Victor Ferreira Ferrari - 187890
Vinícius Couto Espindola - 188115
TP2 - MC658 - Grupo 3

1. Modelagens dos Problemas

1.1. Problema 1 - [gt54]

Convenções:

- A instância do problema é composta por um grafo ponderado $G = (V, E)$, um conjunto de pares de vértices C e um par de vértices que representam o vértice de origem (s) e o de término (t).

Variáveis:

- $x_i \in \{0, 1\} \quad \forall i \in V$
  Indica se o vértice i faz parte (1) ou não (0) da solução.
- $e_{ij} \in \{0, 1\} \quad \forall (i,j) \in E$
  Indica se a aresta (i,j) faz parte (1) ou não (0) da solução.

Constantes:

- $P_{ij} \in Z^+ \ \forall (i,j) \in E$
  Indica o peso da aresta (i,j).

Função Objetivo:

- $min \sum_{\forall (i,j) \in E} P_{ij} \cdot e_{ij}$
  Minimizar o custo das arestas inclusas no caminho escolhido.

Restrições:

- $x_s = 1 \ \& \ x_t = 1$
  Força a inclusão dos vértices de origem s e de término t.
- $x_i + x_j \leq 1 \quad \forall (i,j) \in C$
  Se o par (i, j) está em C: apenas i, apenas j ou nenhum dos vértices podem estar inclusos. Caso contrário, nada pode-se afirmar.
- $x_i + x_j - 2 \cdot e_{ij} \geq 0 \quad \forall (i,j) \in E$
  Se uma aresta (i,j) faz parte da solução, ambos os vértices i e j devem fazer parte da solução também. Caso contrário, nada pode-se afirmar. Ainda assim, permite que i ou j estejam na solução sem a aresta (ou ainda ambos, se a aresta (j, i) existir e estiver na solução.

- $x_i - \sum_{\forall j|(i,j) \in E} e_{ij} = 0 \quad \forall i \in V \setminus \{t\}$
  Os graus de saída dos vértices da solução, com exceção de t que é o vértice de término, devem ser exatamente um.
- $x_j - \sum_{\forall i|(i,j) \in E} e_{ij} = 0 \quad \forall i \in V \setminus \{s\}$
  Os graus de entrada dos vértices da solução, com exceção de s que é o vértice de início, devem ser exatamente um.

1.2. Problema 2 - [mn27]

Convenções:

- A instância do problema é composta por um grafo $G = (V, E)$.
- Observação: No pior caso, teremos uma cor por vértice, logo o número máximo de cores a serem utilizadas é o total de vértices $(|V|)$.

Variáveis:

- $y_k \in \{0, 1\} \quad \forall k \in V$
  Indica se a cor de índice k é usada.
- $X_{ik} \in \{0, 1\} \;\; \forall i, k \in V \times V$
  Indica se o vértice i é colorido com a cor k.

Constantes:

- $n = |V|$
  Constante para representar o total de vertices/cores em G.

Função Objetivo:

- $min \sum_{\forall k \in V} y_k$
  Minimizar quantidade de cores utilizadas.

Restrições:

- $\sum_{\forall k \in V} X_{ik} = 1 \;\; \forall i \in V$
  Cada vértice deve ser colorido com uma e apenas uma cor.
- $X_{ik} + X_{jk} \leq 1 \;\; \forall k \in V \;\; \forall (i,j) \in E$
  Duas extremidades de uma aresta não podem ter a mesma cor.

- $n \cdot y_k \geq \sum_{\forall i \in V} X_{ik} \quad \forall\ k \in V$

  Se algum vértice for colorido com uma cor k, $y_k = 1$.

1.3. Problema 3 - [ss2]

Convenções:

- A instância do problema é composta por um conjunto de tarefas T cada uma com duração e prazo, e um conjunto S de pares de tarefas contidas em T.

Variáveis:

- $y_{ij} \in \{0, 1\} \quad \forall ij \in T \times T \mid i \neq j$

  Indica se a tarefa i antecede j (1) ou não (0).
- $\sigma_i \in Z^+ \quad \forall i \in T$

  Indica o tempo de início da tarefa i. A variável pertence aos inteiros não negativos, evitando tempos de início negativos.
- $X_i \in \{0, 1\} \quad \forall i \in T$

  Indica se a tarefa i está atrasada.

Constantes:

- $t_i \quad \forall i \in T$

  Constantes que representam a duração t da tarefa i.
- $d_i \quad \forall i \in T$

  Constantes que representam o *deadline* d da tarefa i.

- $M = \sum_{\forall i \in T} t_i$

  Para que M seja tão grande quanto o maior instante de término possível (considerando que desejamos minimizar os atrasos), esta é a soma das durações de todas as tarefas em T, uma vez que não é possível obter-se uma solução melhor deixando intervalos de tempo ociosos entre tarefas.

Função Objetivo:

- $min \sum_{\forall i \in T} X_i$

  A função objetivo consiste em minimizar o total de tarefas atrasadas.

Restrições:

- $y_{ij} + y_{ji} = 1 \quad \forall (i,j) \in T \times T \mid i \neq j$

  Neste caso, i antecede j, ou j antecede i. A restrição força uma relação de ordem entre as tarefas, isso também implica que toda tarefa é alocada, ou seja, possui uma posição específica na ordenação imposta.

- $y_{ij} = 1 \quad \forall (i,j) \in S$

  Força y_{ij} a respeitar as relações de antecedência definidas por S.
- $\sigma_i + t_i \leq \sigma_j + (1 - y_{ij}) \cdot M \quad \forall (i,j) \in T \times T \mid i \neq j$

  Garante que o instante de início de toda tarefa é maior ou igual ao instante de término de todas as tarefas anteriores. No caso em que i não antecede j, a restrição é irrelevante, já que $(1-0) \cdot M$ é suficientemente grande para satisfazer a restrição para todo par (i,j).
- $\sigma_i + t_i \leq d_i + MX_i \quad \forall\, i \in T$

  Garante que, se tarefas terminarem depois do prazo $(\sigma_i + t_i \geq d_i)$, serão marcadas como atrasadas $(X_i = 1)$. Como a função objetivo visa minimizar a somatória das variáveis X_i, se viável $(\sigma_i + t_i \leq d_i)$, X_i será zero.

1.4. Problema 4 - [ss5]

<u>Convenções:</u>

- A instância do problema é composta por um conjunto de tarefas T cada uma com duração, prazo e custo por unidade de tempo que deve ser pago no caso de atrasos.

<u>Variáveis:</u>

- $y_{ij} \in \{0, 1\} \quad \forall ij \in J \times J \mid i \neq j$

  Indica se a tarefa i antecede j (1) ou não (0).
- $\sigma_i \in Z^+ \quad \forall i \in J$

  Indica o tempo de início da tarefa i. A variável pertence aos inteiros não negativos, evitando tempos de início negativos.
- $m_i \in Z^+ \quad \forall i \in J$

  Indica o total de atraso, em unidades de tempo, de uma tarefa i $(m_i > 0)$ ou que não houve atraso ($m_i = 0$). A variável pertence aos inteiros não negativos, desconsiderando que tarefas adiantadas seja consideradas como “lucro” pela função objetivo.

<u>Constantes:</u>

- $p_i \quad \forall i \in J$

  Indica o custo por unidade de tempo de atraso para a tarefa i.
- $M = \sum_{\forall i \in J} t_i$

  Para que M seja tão grande quanto o maior instante de término possível (considerando que desejamos minimizar os atrasos), este é a soma das durações de todas as tarefas em T, uma vez que não é possível obter-se uma solução melhor deixando intervalos de tempo ociosos entre tarefas.

Função Objetivo:

- $min \sum_{\forall i \in J} m_i \cdot p_i$

  A função objetivo consiste em minimizar o custo total dos atrasos, ou seja, a soma dos custos por unidade de tempo de cada tarefa multiplicado pelo atraso de cada tarefa.

Restrições:

- $y_{ij} + y_{ji} = 1 \quad \forall (i,j) \in J \times J \mid i \neq j$

  Neste caso, i antecede j, ou j antecede i. A restrição força uma relação de ordem entre as tarefas, isso também implica que toda tarefa é alocada, ou seja, possui uma posição específica na ordenação imposta.
- $\sigma_i + t_i \leq \sigma_j + (1 - y_{ij}) \cdot M \quad \forall (i,j) \in J \times J \mid i \neq j$

  Garante que o instante de início de toda tarefa é maior ou igual ao instante de término de todas as tarefas anteriores. No caso em que i não antecede j, a restrição é irrelevante, já que $(1-0) \cdot M$ é suficientemente grande para satisfazer a restrição para todo par (i,j).
- $(\sigma_i + t_i - d_i) - m_i \leq 0 \quad \forall i \in J$

  Se a tarefa está atrasada ($\sigma_i + t_i - d_i > 0$), para garantir a minimização da função objetivo, m_i deve assumir o menor valor que cumpre a restrição. Logo, m_i será exatamente o atraso da tarefa i $(\sigma_i + t_i - d_i = m_i)$, igualando a combinação linear à zero. Caso contrário $(\sigma_i + t_i - d_i \leq 0)$, a tarefa está adiantada, logo, não há atraso e m_i assume o menor valor possível de seu domínio $(m_i = 0)$.

1.5. Problema 5 - [mn22]

Convenções:

- A instância do problema é composta por um grafo bipartido e ponderado $G = (V, U, E)$ onde V são máquinas e U peças , além de uma constante K indicando o máximo número de máquinas por sala.
- As arestas do grafo bipartido são denotadas por $\forall (i,j) \in E$, e subentende-se que os componentes i da arestas são máquinas $(i \in V)$, e os componentes j, peças $(j \in U)$.
- Observação: Suponha que uma peça j que está alojada na sala r possa ser movida a uma sala r' que possua múltiplas máquinas que utilizem a peça j. Como não há especificações para tal caso, e como visamos resolver o problema para a pior das hipóteses (a peça deve ser movida de volta à r antes de ser

utilizada por outra máquina em r'), consideramos o custo de movimentação de todo par máquina-peça que não esteja em r.

- <u>Observação:</u> No pior caso, teremos uma máquina por sala, logo o número máximo de salas a serem utilizadas é o total de máquinas presentes ($|V|$).

<u>Variáveis:</u>

- $v_{ir} \in \{0,1\} \quad \forall i \in V \quad \forall r \in V$
  Indica se a máquina i está presente na sala r (1) ou não (0).
- $u_{jr} \in \{0,1\} \quad \forall j \in U \quad \forall r \in V$
  Indica se a peça j está presente na sala r (1) ou não (0).
- $\lambda_{ijr} \in \{0,1\} \quad \forall (i,j) \in E \quad \forall r \in V$
  Indica se a máquina i e a peça j estão juntas na sala r (1) ou não (0).
- $\theta_{ij} \in \{0,1\} \quad \forall (i,j) \in E$
  Indica se a máquina i e a peça j estão na mesma sala (0) ou não (1).

<u>Função Objetivo:</u>

- $min \sum\limits_{\forall (i,j) \in E} \theta_{ij} \cdot c(u_j, v_i)$
  A função consiste em minimizar a somatória dos custos de movimentação de peças para o pior dos casos (vide terceira convenção).

<u>Restrições:</u>

- $\sum\limits_{\forall r \in V} v_{ir} = 1 \quad \forall i \in V$
  Cada máquina i deve estar em uma única sala r.
- $\sum\limits_{\forall r \in V} u_{jr} = 1 \quad \forall j \in U$
  Cada peça j deve estar em uma única sala r.
- $\sum\limits_{\forall i \in V} v_{ir} \leq K \quad \forall r \in V$
  Cada sala r pode comportar até K máquinas.
- $\lambda_{ijr} \geq v_{ir} + u_{jr} - 1 \;\; \& \;\; \lambda_{ijr} \leq v_{ir} \;\; \& \;\; \lambda_{ijr} \leq u_{jr} \quad \forall (i,j) \in E \;\; \forall r \in V$
  Se a máquina i está na sala r ($v_{ir} = 1$) e a peça j também está na sala r $(u_{jr} = 1)$, então deve-se indicar que a máquina i e a peça j estão na mesma sala r ($\lambda_{ijr} = 1$). Caso contrário, sabemos apenas que a máquina i e a peça j não estão juntas na sala r ($\lambda_{ijr} = 0$). Este conjunto de restrições cria um operador lógico *and* entre as variáveis v_{ir} e u_{jr} e “guarda” o resultado em λ_{ijr}.

- $\theta_{ij} = 1 - \sum_{r=1}^{|v|} \lambda_{ijr} \quad \forall (i,j) \in E$

  Se nenhuma sala r possui o par máquina-peça (i,j) ($\sum_{r=1}^{|v|} \lambda_{ijr} = 0$), deve-se incluir o custo $c(u_j, v_i)$ na função objetivo ($\theta_{ij} = 1$), caso contrário ($\sum_{r=1}^{|v|} \lambda_{ijr} = 1$), o par máquina-peça se encontram na mesma sala, então o custo $c(u_j, v_i)$ não deve ser incluso ($\theta_{ij} = 0$).

2. Especificações do Ambiente e Resultados

As especificações de hardware do computador no qual foram feitas as execuções estão na Tabela 3.

Modelo da CPU	Intel(R) Core 17-3630QM (4C/8T)
Frequência do *Clock* da CPU	2.40 GHz
RAM Disponível	12 GB/1600 MHz

Tabela 1 - Especificações de Hardware.

O sistema operacional utilizado foi o Ubuntu 18.04 LTS. Foram utilizados como *software* de execução Gnumeric V1.12.35 e Julia V1.1.0 com a biblioteca JuMP V0.18.5 e o resolvedor Gurobi Optimizer V8.10 (pacote V0.5.9). O resolvedor usado no Gnumeric foi o LPSolve.

Foram executados os modelos em Julia com limite de 6000 segundos e sem limite de memória. A instância de nome “gnumeric” foi executada também em planilha. Os resultados alcançados estão na Tabela 2.

Exercício	gnumeric	1	2	3
gt54	56	31	638	496
mn27	3	7	13	23
ss2	2	6	17	11-7
ss5	79668	36159	84222	73920-0
mn22	1	2426	8217	3796-1638

Tabela 2 - Resultados obtidos pelas modelagens com limite de 6000 segundos.

3. Particularidades Notáveis

O Gnumeric fornece opções para dois resolvedores de PL/PLI como padrão: GLPK e LPSolve. A execução no resolvedor GLPK não foi possível em quatro das cinco soluções dos problemas, pois ele retorna uma mensagem de erro sem mais informações. O único problema cujo modelo foi possível de ser resolvido pelo GLPK foi o [mn27]. Por esse motivo foi utilizado o resolvedor LPSolve, que conseguiu resolver os 5 modelos.

No problema [gt54], utilizou-se ambas uma lista de arestas e uma matriz de adjacência. Tal decisão fora tomada devido ao tempo de construção do modelo: às duas últimas restrições do modelo (referentes ao grau de entrada/saída dos vértices), chegaram a levar mais de cinco minutos no modelo utilizando apenas a lista de arestas. Utilizando-se a lista de arestas, é necessário iterar por todas as arestas existentes para buscar as de saída/entrada de cada vértice. Na matriz de adjacência, itera-se somente pelas linhas para encontrar as arestas de saída, e pelas colunas para encontrar arestas de entradas. Supondo um grafo direcionado completo de n vertices, haveria $O(n^2)$ arestas. Utilizando o método da lista, seria necessário percorrer a lista inteira para os n vertices em busca das arestas relacionadas a ele, resultando em $O(n^3)$ operações. Com a matriz de adjacências, para encontrar os graus de entrada e saída dos n vértices, percorremos n linhas e n colunas de tamanhos n, resultando em $O(n^2)$ operações.

O modelo do problema [ss5] não atualizou o dual para a instância 3, isso pode ser explicado pelo tamanho da instância e pela constante M que dificulta a otimização do limitante dual. Visando otimizar o modelo do problema [ss5], foram testadas pequenas mudanças no modelo. Removendo as variáveis negadas $(y_{ij} \leftrightarrow \neg y_{ji})$ e mantendo apenas uma permutação de (i,j) permitiu que a restrição de negação $(y_{ij} + y_{ji} = 1)$ fosse substituída pela restrição $\{\sigma_j + t_j \leq \sigma_i + y_{ij} \cdot M \quad \forall (i,j) \in \ | \ i \neq j\}$ resultando em um total de restrições um pouco menor; também foi testado o uso de M como um *upper bound* para as variáveis m_i. Todavia, nenhuma das mudanças melhoraram a performance do modelo.