# Problema do Trabalho Balanceado

Gustavo Delazeri

12 de Agosto de 2020

## 1 Introdução

O objetivo deste trabalho é implementar, calibrar e testar um algoritmo genético para resolver o problema do Trabalho Balanceado. O problema do Trabalho Balanceado é assim definido: dados um conjunto de $n$ tarefas que precisam ser executadas em sequência, um conjunto de $m$ operadores e uma matriz $p$, onde $p_{ij}$ representa o tempo necessário para o trabalhador $j$ executar a tarefa $i$, encontre uma partição das $n$ tarefas em $m$ intervalos $[b_k, e_k]\, k \in [m]$ e uma permutação dos operadores $\pi$ de forma que $T = max_{j \in [m]} T_j$ é minimizado, sendo que $T_j = \sum_{t \in [b_t, e_t]} p_{t\pi_j}$. Em outras palavras, encontre uma atribuição de tarefas a operadores de tal modo que operadores executem tarefas sequenciais e o tempo gasto pelo operador que gasta mais tempo trabalhando é minimizado.

## 2 Formulação do Problema como Programa Inteiro

**Variáveis:**

- $x_{ij} \in \{0, 1\}, \quad \forall i, j \mid i \in [n] \wedge j \in [m]$, onde

$$x_{ij} = \begin{cases} 1, & \text{Caso tarefa i e executada pelo operador j} \\ 0, & \text{Caso contrario} \end{cases}$$

- $w_{ijk} \in \{0, 1\}, \quad \forall i, j, k \mid i, j \in [n] \wedge k \in [m] \wedge i \neq j$, onde

$$w_{ijk} = \begin{cases} 1, & \text{Caso } x_{ik} \wedge x_{jk} \text{ é verdade} \\ 0, & \text{Caso contrario} \end{cases}$$

- $y \in \mathbb{R}$, onde

$$y = \max\left( \sum_{i \in [n]} p_{ij} \cdot x_{ij}, \forall j \in [m] \right)$$

**Função Objetivo:**

$$\text{Min.} \quad y$$

**Restrições:**

$$\sum_{j \in [m]} x_{ij} = 1, \quad \forall i \in [n] \tag{1}$$

$$\sum_{i \in [n]} x_{ij} \geq 1, \quad \forall j \in [m] \tag{2}$$

$$w_{ijk} \leq (x_{ik} + x_{jk})/2, \quad \forall i, j, k \mid i, j \in [n] \wedge k \in [m] \wedge i < j \tag{3}$$

$$w_{ijk} \geq x_{ik} + x_{jk} - 1, \quad \forall i, j, k \mid i, j \in [n] \wedge k \in [m] \wedge i < j \tag{4}$$

$$w_{ijk} \leq x_{j-1k}, \quad \forall i, j, k \mid i, j \in [n] \wedge k \in [m] \wedge i < j \tag{5}$$

$$y \geq \sum_{i \in [n]} p_{ij} \cdot x_{ij}, \quad \forall j \in [m] \tag{6}$$

A restrição (1) garante que toda tarefa é executada por exatamente 1 operador. A restrição (2) garante que todo operador executa pelo menos uma tarefa. Restrições (3) e (4) formam uma conjunção: se as tarefas i e j são executadas pelo mesmo operador $k$, então $w_{ijk}$ é verdade. A restrição (5) garante que operadores só executam tarefas sequenciais. Por exemplo, um operador não pode executar as tarefas 1,2 e 4. A restrição (6) define um limite inferior para a variável $y$, a qual representa o tempo gasto pelo operador que trabalha por mais tempo.

# 3 O Algoritmo Genético

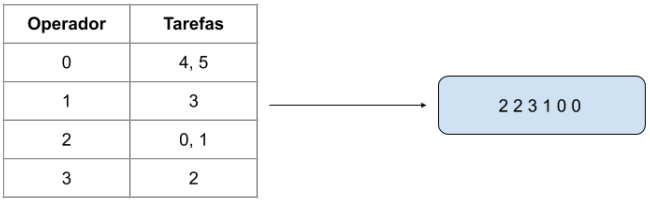
## 3.1 Parâmetros

A tabela abaixo apresenta os parâmetros do algoritmo e a notação adotada.

| | |
|---|---|
| $\mu$ | Quantidade de indivíduos na população inicial |
| $\lambda$ | Quantidade de novos individuos gerados |
| $k$ | Número de participantes de um torneio aleatório |
| $\phi$ | Probabilidade de um indivíduo sofrer mutação |
| $\omega$ | Número máximo de gerações consecutivas que não alteram a melhor solução |

## 3.2 Codificação de uma Solução

Uma solução para uma instância de $n$ tarefas e $m$ operadores é representada por um vetor $v$ de inteiros não negativos de tamanho $n$. Se $v_i = j$ então o operador $j$ executa a tarefa $i$. A figura abaixo ilustra a codificação de uma solução de uma instância com 6 tarefas e 4 operadores.



## 3.3 População Inicial

A população inicial é gerada aleatoriamente. Primeiro criam-se $\mu$ permutações e depois $\mu$ partições. Por último, associa-se a cada permutação uma partição, também de forma aleatória. A tabela abaixo ilustra o processo considerando uma instância de 8 tarefas e 4 operadores.

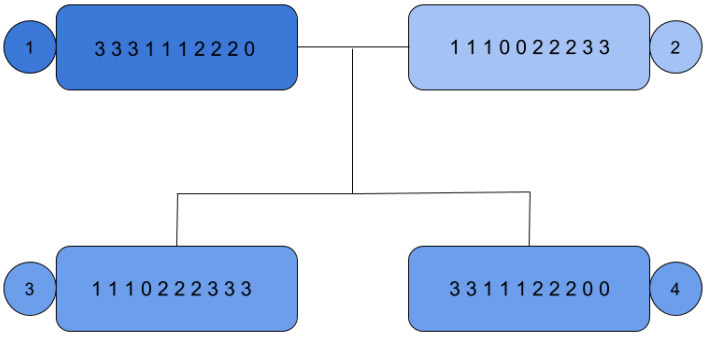
| Permutação | Partição | Indivíduo Gerado |
|---|---|---|
| (0 3 2 1) | 0 0 1 1 2 2 3 3 | 0 0 3 3 2 2 1 1 |
| (2 3 0 1) | 0 1 1 1 2 3 3 3 | 2 3 3 3 0 1 1 1 |
| (3 1 0 2) | 0 0 0 0 0 1 2 3 | 3 1 0 0 0 0 0 2 |
| (1 2 3 0) | 0 1 2 2 2 3 3 3 | 1 2 2 2 3 3 3 0 |

## 3.4 Seleção de Indivíduos para Crossover

A seleção de indivíduos para crossover implica na realização de $\lambda$ $k$-torneios aleatórios. Um $k$-torneio aleatório consiste em selecionar $k$ indivíduos da população de forma aleatória e escolher o melhor desses $k$ indivíduos.

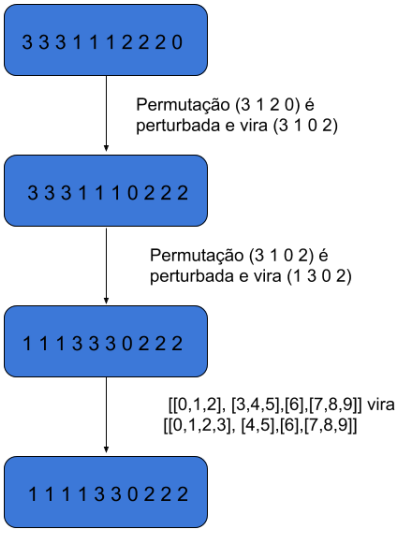
## 3.5 Crossover

O processo de crossover consiste em criar duas novas soluções usando a partição/permutação de um pai com a partição/permutação do outro. A figura abaixo ilustra o processo. O indivíduo 3 herda a permutação do indivíduo 2 (1 0 2 3) e a partição do indivíduo 1 (um 0, três 1's, três 2's e três 3's). Já o indivíduo 4 herda a permutação do indivíduo 1 (3 1 2 0) e a partição do indivíduo 2 (dois 0's, três 1's , três 2's e dois 3's).




## 3.6 Mutação

O processo de mutação tem duas etapas. A primeira consiste em gerar um número aleatório $k \in \{1, 2, 3, ..., 10\}$ e aplicar $k$ pequenas perturbações na permutação da solução. A segunda etapa consiste em escolher dois operadores $w_1$ e $w_2$ aleatoriamente, sendo que $w_1$ terá seu intervalo $l$ unidades maior e $w_2$ terá seu intervalo $l$ unidades menor. O valor de $l$ é um número aleatório maior ou igual a zero e menor ou igual à metade da magnitude do intervalo de $w_2$. A figura abaixo ilustra o processo para $k = 2$, $w_1 = 1$, $w_2 = 3$ e $l = 1$.




## 3.7 Seleção da Nova População

A seleção da nova população depende do parâmetro $\lambda$ . Se $\lambda$ novos indivíduos foram criados via crossover, então os $\lambda$ piores indivíduos entre todos os indivíduos (geração atual e nova geração) são eliminados da população.

## 3.8 Critério de Parada

A execução do algoritmo para e retorna uma solução se $\omega$ gerações consecutivas foram geradas e o valor da função objetivo não diminuiu.

## 3.9 Pseudocódigo

**Algorithm 1** Algoritmo Genético

$populacao \leftarrow populacaoInicial(\mu)$
$M \leftarrow melhorIndividuo(populacao)$
$geracoesSemMelhora \leftarrow 0$
**while** $geracoesSemMelhora < \omega$ **do**
  **for** $i \in [\frac{\lambda}{2}]$ **do**
    $pai \leftarrow torneioAleatorio(populacao, k)$
    $mae \leftarrow torneioAleatorio(populacao, k)$
    $filho1, filho2 \leftarrow crossover(pai, mae)$
    **if** $random[0, 1) < \phi$ **then**
      $filho1 \leftarrow mutacao(filho1)$
    **end if**
    **if** $random[0, 1) < \phi$ **then**
      $filho2 \leftarrow mutacao(filho2)$
    **end if**
    $populacao \leftarrow populacao \cup \{filho1, filho2\}$
  **end for**
  $populacao \leftarrow removePioresIndividuos(populacao, \lambda)$
  $N \leftarrow melhorIndividuo(populacao)$
  **if** $N$ é pior ou igual a $M$ **then**
    $geracoesSemMelhora \leftarrow geracoesSemMelhora + 1$
  **else**
    $M \leftarrow N$
    $geracoesSemMelhora \leftarrow 0$
  **end if**
**end while**
**return** $M$

## 3.10 Implementação

### 3.10.1 Plataforma e Linguagem de Programação

O algoritmo genético foi implementado em Python e usa somente bibliotecas padrão da linguagem. A plataforma utilizada possui sistema operacional macOS Catalina, versão 10.15.6, com um processador Intel(R) Core(TM) i5, com 2 núcleos físicos de 2.3GHz, com cache L2 de 256KB (em cada núcleo) e 8GB de memória.

### 3.10.2 Estruturas de Dados

A representação de um indivíduo (cromossomo) é feita utilizando uma lista de tamanho $n$. Visando facilitar as operações de mutação e crossover, a permutação de operadores e o tamanho do intervalo de cada operador são armazenados durante a execução usando listas de tamannho $m$ e $n$, respectivamente.

### 3.10.3 Cálculo do fitness de um indivíduo

Para calcular o fitness de um indivíduo é necessário percorrer a lista que representa o cromossomo e armazenar o tempo gasto por cada operador em uma lista de tamanho $m$. Retorna-se, então, o maior valor da lista de tamanho $m$.

# 4 Resultados Numéricos

Os testes realizados dividem-se em duas categorias: teste de parâmetros e teste das instâncias. Os testes de parâmetros servem para calibrar o algoritmo genético. Os testes das instâncias servem para comparar o algoritmo genético com a resolução via programação inteira mista usando um solver. Todas as instâncias usadas nos testes podem ser encontradas em `http://www.inf.ufrgs.br/~mrpritt/oc/trsp.zip`

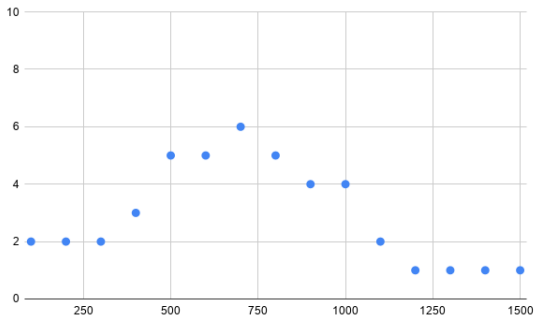
## 4.1 Teste de Parâmetros

Para os testes de parâmetros definiu-se uma configuração base arbitrária, a partir da qual cada parâmetro apresentado na tabela da seção 3.1 é variado individualmente. A configuração base é mostrada abaixo. Para cada variação de parâmetro, o valor da solução associada é a média aritmética das soluções encontradas para as instâncias $tba1$, $tba1$ e $tba3$. O valor da solução de cada instância é a média aritmética de 5 execuções.

| $\mu$ | $\lambda$ | $k$ | $\phi$ | $\omega$ |
|---|---|---|---|---|
| 500 | 100 | 3 | 0.5 | 500 |

### 4.1.1 $\mu$ - Tamanho da população inicial

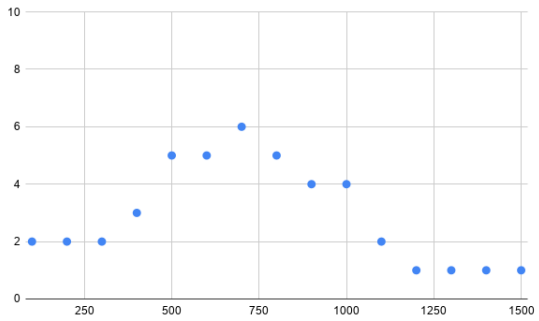
O valor de $\mu$ foi variado de 100 a 1500 com incrementos de 100. A figura abaixo mostra os resultados.




tiam sit amet lectus nec magna aliquet convallis. Curabitur sed lobortis nulla, sed imperdiet libero. Morbi rhoncus eros nec nisi auctor egestas. Quisque ut nisl dictum lectus mollis tristique rutrum eget arcu. Sed sit amet blandit sapien, eget pulvinar enim. Praesent scelerisque quam a leo pellentesque, a congue arcu luctus. Integer augue urna, vestibulum et nulla vel, lacinia volutpat neque.

#### 4.1.2 $\lambda$ - Tamanho da prole

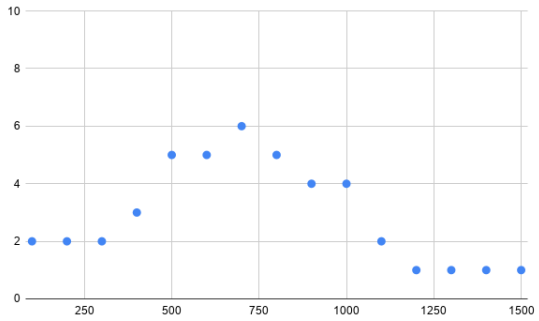
O valor de $\lambda$ foi variado de 10 a 200 com incrementos de 10. A figura abaixo mostra os resultados.




tiam sit amet lectus nec magna aliquet convallis. Curabitur sed lobortis nulla, sed imperdiet libero. Morbi rhoncus eros nec nisi auctor egestas. Quisque ut nisl dictum lectus mollis tristique rutrum eget arcu. Sed sit amet blandit sapien, eget pulvinar enim. Praesent scelerisque quam a leo pellentesque, a congue arcu luctus. Integer augue urna, vestibulum et nulla vel, lacinia volutpat neque.

#### 4.1.3 $k$ - Número de participantes de um torneio aleatório

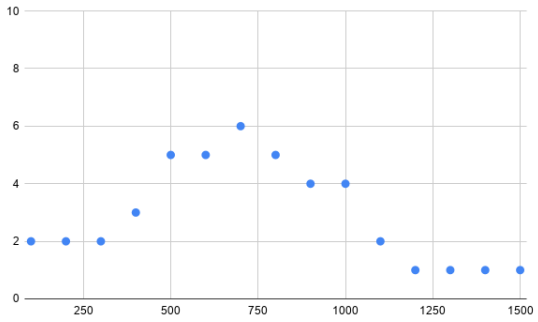
O valor de $k$ foi variado de 3 a 20 com incrementos unitários. A figura abaixo mostra os resultados.




tiam sit amet lectus nec magna aliquet convallis. Curabitur sed lobortis nulla, sed imperdiet libero. Morbi rhoncus eros nec nisi auctor egestas. Quisque ut nisl dictum lectus mollis tristique rutrum eget arcu. Sed sit amet blandit sapien, eget pulvinar enim. Praesent scelerisque quam a leo pellentesque, a congue arcu luctus. Integer augue urna, vestibulum et nulla vel, lacinia volutpat neque.

#### 4.1.4 $\phi$ - Probabilidade de um indivíduo sofrer mutação

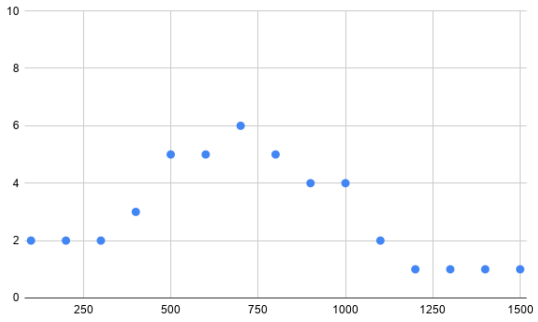
O valor de $\phi$ foi variado de 0.1 a 1 com incrementos de 0.1. A figura abaixo mostra os resultados.




tiam sit amet lectus nec magna aliquet convallis. Curabitur sed lobortis nulla, sed imperdiet libero. Morbi rhoncus eros nec nisi auctor egestas. Quisque ut nisl dictum lectus mollis tristique rutrum eget arcu. Sed sit amet blandit sapien, eget pulvinar enim. Praesent scelerisque quam a leo pellentesque, a congue arcu luctus. Integer augue urna, vestibulum et nulla vel, lacinia volutpat neque.

#### 4.1.5 $\omega$ - Número máximo de gerações consecutivas que não alteram a melhor solução

O valor de $\omega$ foi variado de 100 a 1000 com incrementos de 100. A figura abaixo mostra os resultados.




tiam sit amet lectus nec magna aliquet convallis. Curabitur sed lobortis nulla, sed imperdiet libero. Morbi rhoncus eros nec nisi auctor egestas. Quisque ut nisl dictum lectus mollis tristique rutrum eget arcu. Sed sit amet blandit sapien, eget pulvinar enim. Praesent scelerisque quam a leo pellentesque, a congue arcu luctus. Integer augue urna, vestibulum et nulla vel, lacinia volutpat neque.

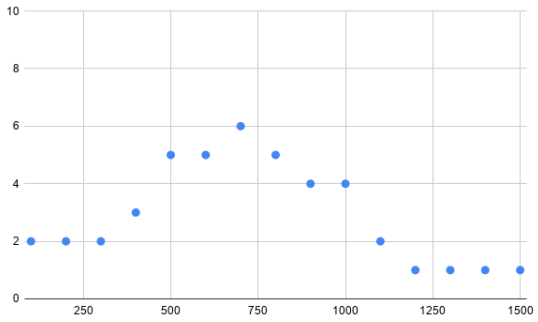
### 4.2 Teste de instâncias

O algoritmo genético e o solver foram executados em 10 instâncias diferentes. A tabela abaixo resume o conjunto de instâncias.

### 4.2.1 Algoritmo Genético

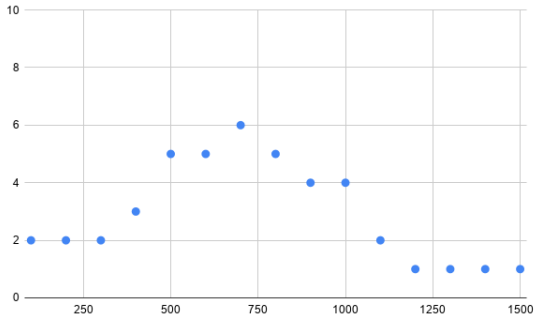
O algoritmo genético foi calibrado usando os melhores resultados de cada parâmetro nos testes apresentados na seção 4.1. A tabela e a figura abaixo apresentam a configuração final de parâmetros do algoritmo genético e os resultados obtidos, respectivamente.

| $\mu$ | $\lambda$ | $k$ | $\phi$ | $\omega$ |
|---|---|---|---|---|
| 1500 | 100 | 3 | 0.5 | 1000 |




### 4.2.2 Execução com o solver

O solver usado na resolução das instâncias foi o CPLEX. Um tempo limite de 1800 segundos foi dado para o retorno de uma solução. A figura abaixo apresenta os resultados




# 5 Conclusão