# UMA ABORDAGEM MIMÉTICA PARA O PROBLEMA DE CORTE NÃO LINEAR UNIDIMENSIONAL

EDERSON FERREIRA DE JESUS

Resumo. Neste trabalho desenvolvo uma primeira abordagem do problema de corte unidimensional (1D-CSPNL) utilizando metaheurísticas miméticas pela Otimização por Enxame de Partículas com o uso de tribalismo para a obtenção de padrões de corte. Apesar de obter péssimos resultados, os mesmos não são ruins comparados aos obtidos em outros papers por se tratar de uma primeira abordagem.

## Parte 1. Introdução

Dentro da área de pesquisa operacional, um dos primeiros problema a serem abordados no século passado foi o problema de corte (CSP, do inglês Cutting Stock Problem) com Katorovich nos anos 30. Este problema consiste em encontrar uma melhor forma de cortar rolos de tamanhos dados por um pedido, de modo a suprir a demanda do pedido e algum outro objetivo importante para o produtor. Desde então, muito trabalho tem sido desenvolvido nesta área, mas o mais impactante foi o de Gilmore & Gomory, que na década de 60 construiu uma heurística para geração dos padrões muito poderoso, porém apenas para problemas lineares.

Na década de 90, Kennedy e Eberhart desenvolveram uma heurística de computação eolvucionária chamada Otimização por Enxame de Partículas (PSO, do inglês Particle Swarm Optimization), onde o espaço de busca era povoado com partículas de baixa cognição e todas se movimentavam em direção da melhor. Esta heurística tem se mostrado extremamente poderosa para problemas de otimização não linear por precisar apenas de definição no movimento das partículas dentro do espaço de busca, não importanto a complexidade do problema.

Neste trabalho desenvolvi uma implementação de PSO para o 1D-CSPNL seguindo algumas premissas.

### 1. O Problema de Corte

O problema de corte, conforme citado anteriormente, um pedido de produtos provenientes do processamento de matéria prima por cortes em um eixo deve ser satisfeito. Neste caso, queremos atender à demanda minimizando o custo com os rolos e com o tempo de setup das facas para os cortes.

O modelo do problema em questão é

$$\begin{array}{ll} \min & c_1 \sum_{i=0}^{m} x_i + c_2 \sum_{i=0}^{m} H(x_i) \\ \text{s.t.} & \sum_{i=0}^{m} a_{ij} x_i \geq d_j \qquad \forall j = 1, 2, \ldots, n \end{array}$$

onde m é o número de padrões de corte possíveis para o tamanho do rolo mestre, n é o número de produtos envolvidos no pedido e $H(x)$ é a função salto de Heaviside,

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

Este problema pode ser dividido em duas partes:

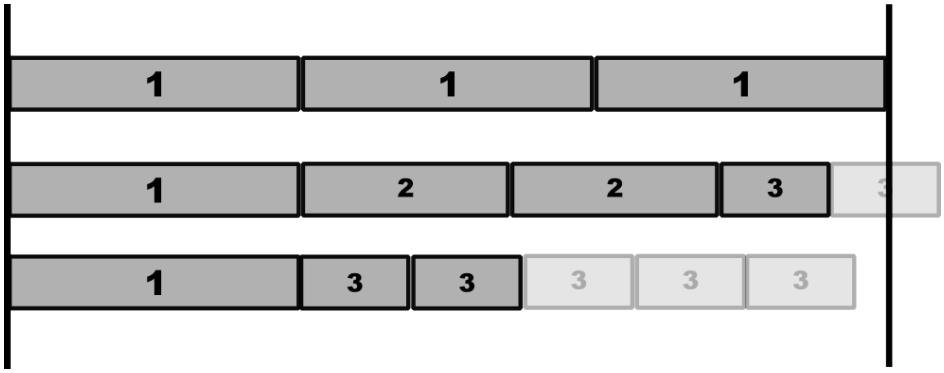
(1) A busca de padrões de corte factíveis para a combinação de produtos para cada rolo mestre;
(2) Dada uma família de padrões de corte, encontrar a melhor configuração para a produção do pedido.

O primeiro problema é nítidamente um problema combinatorial, logo seu crescimento é muito grande para alterações dos parâmetros do problema.

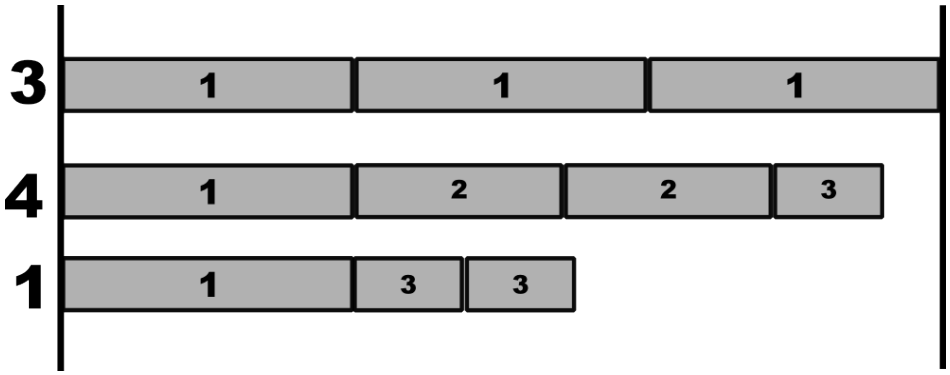
Para o segundo problema, caso tenhamos uma função objetivo linear ele poderia ser resolvido utilizando o método Simplex, que não é o caso. Como a função passo é não-linear, o método Simplex falha na otimização deste objetivo.

### 2. A Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas é uma heurística mimética onde cada partícula tenta imitar a melhor partícula do enxame no momento, baseando-se na movimentação dentro do espaço em direção da melhor partícula do grupo no momento.




(A) Padrões de corte. De cima para baixo: um esquema sem perdas, um esquema com folga insuficiente e um esquema com folga excessiva.




(B) Esquema de corte. Esta figura representa um esquema de corte $(3,4,1)$ para o conjunto de padrões $\{(3,0,0),(1,2,1),(1,0,2)\}$

FIGURA 1.1. Exemplos de padrões de corte e esquemas de corte

A heurística se baseia em partículas simples, sem muita informação ou processamento das informações. Cada agente sabe apenas sua posição atual, sua velocidade de deslocamento, a melhor posição já visitada por si mesmo e qual a melhor posição atual do grupo. Chamando estes valores de $p_i$, $v_i$, $p_i^b$ e $p_g$, respectivamente, podemos definir a situação do sistema como

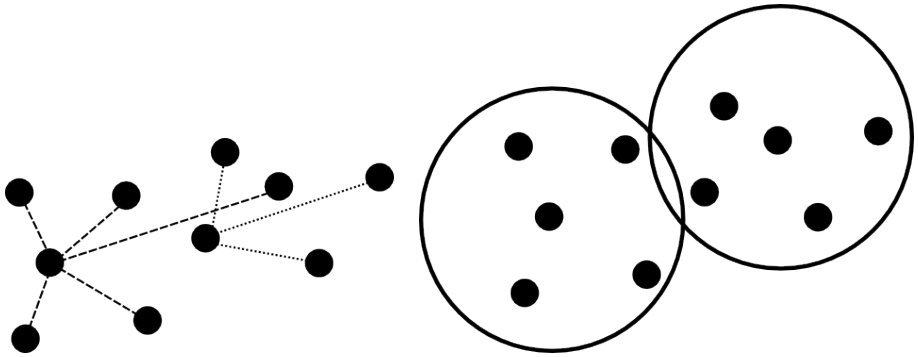
$$\left\{ \begin{array}{lcl} v_{i+1} & = & r_1 \odot v_i \oplus r_2 \odot \left(p_i \ominus p_i^b\right) \oplus r_3 \odot \left(p_i \ominus p_g\right) \\ p_{i+1} & = & p_i \oplus v_{i+1} \end{array} \right. \tag{2.1}$$

Os valores $r_i, i \in \{1,2,3\}$ são aleatórios e dizem o grau de confiança da partícula com relação à viabilidade de cada direção. Pode-se ainda fazer $r_i = I_i \rho_i$, onde $\rho_i$ é um valor aleatório e $I_i$ é um coeficiente constante, dito coeficiente de inércia da velocidade da partícula. Valores pequenos de $I_i$ possibilitam mudanças mais rápidas de velocidade, mas uma pequena varredura do espaço, ao passo que valores maiores causan mudanças menores na valocidade, mas acarretam uma melhor varredura do espaço de busca.

Segundo [CK05], a convergência do método utilizado é convergente dependendo dos valores adotados para as constantes.

A determinação de qual o melhor indíviduo do enxame pode ser feita a partir de qualquer função $f : P \to \mathcal{R}$ que determine o grau de aptidão da partícula com relação ao problema. O tipo de problema a ser resolvido deve fzer exigências apenas na classificação dos elementos: para problemas de maximização, os melhores elementos serão os de função de aptidão maior e para problemas de minimização os menos aptos serão os melhores do grupo.

Os grupos também podem ser separados em grupos com especializações, onde cada grupo tem uma tendência maior a variar mais determinados pontos do espaço. Estes grupos também recebem o nome de tribos, e são mais utilizados em caso de busca de ótimos locais. Estas tribos podem ser estáticas - definidas no momento da criação da população - ou dinâmicas, onde a pertinência dos elementos é determinada pela distância do líder. O segundo caso, por se tratar de um elevado custo computacional, é pouco utilizada.



(A) Vizinhança estática

(B) Vizinhanaça dinâmica (ou por distâncias)

FIGURA 2.1. Exemplos de vizinhanças para PSO com tribalismo

**Algorithm 1** Algoritmo de geração de partículas aleatórias visando minizar a perda inicial.

```
Resíduo = tamanho do rolo mestre.
Para cada produto na carteira faça
    Encontrar o número máximo do produto no resíduo;
    Escolher aleatoriamente uma quantidade que não exceda o valor máximo;
    Atualizar o resíduo com a redução pelo valor utilizado por este produto.
```

## Parte 2. O PSO para o 1D-CSPNL

### 3. Padrões de Corte

Para a implementação de PSO para o 1D-CSPNL, os padrões serão partículas que terão vetores de inteiros, onde o $i$-ésimo elemento do vetor armazena a quantidade do item $i$ no padrão.

Para definir corretamente o espaço das partículas, a soma e a subtração entre elementos será idêntica à soma e subtração de $\mathcal{Z}^j$. A multiplicação por escalar será definida como $\alpha \odot p \equiv ([\alpha p_i]), \forall i \in \{1, 2, \ldots, j\}$, onde $[\cdot]$ indica o arredondamento.

Para o comportamento das partículas dentro do espaço, assumimos que os melhores padrões para o nosso objetivo seriam os padrões que tivessem menor perda, dado que desta maneira seria obtida uma maior quantidade de produtos a partir de um único rolo mestre. Diante disso, para a geração de padrões de corte, teremos como objetivo encontrar a menor perda possível dentre os padrões.

Para alcançar de forma mais breve estes objetivos, é importante partir de uma população inicial já boa, portanto a população inicial foi gerada segundo o pseudo-código descrito no algoritmo (1)

Desta maneira temos uma população com baixas perdas no início.

São realizadas então iterações do algoritmo segundo a equação 2.1 na página anterior. A função de aptidão das partículas será a soma de vários itens, a saber:

- a perda associada ao padrão, com peso 1;
- penalização pela existência de elementos negativos no padrão, com peso 1000 para cada unidade negativa de cada elemento;
- penalização pela infactibilidade do padrão (padrão utiliza mais matéria prima que o rolo mestre fornece), com peso 100 para cada unidade da diferença entre o comprimento do rolo mestre o da quantidade necessária para o padrão.

Após algumas iterações, era iniciado o processo de recolhimento dos bons padrões seguindo o seguinte critério: caso a perda do líder do grupo fosse menor que um certo valor limite, um gerenciador faz uma varredura no grupo localizando as partículas que possuem perda menor que o limite do líder adiocionado uma tolerância. Estes padrões são então removidos do grupo e novos padrões aletórios são adicionados ao grupo e o processo se reinicia até que o gerenciador possua um número razoável de padrões. O pseudo-código está ilustrado no algoritmo (2).

Quando o gerenciador possuir ao menos 10 padrões, inicia-se o processo de encontrar os esquemas de corte, i.e., determinar o número de vezes que cada padrão será utilizado na geração da carteira de pedidos.

**Algorithm 2** Remoção de padrões bons pelo gerenciador.

```
Se o líder tem perda menor que limite
    Para cada partícula no grupo faça
        Se para esta particula temos perda < limite + tolerância
            Realizar captura desta particula
            Remover a partícula do grupo
            Adicionar ao grupo uma nova partícula aleatória.
```

**Algorithm 3** Geração de esquemas de corte aleatórios

```
Gerado = 0;
Residuo = Pedido - Gerado;
Esquema = 0;
enquanto max(residuo) > 0
        Para todo padrao no gerenciador
                Encontrar max(Residuo)
                Encontrar MAX, o máximo de ocorrências do padrão
                                            para superar a demanda
                Se o padrão já é utilizado
                        Incrementar o número de vezes que o padrão é utilizado
                Senão
                        Definir aleatoriamente o número de vezes
                                            que o padrão é utilizado
                                            sem exceder MAX
                Atualizar a quantidade gerada
                Atualizar o resíduo
```

## 4. Esquemas de Corte

Um esquema de corte obedece o mesmo requisito que uma partícula, i.e., deve ser um vetor de inteiros, logo obedece à mesma álgebra utilizada para os padrões de corte.

Para a determinação de um esquema factível de corte, é necessário que a população inicial produza itens o suficiente para cobrir a demanda do pedido. Para isso utilizamos o algoritmo descrito no algoritmo []. Assim, a população inicial é factível.

Foram então realizadas iterações do PSO utilizando como função de aptidão a soma das seguintes parcelas:

- função objetivo do problema original com peso 1;
- penalização por elementos negativos no esquema de corte, com peso 1000 por unidade negativa por elemento;
- penalização por não suprimento da demanda, com peso 100 por unidade por elemento.

Assim foram realizadas 100 iterações do método, e ao fim das iterações a melhor solução do grupo foi utilizada.

# Parte 3. Testes

## 5. Dados

Os dados para testes serão as classes de problemas 1, 2, 3 e 4 utilizadas em [MGdSN09], possibilitando a comparação de resultados. Estes testes foram gerados pela rotina CUTGEN, que cria pedidos aleatórios com base em parâmetro fornecidos pelo usuário. Foram executados 100 testes para cada classe.

As classes foram criadas utilizando os parâmetros ilustrados na tabela...

## 6. Resultados

Os resultados obtidos para a execução do método para as classes descritas estão expostos na 2 na próxima página.

Nas classes 3 e 4 houve uma falha durante a execução da rotina, possivelmente por se tratar de um grande número de objetos. A implementação do método, visando uma variabilidade maior da população, criava tribos com o mesmo tamanho que o tamanho do pedido, logo, a quantidade de elementos alocados na memória cresce com o

| Classe | v1 | v2 | m | d |
|---|---|---|---|---|
| 1 | 0.01 | 0.20 | 10 | 10 |
| 2 | 0.01 | 0.20 | 10 | 100 |
| 3 | 0.01 | 0.20 | 20 | 10 |
| 4 | 0.01 | 0.20 | 20 | 100 |

TABELA 1. Parâmetros utilizados para cada classe

(A) Número médio de itens processados

| Classe | PSO | Symbio1 | Symbio5 | Kombi |
|---|---|---|---|---|
| 01 | 22 | 12,59 | 14,49 | **11,49** |
| 02 | 202,76 | 115,17 | 116,26 | **110,25** |

(B) Numero médio de setups para cada classe

| Classe | PSO | Symbio1 | Symbio5 | Kombi |
|---|---|---|---|---|
| 01 | 13,46 | 3,09 | **2,02** | 3,40 |
| 02 | 35,81 | 6,11 | **5,28** | 7,81 |

TABELA 2. Médias das soluções obtidas. Os itens em negrito indicam qual o método vencedor para a classe.

(A) Erros percentuais para a classe 1

| | Melhor método | Pior método |
|---|---|---|
| Setups | 566,33% (Symbio5) | 388,82% (Kombi) |
| Processados | 99,47% (Kombi) | 51,82% (Symbio5) |

(B) Erros percentuais para a classe 2

| | Melhor método | Pior método |
|---|---|---|
| Setups | 578,21% (Symbio5) | 358,51% (Kombi) |
| Processados | 83,91% (Kombi) | 74,40% (Symbio5) |

TABELA 3. Erros percentuais em relação ao melhor e pior método para cada modalidade

quadrado do tamanho do problema. Portanto, os testes menores (classes 1 e 2) puderem ser executados sem maiores problemas.

## 7. Análise dos Resultados

Esta implementação do PSO se mostrou pior que as outras heurísticas usadas para comparação. O erro percentual com relação à pior e à melhor entre as outras 3 heurísticas estão ilustrados na tabela 3.

Estes erros possivelmente se devem ao fato de o método utilizado se basear em uma premissa errada. Novos testes estão sendo feitos com o uso da distância entre a frequência relativa de itens no padrão e a frequência relativa de itens do pedido. Esta abordagem trará novos e melhores resultados.

Porém, na primeira etapa do processo (a obtenção de padrões factíveis de corte) o método mostrou-se extremamente de veloz e eficaz.

Para as novos testes desta heurística, devido aos péssimos resultados, tomaremos como base a frequência relativa dos elementos no pedido, atualizado a cada iteração visando reduzir o número de setups sem criar um grande excesso de produção, e em métodos que asseguram a variabilidade da população, para que não seja necessário criar tribos muito grandes para a os padrões.

## Referências


[CK05] Maurice Clerc and James Kennedy, *The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space*, IEEE Transactions on Evolutionary Computation (2005).

[CS04] M. Clerc and P. Siarry, *Une nouvelle métaheuristique pour l'optimisation : la méthode des essaims particulaires*, J3eA **3** (2004).

[KES01] James Kennedy, Russel C. Eberhart, and Yuhui Shi, *Swarm intelligence*, Morgan Kauffmann Publishers, 2001.

[MGdSN09] Antonio Carlos Moretti, Rodrigo Rabello Golfeto, and Luiz Leduíno de Salles Neto, *A genetic symbiotic algorithm applied to the one-dimensional cutting stock problem*, Pesquisa Operacional **29** (2009), 365–382.