MC658: Análise de Algoritmos III

Turma A – Prof. Cid C. de Souza

Instituto de Computação – UNICAMP – 1º semestre de 2019

3º Trabalho de Prático[1]

1 Introdução

1. *Data/hora de entrega*: 30/05/2019 (até as 23:59.59, de acordo com horário do servidor de *email* do IC).

2. *Número de integrantes por grupo*: 2 (dois). Excepcionalmente poderá ser aceito um **único** grupo com 3 (três) alunos.

3. *Descrição do trabalho*:

   O trabalho consiste em implementar um modelo de programação por restrições (PR) para o *Problema do Escalonamento de Tarefas com Restições de Mão de Obra* (SPLC, do inglês, *Scheduling Problem under Labor Constraints*) usando o `minizinc`.

   O problema de escalonamento com restrição de mão-de-obra diz respeito ao seqüenciamento de um conjunto de jobs dependentes. Cada job tem uma duração específica, durante a qual exige uma quantidade de mão-de-obra para ser executado. A necessidade de mão-de-obra varia à medida que o job é processado. Dado o total de mão-de-obra disponível em cada período, o problema consiste em concluir todos os jobs o mais cedo possível, respeitando as restrições de precedência e a disponibilidade de mão-de-obra. Formalizando esta descrição, pode-se enunciar o SPLC da seguinte forma.

   Seja I um conjunto de ordens ($|I| = m$), onde cada ordem é composto por n_i jobs idênticos. O conjunto de todos os jobs é J ($|J| = \sum_{i=1}^{m} n_i$). Cada job é composto por um conjunto de p_j tarefas de duração um, ou seja, o job dura p_j unidades de tempo. As tarefas de um mesmo job devem ser executadas imediatamente uma após a outra, ou seja, o job deve ser executado sem *preempção* (uma vez iniciados, não podem ser interrompidos). A necessidade de mão-de-obra de um job j é especificada por um vetor $(\ell_{j1}, \ell_{j2}, \ldots, \ell_{jp_j})$, onde ℓ_{js} denota o número de trabalhadores necessários para execução da s-ésima tarefa do job j. Trabalhadores são necessários durante toda a execução de um job e existe um limite L no total de trabalhadores disponíveis em cada instante do horizonte de planejamento. As relações de precedência entre os jobs são representadas através de um grafo direcionado acíclico $G = (V, A)$ onde nós e arcos representam jobs e relações de precedência entre eles, respectivamente. Por definição, o subgrafo induzido em G por todos os jobs de uma mesma ordem é um caminho orientado.

   Uma solução para o SPLC é um escalonamento S descrito pelos instantes de início de processamento de todos os jobs $j \in J$. Um escalonamento S é viável se: (i) $\forall (i, j) \in A$, o job j não começar antes do término do job i; (ii) o total de trabalhadores solicitados por todos os jobs em processamento em t não exceder L, para todo instante t do horizonte de planejamento.

   O objetivo do SPLC é encontrar um escalonamento viável de mínimo *makespan*, isto é, um escalonamento que satisfaz às restrições de mão-de–obra e precedência entre jobs e onde o último job a ser concluído termina o mais cedo possível. Um exemplo de instância para o SPLC é mostrado na Figura 1.

4. *Instâncias para teste, código e formato de saída*:

   As instâncias de teste estarão disponíveis para *download* na página da disciplina em breve. O formato do arquivo de entrada com os dados da instância da Figura 1 pode ser visto

[1] Preparado pelo docente em colaboração com o PED da disciplina, Natanael Ramos.

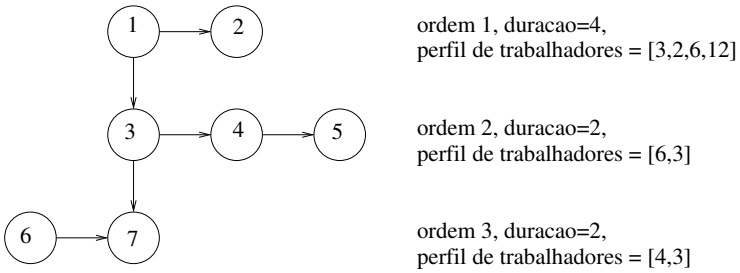

Figura 1: *Instância exemplo do SPLC.*

abaixo. Note que os arquivos terão extensão `dzn` que, como visto em aula, são processados pelos códigos do `minizinc`. Todos dados de entrada são representados por números inteiros positivos.

```
L=14;                    % número de trabalhadores disponíveis
no=3;                    % número de ordens
njo=[2,3,2];             % número de jobs por ordem
                         % os jobs de 1 a njo[1] estão na ordem 1. Para i > 2,
                         % os jobs de njo[i-1]+1 a njo[i] estão na ordem.
djo=[4,2,2];             % duração dos jobs nas ordens
trab=[3,2,6,12,6,3,4,3]; % perfil de demanda de trabalhadores das ordens:
                         % as djo[1] primeiras posições descrevem o perfil de
                         % trabalhadores para jobs da ordem 1, as djo[2] posições
                         % seguintes o perfil da ordem 2 e assim por diante.
nprec=2;                 % número de precedências entre jobs de ordens diferentes
prec=[| 1, 3,            % precedência entre jobs de ordens diferentes, uma por
      | 3, 7 |];         % linha.
```

O código `minizinc` que resolve o SPLC para a instância `instancia.dzn` deve se chamar `pert.mzn` e ser executado através do comando

`minizinc -s -t <time_limit> pert.mzn instancia.dzn`.

O tempo limite de execução (`time_limit`, dado em milissegundos) que será usado na correção do trabalho será de 3 minutos por instância.

Importante: para a correta leitura dos arquivos de entrada, o seu código `minizinc` deve conter as variáveis definidas no arquivo `dzn`, ou seja, `L, njo, djo, trab, nprec` e `prec`.

Na saída, o código deve imprimir uma linha contendo um inteiro correspondente ao valor do *makespan* obtido no tempo limite de computação fornecido e, para cada *job*, uma linha contendo dois valores inteiros, separados por um ou mais espaços em branco, sendo o primeiro valor o identificador do *job* e o outro o instante de início de sua execução, contado a partir de *zero*. Para o exemplo da Figura 1 a saída (solução ótima) seria:

```
11
1   0
2   5
3   4
4   6
5   9
6   0
7   6
```

5. *Relatório*:

   O relatório a ser entregue deverá atender aos seguintes requisitos:

   (a) O arquivo do relatório deverá estar no formato `pdf` e conter **no máximo** 8 páginas em fonte `11` ou `12pt`. **Arquivos em outros formatos não serão aceitos!**

   (b) Deverá ser dada uma breve descrição do modelo implementado explicando o significado das restrições e das variáveis. Além disso, deverá ser informada a estratégia de busca adotada na resolução das instâncias e uma discussão que justifique as suas escolhas.

   Faça um resumo dos testes que o seu grupo realizou para tentar acelerar a computação do modelo. Por exemplo: (*a*) foram feitos cálculos para reduzir domínios de variáveis? Quais? (*b*) foram testadas alternativas para estratégias de busca? Quais? (*c*) foram usadas restrições globais? Quais?; etc.

   (c) Deverá ser feita uma análise dos resultados obtidos pelo modelo do `minizinc`, apresentados no texto sob a forma de uma tabela. Especificamente, esta tabela deverá conter, para cada instância testada o valor do makespan encontrado bem como os valores reportados nas estatísticas `solveTime` e `nodes`.

   (d) O texto deverá conter ainda uma descrição do equipamento utilizado (*hardware*), incluindo memória RAM disponível, tipo de CPU, frequência do *clock*, etc.

2 Forma de entrega do trabalho

A entrega deve ser feita por *email* enviado ao docente, **com cópia para o PED**, sendo que:

- o campo `subject` deverá vir preenchido **obrigatoriamente** com os seguintes dizeres:

  ```
  [MC658-2019s1] TP3 - grupoXX
  ```

  onde `XX` é o identificador do grupo (a ser divulgado oportunamente).

- A mensagem deverá conter um anexo composto de um **único** arquivo compactado (com o comando `tar`) e chamado `grupoXX.tgz`. Ao descompactar este arquivo, além do arquivo com o código do `minizinc` (extensão `mzn`), deverá também estar presente o arquivo `grupoXX-relatorio.pdf` contendo o texto do relatório.

3 Considerações finais

As notas terão um fator comparativo que levará em consideração a qualidade dos programas, das soluções obtidas por eles e dos textos entregues pelos grupos *(você já sabe, o mundo é competitivo!)*. Os códigos devem ser implementados pelos grupos **separadamente** e não serão toleradas de forma alguma cópias parciais ou totais de códigos entre os grupos ou de material disponível na rede. Ou seja, a implementação de **todas** as linhas de código deve ser feita **exclusivamente** pelos integrantes do grupo. Qualquer desvio em relação a essa norma resultará em média semestral ZERO para todos os envolvidos, sem prejuízo de outras sanções previstas pelas regras da universidade.

4 Critérios de Correção

A distribuição de pontos do trabalho será feita do seguinte modo:

- Implementação (código): até 6 pontos, dependendo da qualidade do código e dos resultados;
- Relatório: até 4 pontos, dependendo da qualidade do documento;
- Comparativo dos grupos: *bônus* de até 1 ponto (ver detalhes abaixo);

Sobre o comparativo dos grupos. Esse problema foi estudado na dissertação de Mestrado da aluna Cristina C. B. Cavalcante, defendida no IC sob orientação do professor Cid C. de Souza. Os melhores resultados obtidos naquele trabalho estão disponíveis em `http://www.ic.unicamp.br/~cid/SPLC/SPLC.html` (seção **Best Solutions known (September 1998)**). Para uma dada instância i, seja z_i^* o valor de função objetivo para i na página de resultados, z_i^k a função objetivo obtida pelo k-ésimo grupo e $f = 1/N$ (1 ponto dividido por N instâncias). Então, a nota g_i^k do grupo k para i é dada por:

$$g_i^k = \begin{cases} f, & \text{se } z_i^k < z_i^* \\ (\frac{z_i^*}{z_i^k} \cdot f), & \text{caso contrário.} \end{cases}$$

Com isso, a nota do k-ésimo grupo no aspecto *comparativo* será dada por:

$$G^k = \sum_{i=1}^{N} g_i^k.$$