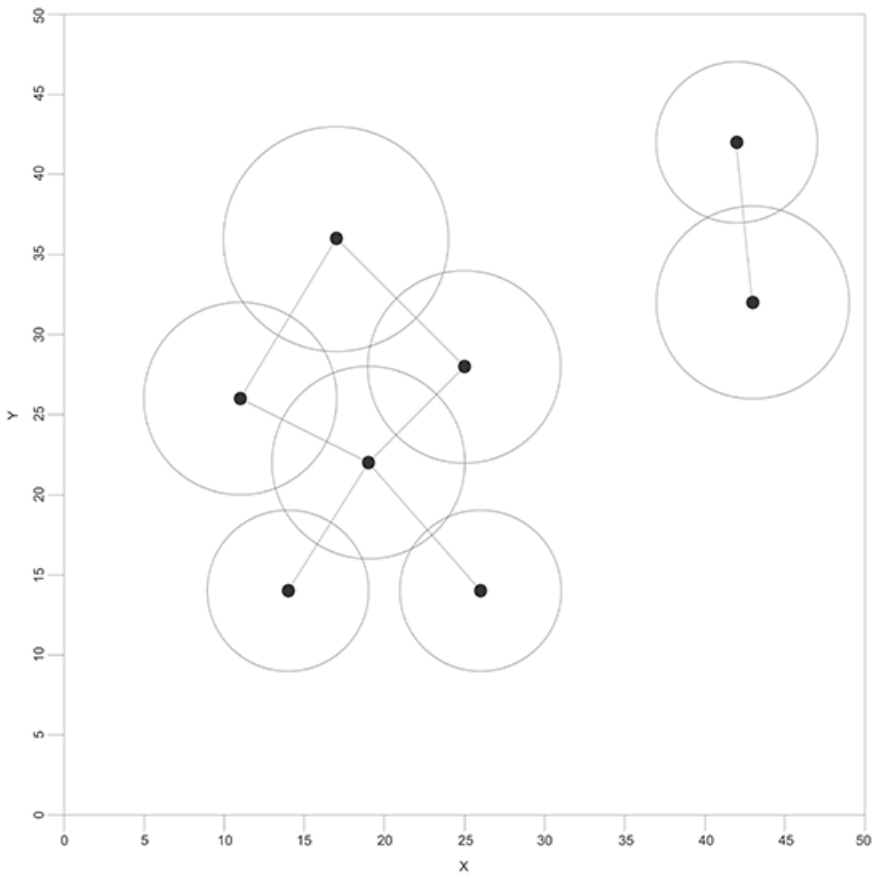

Figura 2: Grafo gerado pelo arquivo de entrada dado como exemplo

Entrega

- A data de entrega desse trabalho é **25 de Maio**.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$, onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <numero_matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.
- Não inclua arquivos compilados ou gerados por IDEs. **Apenas** os arquivos abaixo devem estar presentes no arquivo zip.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf
- Não inclua **nenhuma pasta**. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saida descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída
- NÃO SERÁ NECESSÁRIO ENTREGAR DOCUMENTAÇÃO IMPRESSA!
- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.

Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e solução proposta** para o problema. O algoritmo deve ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos.
- **Análise de complexidade** de tempo e espaço da solução implementada.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem significavelmente a execução.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.
- Espera-se gráficos que explorem o tamanho da entrada do problema (número de ERBs) versus tempo de execução e comparem a qualidade do algoritmo ótimo versus o paralelizado.

Código

- O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.
- O utilitário ***make*** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido.
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)
- **Variáveis globais** devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanto **siga rigorosamente os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- **Legibilidade e boas práticas** de programação serão avaliadas.