# Trabalho Prático de MC658 - Dupla 19


| Nome | RA |
|---|---|
| **Lucas Alves Racoci** | 156331 |
| **Luiz Fernando Rodrigues da Fonseca** | 156475 |


# Introdução

O objetivo deste trabalho prático foi desenvolver algoritmos exatos e uma metaheurística que resolvem o problema de minimizar o custo de dias de espera dos atores em um set de gravação, como descrito no enunciado. Foram desenvolvidos três algoritmos: um Branch And Bound similar ao descrito no enunciado com pequenas modificações, um modelo de Programação Linear Inteira (PLI) e um algoritmo genético. Os resultados obtidos são comparados levando em consideração custo da solução, tempo, número de nós explorados e melhor limitante dual encontrado (estes dois últimos para o Branch And Bound e o PLI).

# Metodologia

## Branch And Bound

O algoritmo implementado em sua maior parte segue as recomendações do enunciado, implementando os limitantes inferiores $k_1$, $k_2$, $k_3$, e $k_4$ e usa a soma deles como limitante dual. A construção de soluções também segue a ideia do enunciado de ir construindo "de fora para dentro", ou seja, primeiro definimos a cena gravada no dia $1$, depois a cena no dia $n$, depois a cena no dia $n-1$.

Para calcular um limitante primal inicial, roda-se a metaheurística de algoritmo genético. Ela é rodada por $n^3$ iterações, sendo $n$ o número de cenas na instância do problema.

No caso da estratégia de exploração, a árvore é percorrida de acordo com a estratégia best bound, isto é, a solução parcial explorada é sempre a com menor limitante entre os nós ativos. A única modificação feita é que no empate, prefere-se o nó com mais cenas escolhidas (mais profundo na árvore).

Seja $P[\ell]$ a cena alocada para o dia $\ell$. Uma otimização feita para evitar o recálculo de soluções simétricas que tem o mesmo custo da forma $P'[\ell]==P[n-\ell+1]$, para $\ell \in \{1,\dots,n\}$ foi forçar $P[1]<P[n]$ quando a profundidade da árvore de soluções é 2, isto é, quando $P[1]$ e $P[n]$ já foram escolhidas. Isto faz com, que dado $P[1]$, $P[n]$ não pode ser menor ou igual ao primeiro, quebrando a simetria.

Para calcular os limitantes $k_3$ e $k_4$ foi implementado um algoritmo para encontrar um empacotamento de cenas a partir de um subconjunto de atores X, isto é, um subconjunto de cenas ainda não decididas com participação unitária de cada ator de X.

Para isso, implementou-se o seguinte algoritmo:

```
1) Q ← Cenas ainda não decididas
2) Para cada ator i do subconjunto de atores X:
      a) participação ← 0
      b) Para cada j ∈ Q em ordem decrescente de participação de atores:
          i)   participação ← participação + T[i][j];
         ii)   Se participação > 1:
                  (1)   Retire j de Q
3) Retorne Q
```

Denote por $Q_i$ o subconjunto Q quando i está sendo analisado.

Note que $Q_m \subseteq Q_{m-1} \subseteq \ldots \subseteq Q_2 \subseteq Q_1$, pois a única modificação que pode ser feita a Q no algoritmo é retirar elementos.

Suponha por contradição que o subconjunto $Q_n$ retornado não seja um empacotamento de X.

Assim existe pelo menos um ator $i \in X$ que participa de mais de 1 cena de $Q_n$. Seja $j_1$ a primeira cena e $j_2$ a segunda cena em $Q_n$ de que i participa.

Quando `j == j1` na linha `i)` a variável participação passa de `0` para 1. Quando `j == j2` na mesma linha `a` variável participação passa de 1 para 2. Assim a condicional da linha `ii)` será atendida e $j_2$ será retirada de $Q_i$. Como $Q_n \subseteq Q_{n-1} \subseteq \ldots \subseteq Q_{i+1} \subseteq Q_i$. Então $j_2$ também não estará em $Q_n$. Contradição. Assim não pode haver ator com mais de uma cena em $Q_n$. Então o Q retornado ao final é um empacotamento.

# PLI

## Ideia Inicial

Dado um escalonamento de cenas, cada ator *a*, em cada dia *d* pode estar em uma e apenas uma de quatro situações:

1. Início: O ator ainda não chegou no set no dia *d*, pois ainda não gravou nenhuma cena.
2. Gravação: O ator já chegou no set e está escalado para alguma cena no dia *d*
3. Espera: O ator já chegou no set, ainda não foi embora porque ainda tem cenas a gravar, mas não está escalado para nenhuma cena no dia *d*.
4. Final: O ator não tem mais cenas para gravar e foi embora.

Para representar o escalonamento de cenas, foi usada uma matriz de permutação, `Permutacao[c,d]`, que indica se a cena c será gravada no dia d.

Assim para cada ator `a` e cada dia `d`, definem-se as matrizes `Inicio[a,d]`, `Gravacao[a,d]`, `Espera[a,d]`, `Final[a,d]`.

Assim, deseja-se minimizar:

$$\sum_{a=1}^{m} salário[a] \cdot \sum_{d=1}^{n} Espera[a,d]$$

Sujeito às seguintes restrições:

A. Cada cena é gravada em apenas um dia:

$$\sum_{d=1}^{n} Permutacao[c,d] == 1, \ \forall c \in \{1,\ldots,n\}$$

B. Em cada dia pode ser gravada apenas uma cena:

$$\sum_{c=1}^{n} Permutacao[c,d] == 1, \forall d \in \{1,...,n\}$$

C. Cálculo dos dias de gravação a partir da entrada:

$$Gravacao[a,d] == \sum_{c=1}^{n} T[a,c] \cdot Permutacao[c,d], \forall a \in \{1,...,m\}, \forall d \in \{1,...,n\}$$

D. Cada ator em cada dia pode estar em uma e apenas uma das quatro situações:

$$Inicio[a,d] + Gravacao[a,d] + Espera[a,d] + Final[a,d] == 1, \forall a \in \{1,...,m\}, \forall d \in \{1,...,n\}$$

E. Um dia de início é precedido apenas por dias de início:

$$Inicio[a,d] <= Inicio[a,d-1], \forall a \in \{1,...,m\} \forall d \in \{2,...,n\}$$

F. Um dia de final é sucedido apenas por dias de final:

$$Final[a,d] <= Final[a,d+1], \forall a \in \{1,...,m\} \forall d \in \{1,...,n-1\}$$

A princípio todas essas variáveis foram colocadas com domínio {0,1}. Mas isso é claramente desnecessário como veremos a seguir nas otimizações:

## Otimizações

1. Pela restrição C, podemos substituir toda ocorrência de Gravacao[a,d] pela multiplicação matricial de T por Permutacao.
2. Se 3 das 4 matrizes de situação forem binárias, pela restrição D a outra também o será. Como Gravacao será substituída por uma expressão que é binária, pois apenas troca a ordem das colunas da entrada, podemos relaxar ainda uma outra matriz. Escolhemos relaxar a Espera.
    a. Assim surge a restrição: $0 \leq Espera[a,d] \leq 1, \forall a \in \{1,...,m\} \forall d \in \{1,...,n\}$
3. Pela restrição D, podemos substituir toda ocorrência de Espera[a,d] por

$$1 - (Inicio[a,d] + \sum_{c=1}^{n} T[a,c] \cdot Permutacao[c,d] + Final[a,d])$$

    a. Assim, a nova restrição (2a) fica:

$$0 <= 1 - (Inicio[a,d] + \sum_{c=1}^{n} T[a,c] \cdot Permutacao[c,d] + Final[a,d]) <= 1$$

        i. Lembrando que $0 \leq 1-x \leq 1 \Leftrightarrow 0 \leq x \leq 1$, podemos simplificá-la para:

$$0 <= Inicio[a,d] + \sum_{c=1}^{n} T[a,d] \cdot Permutacao[c,d] + Final[a,d] <= 1$$

    b. A função objetivo fica:

$$\sum_{a=1}^{m} salário[a] \cdot \sum_{d=1}^{n} (1 - (Inicio[a,d] + \sum_{c=1}^{n} T[a,c] \cdot Permutacao[c,d] + Final[a,d]))$$

        i. Distribuindo o somatório e combinando alguns termos temos

$$\sum_{a=1}^{m} salário[a] \cdot (n - \sum_{d=1}^{n} \sum_{c=1}^{n} T[a,c] \cdot Permutacao[c,d] - \sum_{d=1}^{n} Inicio[a,d] + Final[a,d])$$

        ii. Como Permutacao[c,d] só troca a ordem das colunas de T[a,c] e todas as colunas estão sendo somadas:

$$\sum_{a=1}^{m} salário[a] \cdot (n - \sum_{d=1}^{n} T[a,c] - \sum_{d=1}^{n} Inicio[a,d] + Final[a,d])$$

## Modelo Final

Abaixo segue o modelo final do PLI e uma análise do número de variáveis e restrições do modelo em função de `n` e `m`, que são o número de cenas e o número de atores do problema, respectivamente:

Variáveis:

{0,1} ∋ Permutacao: n x n

{0,1} ∋ Inicio: m x n

{0,1} ∋ Final: m x n

Minimizar:

$$\sum_{a=1}^{m} salário[a] \cdot (n - \sum_{d=1}^{n} T[a,c] - \sum_{d=1}^{n} Inicio[a,d] + Final[a,d])$$

Sujeito a:

A. Cada cena gravada em um dia (2n restrições) $\sum_{d=1}^{n} Permutacao[c,d] == 1, \ \forall c \in \{1,...,n\}$

B. Cada dia com apenas uma cena gravada (2n restrições) $\sum_{c=1}^{n} Permutacao[c,d] == 1, \forall d \in \{1,...,n\}$

C. Limita a espera (2mn restrições) $0 <= Inicio[a,d] + \sum_{c=1}^{n} T[a,d] \cdot Permutacao[c,d] + Final[a,d] <= 1$

D. Início decrescente (m(n-1) restrições) $Inicio[a,d] <= Inicio[a,d-1], \ \forall a \in \{1,...,m\} \forall d \in \{2,...,n\}$

E. Final crescente ( m(n-1) restrições) $Final[a,d] <= Final[a,d+1], \ \forall a \in \{1,...,m\} \forall d \in \{1,...,n-1\}$

Total de Restrições: 4·n + 2·m(n-1) + 2mn = O(mn)

Total de Variáveis: 2mn + $n^2$ = O((m+n)n)

## Algoritmo Genético

Como metaheurística, foi escolhido e implementado um algoritmo genético. Os indivíduos do algoritmo são permutações que indicam a ordem que as cenas seriam gravadas no set, ou seja, a permutação `3 1 2` indica que a cena 3 é gravada no dia 1, a cena 1 é gravada no dia 2 e a cena 2 é gravada no dia 3. Abaixo está um pseudocódigo para o algoritmo:

```
1) Enquanto não acabar o tempo:
    a) Crie uma população inicial de indivíduos
    b) Para i de de 1 até 10000
       i)   Gere indivíduos por cruzamentos na população
      ii)   Gere indivíduos por mutações na população
     iii)   Atualize a população com o melhor indivíduo
```

Para criar uma população inicial, são criados `TAM_POPULACAO = 10` indivíduos diferentes utilizando a função `random_shuffle` do C++, eliminando indivíduos duplicados. O passo `1` foi feito para o algoritmo recomeçar várias vezes e gerar novas populações várias vezes. Foram feitos alguns testes com as instâncias abertas, e após pouco tempo (aproximadamente 2 ou 3 segundos) o algoritmo tinha dificuldades em encontrar soluções melhores do que a que ele tinha. Porém, se rodasse novamente, talvez ele caísse em uma solução melhor do que a que ele tinha encontrado na rodada passada onde demorou mais tempo. Assim, essa estratégia é uma tentativa de fugir de mínimos locais que possivelmente o algoritmo estava caindo. Não eram

de fato mínimos locais, pois foram impressos para testes os elementos da população, e os custos entre eles variaram bastante. O `for` do passo `b` foi colocado de 1 a 10000 para os menores testes rodarem uma iteração em aproximadamente 0,5 segundo e os maiores em aproximadamente 1 segundo. No algoritmo de Branch And Bound, o passo `1` não existe, somente criando e evoluindo uma única população.

Tanto para o caso de cruzamentos quando o caso de mutações, a escolha de qual operação fazer foi feita usada uma estratégia round-robin, ou seja, elas foram escolhidas na ordem que estavam no código e depois voltavam para a primeira. A cada iteração, foram gerados `TAM_CRUZAMENTO = 10` novos indivíduos por cruzamento e `TAM_MUTACAO = 20` novos indivíduos por mutação.

Para gerar indivíduos por cruzamentos, foram utilizadas duas estratégias diferentes, mas ambas sorteavam dois pais distintos para realizar a operação:

- A primeira estratégia sorteia um índice no vetor, e copia o início da permutação do primeiro pai para o filho. Então, copia os elementos faltantes do segundo pai na ordem em que eles aparecem.
- A segunda estratégia sorteia dois índices distintos no vetor, e então copia o trecho entre estes dois índices do primeiro pai para o filho. Então, percorre o segundo pai copiando os elementos faltantes do segundo pai na ordem em que eles aparecem. Esta estratégia é parecida com a primeira, exceto pelo fato de que a primeira fixa o menor índice no início do primeiro pai.

Para gerar indivíduos por mutação, foram utilizadas três estratégias diferentes, mas todas sorteavam dois índices distintos do indivíduo:

- A primeira estratégia simplesmente troca os elementos dos dois índices da permutação.
- A segunda estratégia pega o elemento do maior índice e traz para o lado do elemento do menor índice, ou seja, desloca um trecho para a direita.
- A terceira estratégia seleciona um trecho e inverte os valores dentro do trecho. A única exceção é que quando ambas as extremidades são selecionadas, então os índices são trazidos uma unidade para dentro cada, pois se invertêssemos tudo, teríamos o mesmo custo da solução após a mutação.

Por fim, para atualizar a população, foi selecionado o melhor indivíduo para a nova geração. Os outros `TAM_POPULACAO - 1` indivíduos eram selecionados aleatoriamente entre os indivíduos da população e os indivíduos gerados.

# Resultados

Abaixo, estão os resultados obtidos para os três algoritmos com as instâncias dadas.

## Branch And Bound

| BnB | | | | | |
|---|---|---|---|---|---|
| Instancia | Solução | Custo | Lim. Inf. | Nos | Tempo |
| e0806.txt | 1 5 2 4 7 6 3 8 | 4 | 4,00 | 32 | 0,00 |
| e1006.txt | 3 1 9 4 7 10 2 5 6 8 | 12 | 12,00 | 232 | 0,01 |
| e1008.txt | 7 2 9 1 4 10 3 6 5 8 | 24 | 24,00 | 82 | 0,01 |
| e1010.txt | 7 10 1 2 9 5 3 4 8 6 | 51 | 51,00 | 129 | 0,01 |
| e1206.txt | 8 5 6 1 10 11 3 2 7 9 4 12 | 22 | 22,00 | 1.830 | 0,04 |
| e1208.txt | 3 10 7 11 12 2 9 6 1 5 4 8 | 69 | 69,00 | 5.146 | 0,10 |
| e1210.txt | 7 6 4 8 11 10 5 1 2 12 9 3 | 81 | 81,00 | 5.710 | 0,12 |
| e1211.txt | 4 1 9 3 10 7 8 5 6 12 11 2 | 104 | 104,00 | 2.894 | 0,07 |
| e1410.txt | 14 11 8 2 6 3 4 12 9 1 13 10 5 7 | 115 | 115,00 | 34.093 | 0,68 |
| e1411.txt | 14 7 8 12 11 2 3 10 6 13 9 1 4 5 | 132 | 132,00 | 28.300 | 0,61 |
| e1808.txt | 16 8 17 2 11 4 7 5 3 15 10 14 9 6 13 18 1 12 | 135 | 135,00 | 1.203.369 | 30,78 |

## PLI

| PLI | | | | | |
|---|---|---|---|---|---|
| Instância | Solução | Custo | Lim. Inf. | Nós | Tempo |
| e0806.dat | 1 6 8 5 3 4 7 2 | 4 | 4.00 | 203 | 0,21 |
| e1006.dat | 3 1 9 5 7 10 2 4 6 8 | 12 | 12.00 | 595 | 0,61 |
| e1008.dat | 4 10 2 9 7 1 6 5 8 3 | 24 | 24.00 | 775 | 0,88 |
| e1010.dat | 7 10 2 3 9 4 1 5 8 6 | 51 | 51.00 | 3.281 | 5,60 |
| e1206.dat | 7 8 5 12 3 2 10 11 6 4 9 1 | 22 | 22.00 | 10.427 | 14,46 |
| e1208.dat | 3 10 7 11 12 2 9 6 1 4 5 8 | 69 | 69.00 | 23.463 | 38,85 |
| e1210.dat | 6 7 9 5 2 3 8 12 11 1 4 10 | 81 | 81.00 | 46.199 | 106,66 |
| e1211.dat | 10 12 4 9 3 6 5 8 7 1 2 11 | 104 | 104.00 | 57.969 | 154,11 |
| e1410.dat | 2 5 8 1 10 13 12 4 6 14 3 7 11 9 | 115 | 0.00 | 61.075 | 180,03 |
| e1411.dat | 11 6 7 13 9 1 2 10 5 12 8 14 3 4 | 137 | 0.00 | 64.827 | 180,03 |
| e1808.dat | 3 9 4 17 7 14 11 15 16 2 12 5 10 13 6 1 18 8 | 135 | 0.00 | 48.901 | 180,05 |

# Algoritmo Genético

| Algoritmo Genético | | | |
|---|---|---|---|
| Instancia | Solução | Custo | Tempo |
| h3016a.txt | 30 12 3 25 18 20 4 23 11 14 7 15 13 2 17 26 21 1 24 8 22 9 29 28 19 27 16 6 10 5 | 4.842 | 30,01 |
| h3016b.txt | 28 22 11 12 30 24 9 27 10 16 18 20 23 2 26 25 6 19 14 29 4 17 8 3 1 21 15 5 7 13 | 2.972 | 30,00 |
| h3016c.txt | 16 20 2 29 18 1 21 10 24 19 13 17 22 5 28 27 30 11 4 23 25 6 14 26 15 9 12 3 7 8 | 3.495 | 30,00 |
| h3018a.txt | 8 29 24 19 20 10 14 16 18 12 1 27 6 4 5 22 9 30 3 23 25 28 21 13 11 17 2 26 15 7 | 5.280 | 30,00 |
| h3018b.txt | 20 16 10 27 15 22 2 8 3 6 9 29 14 18 19 28 13 30 24 26 11 23 5 21 4 7 17 1 25 12 | 5.502 | 30,00 |
| h3018c.txt | 8 29 17 6 7 11 23 24 21 14 27 5 28 9 26 25 18 30 12 22 2 20 10 19 16 4 3 15 1 13 | 4.204 | 30,00 |
| h3020a.txt | 27 10 2 21 6 8 13 14 16 30 25 29 28 1 5 23 22 3 15 18 11 24 17 19 4 26 20 7 12 9 | 6.144 | 30,00 |
| h3020b.txt | 19 15 5 7 11 26 1 22 6 2 29 10 16 3 9 18 13 20 27 25 14 17 21 23 12 24 4 30 28 8 | 4.488 | 30,00 |
| h3020c.txt | 18 4 11 29 23 19 12 8 14 24 30 28 27 17 3 13 20 22 7 21 6 25 1 26 9 16 10 5 15 2 | 5.747 | 30,00 |
| h4016a.txt | 4 29 7 10 26 37 27 1 16 32 25 18 5 2 15 31 13 30 40 22 14 35 8 3 38 36 6 33 39 28 9 19 34 12 17 23 11 21 20 24 | 4.247 | 30,00 |
| h4016b.txt | 17 37 10 20 23 25 13 9 39 6 33 38 1 24 15 34 27 7 4 11 21 30 3 18 19 31 40 8 28 36 29 5 12 16 22 32 26 14 2 35 | 4.707 | 30,00 |
| h4016c.txt | 36 6 7 13 27 9 15 12 14 1 30 21 33 11 5 40 2 26 19 25 10 32 35 20 28 18 3 29 4 37 22 31 8 34 39 16 23 24 17 38 | 3.527 | 30,00 |
| h4018a.txt | 30 11 7 17 39 16 29 31 33 19 26 10 32 4 20 40 8 15 5 3 34 36 24 25 35 23 27 28 22 21 6 14 38 18 1 2 13 9 37 12 | 5.051 | 30,00 |
| h4018b.txt | 21 10 15 12 33 6 24 31 7 38 20 23 35 29 25 8 18 27 2 28 11 37 9 32 34 16 4 14 3 5 17 19 36 40 13 39 26 1 30 22 | 5.095 | 30,00 |
| h4018c.txt | 33 14 34 23 5 20 3 22 27 21 6 30 17 39 1 16 10 9 8 31 24 7 19 38 12 37 36 15 13 32 11 40 35 18 26 25 4 28 29 2 | 5.183 | 30,00 |
| h4020a.txt | 14 36 11 28 33 9 24 22 26 4 2 30 17 8 10 38 13 40 7 32 19 39 27 34 3 25 23 16 20 31 5 1 29 37 15 21 35 6 18 12 | 7.366 | 30,00 |
| h4020b.txt | 34 14 8 30 27 35 9 19 31 7 4 2 37 6 24 36 5 1 38 32 22 11 18 10 33 15 39 21 17 23 29 40 12 25 28 20 13 26 3 16 | 5.681 | 30,00 |
| h4020c.txt | 15 23 11 13 39 18 31 6 33 12 17 40 14 2 1 19 9 5 21 8 3 10 28 38 29 4 25 24 20 32 7 35 22 30 27 37 36 16 34 26 | 4.906 | 30,00 |
| h5016a.txt | 34 38 16 48 35 43 33 12 31 50 44 15 36 21 7 29 13 20 6 10 24 5 40 9 8 30 4 2 3 14 49 42 47 11 37 17 1 27 23 46 22 26 45 28 19 32 25 41 18 39 | 7.028 | 30,00 |
| h5016b.txt | 32 49 24 7 26 23 21 28 25 39 34 4 6 8 17 1 18 19 35 10 47 15 30 33 50 13 16 36 11 12 45 14 27 37 29 2 46 3 5 43 9 48 22 20 44 31 41 42 38 40 | 5.324 | 30,00 |
| h5016c.txt | 22 34 44 47 1 23 48 42 17 31 43 37 11 41 8 45 14 50 21 15 12 6 16 35 40 25 9 20 39 46 28 19 18 30 38 2 36 32 5 13 4 3 49 26 24 27 10 33 7 29 | 10.127 | 30,00 |
| h5018a.txt | 31 9 11 45 47 6 14 48 17 44 8 21 25 39 4 41 20 10 12 15 5 26 46 1 2 16 34 29 7 27 19 37 49 40 50 24 38 3 18 13 32 33 42 43 36 22 23 30 35 28 | 10.171 | 30,00 |
| h5018b.txt | 16 33 35 8 18 15 36 4 29 50 49 39 40 6 47 38 12 17 14 5 13 10 41 23 28 9 48 20 34 1 37 2 43 42 31 11 22 7 19 44 46 3 45 21 32 25 27 24 30 26 | 6.201 | 30,00 |
| h5018c.txt | 20 6 10 12 7 27 14 13 15 47 50 40 24 5 31 42 33 30 26 46 28 35 45 16 17 3 19 29 2 18 9 32 25 1 22 8 23 11 44 4 48 49 34 21 36 39 43 37 41 38 | 6.860 | 30,00 |
| h5020a.txt | 5 44 12 23 15 38 1 9 25 2 42 19 13 39 37 47 35 20 32 40 36 18 11 4 16 34 10 3 8 49 45 6 33 21 22 43 48 29 7 17 41 46 14 50 28 26 31 24 30 27 | 9.512 | 30,00 |
| h5020b.txt | 40 14 1 39 36 4 15 16 43 42 49 29 12 10 17 32 28 5 9 23 26 34 19 37 3 35 30 11 2 46 18 13 47 7 45 41 44 6 48 8 21 50 31 38 33 27 22 20 25 24 | 8.922 | 30,00 |
| h5020c.txt | 1 18 37 41 26 33 30 13 8 46 36 44 7 9 48 39 19 4 6 45 34 29 42 32 49 17 2 14 23 10 20 25 27 31 11 50 5 16 21 15 12 43 3 47 22 38 35 40 24 28 | 8.367 | 30,00 |

# Análise

Analisando os resultados do Branch and Bound, pode-se perceber que em todos os casos o custo da melhor solução encontrada coincidiu com o custo do melhor limitante inferior encontrado, provando a otimalidade da solução. Também foi possível perceber que os tempos de execução foram bem rápidos (menos de 1 segundo), exceto para o último teste `e1808`, que demorou aproximadamente 30 segundos. Mesmo assim neste caso, o tempo de execução foi bem menor do que o máximo de tempo que seria executado (3 minutos), o que é um resultado muito bom. Olhando a quantidade de nós explorados, ocorreu em alguns casos de mesmo número de cenas, que se existiam mais atores foram explorados menos nós da árvore. Isso pode ter acontecido pelas características dos limitantes que envolviam empacotamento, pois com mais atores possivelmente o algoritmo teria mais chances de encontrar um empacotamento maior em alguns casos, mas isso nem sempre acontece.

Analisando os resultados do PLI, pode-se perceber que até o teste `e1211`, o custo da melhor solução coincidiu com o limitante inferior, provando a otimalidade da solução. Nestes casos também o tempo foi menor do que o limite de 3 minutos. Porém, nos últimos três casos o algoritmo não terminou a sua execução, e os limitantes inferiores deram 0.00, indicando que nós ainda precisavam ser explorados. Também é importante notar que em dois destes três casos, `e1410` e `e1808`, os custos das melhores soluções encontradas pelo algoritmo coincidiu com o valor obtido no Branch And Bound, e no teste `e1411` foi um pouco maior.

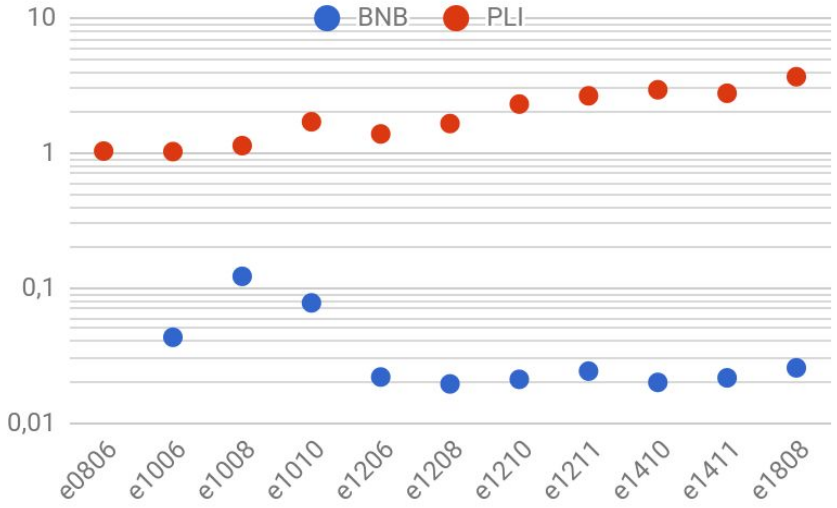


Gráfico 1: Tempo médio por nó (em milissegundos) (eixo y) por instância (eixo x) nas execuções do BNB e PLI

Assim, comparando o Branch And Bound com o PLI, podemos perceber que, nos casos que o PLI terminou, os resultados coincidiram. Além disso o tempo médio por nó executando o Branch and Bound fica sempre uma ordem de grandeza abaixo do tempo médio por nó no PLI, o que faz sentido considerando que a cada nó o PLI resolve uma Programação Linear, enquanto que o BnB apenas calcula os limitantes da forma como explicada no enunciado.

Analisando os resultados do Algoritmo Genético, não é possível tirar uma conclusão precisa sobre os resultados, pois como se trata de uma heurística e as entradas são grandes, não são conhecidas as soluções ótimas para os testes. Porém, a heurística também foi testada com os casos pequenos dos algoritmos exatos,

e os resultados convergiram para as soluções ótimas em menos de 2 segundos em todos os casos. Também é importante ressaltar a modificação feita no Algoritmo Genético, pois sem a ideia de recomeçar todo o processo novamente várias vezes, o algoritmo acabava convergindo para uma resposta rapidamente, mas se fosse executado novamente, o algoritmo convergiu para outra resposta rapidamente. Assim, adicionando essa ideia, os resultados do algoritmo genético começaram a variar menos e começar a dar custos menores com uma maior constância.

# Conclusão

Após o desenvolvimento dos três algoritmos e a execução dos testes com as instâncias dadas, foi possível perceber que o Branch And Bound teve um desempenho superior ao PLI para os casos pequenos, já que em alguns casos o PLI não conseguiu terminar a sua execução, e o Branch And Bound terminou todas as execuções em menos de 1 minuto. O Algoritmo Genético também obteve ótimos resultados para os casos pequenos, convergindo para as respostas ótimas do Branch And Bound em menos de 2 segundos em cada teste. Em casos maiores, também foi conseguido uma maior constância nos resultados utilizando a modificação proposta.