

# ELE083 Computação Evolucionária

## Trabalho Prático - Makespan

Davi Pinheiro Viana - 2013029912

Rafael Carneiro de Castro - 2013030210

*Minas Gerais, Brasil*

---

*Keywords:* Makespan, Computação Evolucionária, Evolução Diferencial

---

### 1. Introdução

O trabalho final da disciplina de Computação Evolucionária consiste no emprego de uma das técnicas estudadas em sala de aula no decorrer do semestre para a solução de um problema de Engenharia. Tal problema é *Job Shop Scheduling Problem* (JSSP).

O *Job Shop Scheduling Problem* é um problema clássico de otimização combinatória e possui diversas aplicações nas indústrias e empresas. Seu objetivo é obter uma sequência de tarefas a serem executadas de forma a maximizar a utilização dos recursos disponíveis. Por recursos, entende-se como máquinas, pessoas, ou ambos.

A nível de contexto, imagina-se uma linha de produção com X etapas, que devem ser executadas em ordem. Tem-se como entrada um arquivo texto contendo Y linhas, que representam Y pedidos, com X números inteiros em cada linha. Estes números representam o tempo que o pedido daquela linha vai gastar em cada etapa. O objetivo é retornar uma sequência de produção dos pedidos em tempo ótimo, de forma que o conjunto de pedidos será entregue o mais rápido possível.

Para este trabalho, considera-se que a linha de produção tem três etapas. Sendo assim, os arquivos de entrada contêm três colunas.

### 2. Desenvolvimento

Nesta seção serão apresentadas as decisões tomadas para a implementação de um algoritmo que soluciona o problema descrito na *Introdução*.

#### 2.1. Representação de Indivíduos

Para a decisão da forma de se representar indivíduos, primeiro é importante levar em consideração a entrada do algoritmo. Conforme especificações, a entrada é dada por arquivos textos no formato que pode ser visto na Figura 1. Conforme descrito na *Introdução*, cada linha

representa um pedido, e cada coluna representa o tempo gasto em uma etapa de produção (máquina). Os arquivos *entrada\_3.txt*, *entrada\_10.txt* e *entrada\_25.txt* foram disponibilizados junto à especificação deste trabalho.

entrada_25.txt		entrada_10.txt		entrada_3.txt	
1	19, 11, 13	1	11, 22, 29	1	10, 14, 13
2	9, 27, 20	2	26, 30, 13	2	15, 13, 15
3	14, 12, 13	3	19, 16, 17	3	10, 30, 6
4	7, 19, 7	4	9, 29, 9		
5	28, 25, 28	5	20, 25, 28		
6	21, 30, 30	6	8, 6, 30		
7	7, 10, 10	7	30, 28, 26		
8	17, 30, 24	8	30, 17, 26		
9	16, 24, 6	9	23, 26, 12		
10	30, 20, 15	10	6, 10, 19		
11	19, 28, 20				
12	12, 6, 26				
13	9, 23, 26				
14	22, 16, 7				
15	19, 14, 26				
16	24, 22, 21				
17	10, 30, 9				
18	10, 21, 13				
19	29, 22, 24				
20	27, 29, 6				
21	26, 14, 11				
22	28, 30, 11				
23	17, 14, 29				
24	9, 30, 13				
25	26, 15, 12				

Figura 1: Exemplos de arquivos de entrada

Em virtude de este ser um problema combinatório cuja saída é uma sequência de produção dos pedidos, os indivíduos podem ser representados por uma sequência de  $Y$  números inteiros que vão de 1 até  $Y$ , onde  $Y$  é a quantidade de pedidos na entrada. Esta sequência é a que otimiza a produção dos pedidos em questão. Sendo assim, as soluções candidatas do problema são representadas por vetores de números inteiros.

## 2.2. Algoritmo Evolucionário Escolhido

Uma das técnicas e algoritmos estudados em sala de aula foi a chamada *Evolução Diferencial*. Esta é uma técnica amplamente utilizada na solução de problemas através da Computação Evolucionária, possuindo alto desempenho e implementação simples. Sendo assim, esta foi a técnica usada como base para a implementação de um algoritmo que soluciona o problema em questão neste trabalho.

Conforme visto em sala, um pseudo-código para a *Evolução Diferencial* pode ser observado na Figura 2. No código da figura, é importante se destacar a equação:

$$u_{t,i,j} = x_{t,r1,j} + F(x_{t,r2,j} - x_{t,r3,j})$$

onde  $u_{t,i,j}$  representa um indivíduo que sofreu mutação baseado na manipulação vetorial de outros três indivíduos selecionados aleatoriamente. O algoritmo diferencial puro, conforme o representado, é diretamente utilizado para problemas envolvendo otimização contínua. Para problemas combinatórios, como é o caso, adaptações precisam ser feitas, e serão descritas mais a diante.

```
Enquanto algum critério de parada não for satisfeito faça
  Para  $i = 1$  até  $N$  faça
    Selecione aleatoriamente  $r1, r2, r3 \in \{1, \dots, N\}$ 
    Selecione aleatoriamente  $\delta_i \in \{1, \dots, n\}$ 
    Para  $j = 1$  até  $n$  faça
      Se  $\mu_{[0,1]} \leq C \vee j == \delta_i$  então
         $u_{t,i,j} = x_{t,r1,j} + F(x_{t,r2,j} - x_{t,r3,j})$ 
      Senão
         $u_{t,i,j} = x_{t,i,j}$ 
      Fim se
    Fim para
    Se  $f(u_{t,i}) \leq f(x_{t,i})$  então
       $x_{t+1,i} \leftarrow u_{t,i}$ 
    Senão
       $x_{t+1,i} \leftarrow x_{t,i}$ 
    Fim se
  Fim para
   $t \leftarrow t + 1$ 
Fim enquanto
```

Figura 2: Pseudo-código para a evolução diferencial

### 2.3. Operador de Seleção

O operador de seleção de sobrevivência do algoritmo de evolução diferencial implementado pode ser visto na condicional **Se**  $f(u_{t,i}) \leq f(x_{t,i})$  **então**. Nesta condicional, entende-se por  $f(x)$  como a avaliação do *fitness* de um indivíduo. Assim, um indivíduo mutante é selecionado se seu *fitness* é menor que o *fitness* de seu correspondente na população original.

### 2.4. Operadores de Cruzamento e Mutação

No algoritmo da *Evolução Diferencial*, tanto o cruzamento quanto a mutação podem ser representados pela equação:

$$u_{t,i,j} = x_{t,r1,j} + F(x_{t,r2,j} - x_{t,r3,j})$$

já evidenciada na seção *Algoritmo Evolucionário Escolhido*, junto com a condicional que a cerca, a recombinação discreta. Os parâmetros para tal procedimento são o  $C$ , representado uma taxa de recombinação, ou seja, uma probabilidade de um indivíduo de uma população mutante ser idêntico a um indivíduo da população original, ou ser um indivíduo mutante de fato; e  $F$ , que é um fator de escala aplicado ao vetor de diferenças  $x_{t,r2,j} - x_{t,r3,j}$ .

É importante salientar que a *Evolução Diferencial* seguindo o algoritmo da Figura 2 é amplamente utilizada em problemas que envolvem variáveis reais, otimização contínua. Para problemas combinatórios, existem em estudo diversas abordagens e adaptações do mecanismo de cruzamento e mutação. Neste trabalho, optou-se pela abordagem da *Lista de Movimentos*, que é derivada diretamente da equação  $u_{t,i,j} = x_{t,r1,j} + F(x_{t,r2,j} - x_{t,r3,j})$ . As operações da equação são adaptadas de forma a se encaixar em problemas combinatórios, conforme o descrito a seguir:

- Subtração: como no problema em questão os indivíduos são vetores combinatórios, a subtração é dada por uma lista de movimentos que, se aplicada no primeiro vetor, chega-se ao segundo.
- Multiplicação: neste caso, devido ao que foi descrito pela subtração, a multiplicação ocorre entre um escalar e uma lista de movimentos. Assim, definiu-se a multiplicação como sendo a seleção de alguns movimentos da lista de movimentos a que se multiplica.
- Adição: neste caso, estamos adicionando a um vetor combinatório uma lista de movimentos; ou seja, está-se aplicando a este vetor uma dada quantidade de movimentos.

Como resultado destas operações, tem-se um indivíduo recombinado. Esta técnica é de fácil implementação e possui boa eficiência. Mais detalhes sobre esta e outras abordagens podem ser vistos na Referência [3].

### 2.5. Critérios de Parada

Como critérios de parada, primeiro estabeleceu-se um número máximo de de gerações como sendo 500, número este que pareceu razoável pelo ajuste de parâmetros que será descrito a seguir. Outro critério de parada baseia-se no histórico.

### 2.6. Ajustes de Parâmetros

Como mencionado na seção *Operadores de Cruzamento e Mutação*, um dos parâmetros do algoritmo é o  $C$  que é a chance de um indivíduo na população mutante ser um indivíduo realmente mutante. Pode ser que seja adicionada à população mutante um indivíduo idêntico à população original, dependendo do valor de  $C$ . Este parâmetro foi ajustado com o valor de 0.8, fazendo com que a população mutante criada a cada geração tenha uma grande quantidade de indivíduos mutantes, aumentando a abrangência de uma iteração, ou seja, de um teste sobre uma geração, envolvendo população original e população mutante.

O parâmetro  $F$ , que representa o fator de escala, foi ajustado com o valor 0.9, de forma a gerar um indivíduo com uma boa quantidade de permutações de valores partindo de um indivíduo base.

Para o tamanho da população, achamos razoável utilizar o quadrado da quantidade de linhas da entrada, ou seja, o quadrado da quantidade de pedidos. Este valor foi utilizado por causa da variabilidade combinada com os outros parâmetros ajustados. É importante ressaltar que o tamanho populacional foi definido pensando que a entrada com maior quantidade de linhas que será passada ao algoritmo possui 25 linhas. Definiu-se como tamanho máximo da população 1000 indivíduos, para casos de entrada com muitos pedidos. Também é importante destacar que cada geração do algoritmo de *Evolução Diferencial* não aumenta o tamanho da população, ou seja, este valor se mantém constante durante a execução do algoritmo.

### 2.7. Avaliação do Fitness

## 3. Resultados

Esta seção apresentará os resultados encontrados para as entradas *entrada\_3.txt*, *entrada\_10.txt* e *entrada\_25.txt* disponibilizadas junto às especificações do trabalho e ilustradas na Figura 1.

### 3.1. entrada\_3.txt

Executando o algoritmo, o resultado obtido para esta entrada para a melhor sequência é  $[1, 2, 3]$ , com *makespan* igual a 74. Os gráficos da média de *makespan* e melhor indivíduo por geração podem ser vistos na Figura

### 3.2. *entrada\_10.txt*

Executando o algoritmo, o resultado obtido para esta entrada para a melhor sequência é [6, 10, 1, 5, 9, 7, 2, 8, 3, 4], com *makespan* igual a 227. Os gráficos da média de *makespan* e melhor indivíduo por geração podem ser vistos na Figura

### 3.3. *entrada\_25.txt*

Executando o algoritmo, o resultado obtido para esta entrada para a melhor sequência é , com *makespan* igual a . Os gráficos da média de *makespan* e melhor indivíduo por geração podem ser vistos na Figura

## 4. Conclusão

A partir dos gráficos e resultados obtidos na seção de *Resultados*, como os valores de média e melhor indivíduo sempre caem, podemos concluir que o algoritmo possui um resultado satisfatório. O resultado final das execuções podem ter certa variabilidade por causa dos critérios de parada, mas o resultado no geral é bem aceitável. As maiores dificuldades na implementação se deram na criação do algoritmo de cálculo de *makespan* e nas adaptações para uma abordagem combinatória do algoritmo de *Evolução Diferencial*. Contudo, estes obstáculos foram superados e o resultado final foi satisfatório.

## 5. Referências

1. Especificação do trabalho
2. Notas de aula: *Algoritmo de Evolução Diferencial* - CASTRO, Cristiano
3. *Uma Nova Abordagem Para A Evolução Diferencial Em Otimização Discreta* - PRADO, Ricardo Sérgio. Disponível em [http://www.eletrica.ufpr.br/anais/cba/2010/Artigos/65881\\_1.pdf](http://www.eletrica.ufpr.br/anais/cba/2010/Artigos/65881_1.pdf)