1 Identificação

- Projeto: Uma adaptação do Método de Haessler para Corte Unidimensional
- Bolsista: Luís Felipe Melo de Miranda
- RA: 071622
- Local de execução: Unicamp - Imecc (Instituto de Matemática, Estatística, e Ciências Computacionais)
- Orientado por: Antônio Carlos Moretti

2 Introdução

O problema do corte unidimensional é uma variação do problema do mochileiro (knapsack problem), um dos desafios clássicos da computação operacional. Dado mochilas de tamanho w_0, temos que encaixar r_i peças de tamanho w_i ($i \in 1, 2, ..., n$), utilizando o menor número de mochilas possíveis. Este problema tem aplicações inúmeras, como na indústria de papel, materiais de construção, entre outros.

Mais que isso, o que se propõe à fazer neste projeto é a utilização de uma heurística dentro de uma heurística–uma forma imaginativa e, é esperado, eficiente de resolver um problema já tão trabalhado.

2.1 O Problema de Corte Unidimensional

Estritamente falando, o problema de corte unidimensional é linear. Ele se resume à, dado um suprimento ilimitado de rolos de largura w_0, minimizar a quantidade utilizada deste para extrair r_i rolos de largura w_i, $i \in 1, 2, ..., n$. Ou seja, sendo A uma matriz contendo todos os arranjos possíveis de tamanhos w_i dentro de um w_0, onde para cada coluna/arranjo $\mathbf{a_j}$ há um $x_j \in \mathbf{x}$, a quantidade do arranjo utilizado, e sendo $\mathbf{r}$ o vetor contendo todos os r_i,

$$\begin{aligned} &\min \sum_{j=1}^{m} x_j \\ &\text{s.a. } A\mathbf{x} = \mathbf{r} \end{aligned}$$

A "deslinearização" do problema provém do fato que minimizar a quantidade de material utilizado é raramente a única consideração. Em seu paper [1], Haessler explica que alterar o arranjo de corte é um processo caro que também deve ser minimizado. Logo, seja definido

$$I(x_j) = \begin{cases} 1 & x_j > 0 \\ 0 & x_j = 0 \end{cases}$$

Uma função contadora de arranjos utilizados. Nossa nova função objetivo seria então

$$C_1 \cdot \sum_{j=1}^{m} x_j + C_2 \cdot \sum_{j=1}^{m} I(x_j)$$

Onde C_1 e C_2 são o custo por rolo w_0 e arranjo utilizado, respectivamente. O último fator a ser considerado em nosso caso é o número limitado de facas em uma máquina de corte, de modo que não possa haver mais que p rolos extraídos de um único w_0. Formalmente, o problema completo seria:

Uma ordem é feita para uma fábrica de papel por rolos de n larguras w_i diferentes, em quantidades r_i. Estes tem que ser obtidos à partir de rolos mestres de tamanho w_0. Há m maneiras de encaixar os tamanhos w_i em w_0 de forma que não seja possível adicionar nem mesmo o menor w_i uma vez mais, levando em conta que o número máximo de rolos à serem obtidos de um único rolo mestre é limitado à p devido ao número limitado de lâminas na máquina de corte. O custo por rolo mestre utilizadoé de C_1, enquanto o custo de alterar a posição das lâminas na máquina de modo à obter um novo arranjo de corte é de C_2 (por arranjo). Em forma algébrica, sendo $\mathbf{N^x}$ o espaço dos naturais de x dimensões,

$$\text{Dado } \mathbf{r}, \mathbf{w} \in \mathbf{N}^n$$
$$p, w_0 \in \mathbf{N}$$
$$A \in \mathbf{N}^{n \times m} \text{ é a matriz que satisfaz}$$

$$\begin{cases} \sum_{i=1}^{n} a_{ij} \leq p \\ w_0 - \min w_i < \sum_{i=1}^{n} a_{ij} \cdot w_i \leq w_0 \\ \forall j \in 1, 2, ..., m \end{cases}$$

E sendo

$$I(x_j) = \begin{cases} 1 & x_j > 0 \\ 0 & x_j = 0 \end{cases}$$

$$\text{minimizar } C_1 \cdot \sum_{j=1}^{m} x_j + C_2 \cdot \sum_{j=1}^{m} I(x_j)$$

$$s.a. \begin{cases} A\mathbf{x} = \mathbf{r} \\ \mathbf{x} \in \mathbf{N}^m \end{cases}$$

2.2 O Método de Haessler

O que parece ser um clássico problema de programação não-linear inteira é na verdade bem mais complicado que isso. O que faz dessa questão um desafio é a matriz A. Em um exemplo real [1], uma ordem por 50 tamanhos diferentes gerou uma matriz de arranjos possíveis com mais de 900,000 colunas. Computar tal matriz é caro demais para tornar métodos clássicos viáveis, e mesmo que o fizesse, o domínio de soluções possíveis ainda seria um poliedro no $\mathbf{N}^{900000}$. Neste mesmo paper, Haessler não só discute tudo isso à fundo como cria uma heurística para a resolução do problema. Esta é baseada em um processo recursivo onde à partir de critérios à serem discutidos (ele refere à estes como aspiration levels), é feita uma busca por um único arranjo, que é então utilizado o máximo de vezes possíveis sem ultrapassar a demanda de nenhuma largura desejada.

$$\begin{gathered}\text{maximizar } x_j \\ \text{s.a. } \mathbf{a_j} \cdot x_j \leq \mathbf{r}\end{gathered}$$

E o vetor $\mathbf{r}$ é então recalculado

$$\mathbf{r'} = \mathbf{r} - \mathbf{a_j} \cdot x_j$$

E é feita uma nova busca utilizando os novos critérios obtidos do novo $\mathbf{r'}$, sendo o processo repetido até que $\mathbf{r'}$ seja um vetor nulo ou próximo o suficiente de tal. A chave do método, é claro, são os critérios e como estes se encaixam no algoritmo em si:

2.2.1 Desperdício

$MAXTL$ (maximum trim-loss) é o máximo de desperdício permitido por arranjo. Ou seja, para todo $\mathbf{a_j}$ à ser utilizado,

$$w_0 - \sum_{i=1}^{n} a_{ij} \cdot w_i \leq \text{MAXTL}$$

Segundo Haessler [1], MAXTL costuma ser entre 3 e 0,6 porcento de w_0, logo:

$$\begin{gathered}\text{MAXTL} = \lambda_1 \cdot w_0 \\ \lambda_1 \in [0,006; 0,03]\end{gathered}$$

2.2.2 Média de rolos por arranjo

MINR e MAXR (minimum and maximum rolls) são o mínimo e máximo número de rolos à serem obtidos de um único w_0. Ou seja, para todo $\mathbf{a_j}$ á ser utilizado,

$$\text{MINR} \leq \sum_{i=1}^{n} a_{ij} \leq \text{MAXR}$$

MAXR fica por critério do usuário, já que é normalmente igual ao máximo da máquina.

MINR é igual ao chão da média de rolos à serem obtidos por w_0 ainda à ser processado. Ou seja, uma vez que $\sum_{i=1}^{n} r'_i \cdot w_i$ é o tamanho total dos rolos que faltam, podemos concluir que $\lceil \frac{\sum_{i=1}^{n} r'_i \cdot w_i}{w_0} \rceil$ é o mínimo de rolos mestres ainda à serem utilizados. Logo,

$$\frac{\sum_{i=1}^{n} r'_i}{\lceil \frac{\sum_{i=1}^{n} r'_i \cdot w_i}{w_0} \rceil} = \lfloor \frac{w_0 \cdot \sum_{i=1}^{n} r'_i}{\sum_{i=1}^{n} r'_i \cdot w_i} \rfloor$$

é o número de rolos médio à ser obtido por w_0 ainda à ser processado. Logo

$$\text{MAXR} = p$$
$$\text{MINR} = \lfloor \frac{w_0 \cdot \sum_{i=1}^{n} r'_i}{\sum_{i=1}^{n} r'_i \cdot w_i} \rfloor$$

2.2.3 Número de arranjos

MINU (minimum usage) é o mínimo do máximo de vezes que o arranjo pode ser utilizado sem ultrapassar a demanda restante. Ou seja, dado

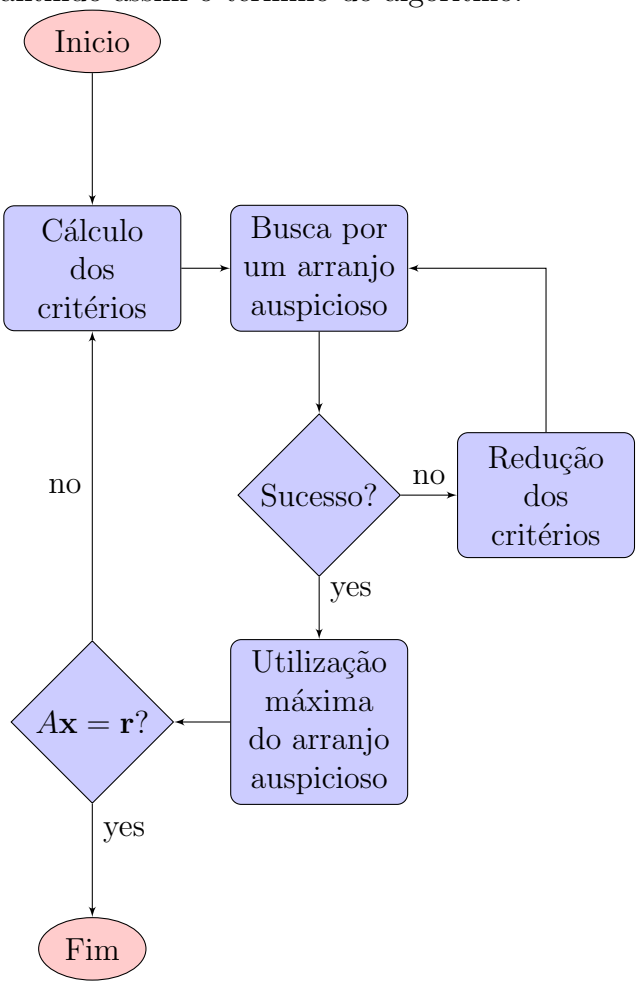
$$\text{maximizar } x_j$$
$$\text{s.a. } \mathbf{a_j} \cdot x_j \leq \mathbf{r'}$$

x_j tem que ser maior ou igual à MINU. Isso garante uma boa, senão perfeita, minimização do número de arranjos utilizados. Haessler afirma que MINU costuma ser entre 50 e 90 porcento dos rolos mestres ainda à serem processados. Logo:

$$\text{MINU} = \lceil \lambda_2 \cdot \frac{\sum_{i=1}^{n} r'_i \cdot w_i}{w_0} \rceil$$
$$\lambda_2 \in [0,5; 0,9]$$

2.2.4 O Algoritmo de Haessler

Faltou discutir o que acontece quando nenhum arranjo é encontrado. A resposta é simples: MINU é reduzido até que algum arranjo seja encontrado. Quando MINU chegar à 1, é feita a busca pelo arranjo com o menor desperdício que possa ser utilizado uma vez sem ultrapassar a demanda, garantindo assim o término do algoritmo:

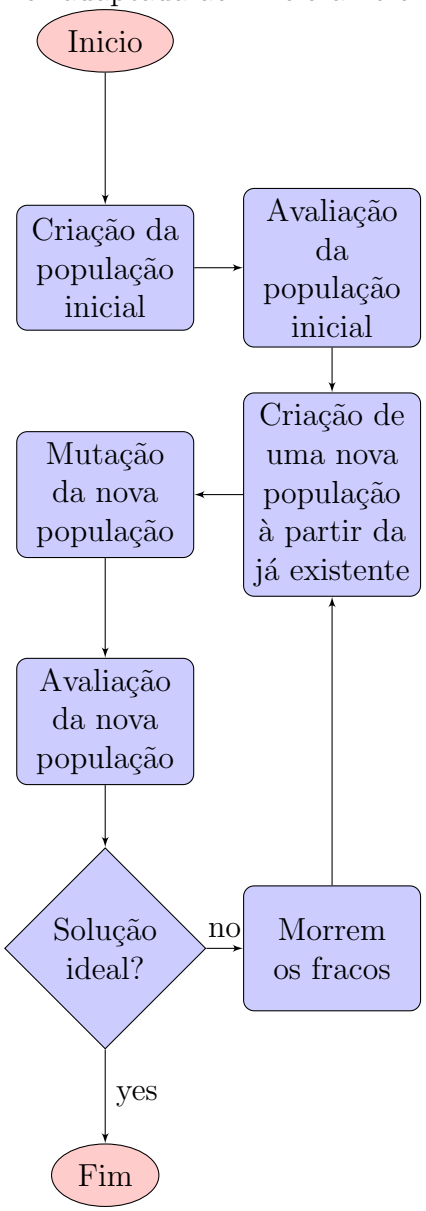


2.3 O Algoritmo Genético

O que Haessler não deixou explicito é a forma da busca por um arranjo auspicioso. Uma busca aleatória seria um desperdício de informação, já que sabemos tanto sobre a natureza do que procuramos (3 critérios), além de demorada e ineficiente. Já uma busca determinística é improvável devido à natureza do problema, que é inteiro e possui várias restrições. O que nos leva ao objetivo deste trabalho: a implementação de um algoritmo genético–

um meio termo entre aleatoriedade e determinismo, uma heurística–para executar a busca do algoritmo de Haessler.

O algoritmo genético (a.g.) simula um eco-sistema, com possíveis soluções atuando a parte de indivíduos lutando por sobrevivência, sendo uma função objetivo o juiz do grau de sucess delas. Soluções com um maior grau de sucesso propagam suas qualidades através de um processo análogo a reprodução meiótica, e tem também uma chance maior de sobreviver. A solução melhor adaptada ao "meio-ambiente" é a solução que procuramos.



O que nos leva aos dois principais aspectos de qualquer a.g.: uma repre-

sentação abstrata do domínio de possíveis soluções, geralmente na forma de um vetor; e um critério, um sistema de avaliação que garanta a convergência dos indivíduos para uma solução propícia.

3 Materiais e Métodos

Na busca pela melhor metodologia, o primeiro fator a ser considerado é o propósito; o segundo os materiais disponíveis. Neste caso, o propósito é simples: testar a efetividade de um algorítmo genético como método de busca do Algorítmo Haessler para o problema de corte. Para tal, temos um laptop Dell Inspiron 700m, com 512 mb de RAM e um processador Intel Centrino. Logo, devido as limitações da máquina e ao potencial econômico deste algorítmo, não nos preocuparemos com o tempo gasto na resolução de problemas. Basta dizer que nenhum deles levou mais do que quinze minutos. O maior problema de implementar o algorítmo genético neste caso é que a função de avaliação, o nosso "'meio-ambiente"', é resultado de três variáveis independentes, sendo estas o desperdício, o número de cortes por padrão e quantas vezes este último pode ser utilizado sem ultrapassar a demanda. Como dar prioridade à um destes? A solução que ofereci no relatório parcial, de "'normalizar"' cada termo resultante destas variáveis:

$$\text{NOTA}(\mathbf{y}) = [\frac{MAXTL-(w_0-\sum_{i=1}^{n} y_i \cdot w_i)}{MAXTL} + \frac{\sum_{i=1}^{n} y_i - MINR}{MINR} + \frac{\min \frac{r_i}{y_i} - MINU}{MINU}]^3$$

falhou. Um resultado fantástico em algum dos termos muitas vezes acabava por encobrir um resultado inaceitável em outro. Optei então por delimitar o espaço de soluções de maneira que dois aspectos sempre estivessem dentro dos parâmetros aceitáveis, e a "'nota meio-ambiente"' dependesse apenas de uma variável. Neste caso, todos os indivíduos-soluções não contém nem menos elementos que $MINR$ nem mais que $MAXR$, e estes também não são em quantidade suficiente tal que, dado o padrão $\mathbf{y}$

$$MINU * \mathbf{y_i} > \mathbf{r'_i}$$

para qualquer i. Nossa nota depende então do desperdício e apenas dele. É claro que, para nos mantermos dentro do espaço factível, as funções de crossover e mutação do algoritmo genético, que geram e mutacionam novas soluções, tem que ser construídas de maneira especial, com muitas correções que (e isso foi testado) causou tremendo aumento no tempo de execução. Para cada busca por um padrão de corte, foi criada uma população de mil "'indivíduos-soluções"', e foram executados 100 ciclos onde outros mil eram criados a partir dos sobreviventes. Foi utilizado completo determinismo nas

funções de crossover e sobrevivência (ou seja, a melhor nota ganhava, sempre), visando eliminar o mais rápido possível padrões que ultrapassavam o tamanho mestre. A função avaliadora pode, então, ser arbitrária, uma vez que não importa mais quão melhor um padrão é, apenas que ele é melhor. O λ_1, referente ao critério $MAXTL$, foi de 0.3, em quanto o λ_2, que diz respeito ao $MINU$, foi de 0.51.

Outro fator importante a ser considerado, que demorou para o bolsista aperfeiçoar, é como reduzir o critério $MINU$ quando este se encontra demasiado alto. Primeiro foi feita uma tentativa onde era reduzido de .05 em .05. Porém, foi constatado que isso resultava em um espaço de soluções grande demais, rápido demais, e se tornava mais difícil conseguir bons padrões uma vez que a área a ser vasculhada era maior do que deveria. Foi então adotado um decremento de 0.005 por ciclo, sendo este resetado para .51 sempre que fosse encontrado um padrão auspicioso. Esta última medida faz com que seja procurado o espaço de um $MINU$ já considerado grande demais. Porém, esta é uma heurística, e como o custo-tempo parecia aceitável, pareceu sensato ao bolsista checar mais vezes.

4 Resultados

4.1 O gerador de problemas

Foi utilizado para esse trabalho o excelente gerador de problemas de corte sugerido por Wascher e Gau [2], e implementado por este último. Baseado nos padrões deste, o problema foi variado em três aspectos: variedade da demanda, tamanho relativo da demanda, e quantidade média da demanda.

Table 1: Problemas testados- CUTGEN1 Input

Dimensão	14	15	38	40
Variedade da demanda	15	20	40	40
Tamanho mestre	1000	1000	1000	1000
Tamanho relativo menor	0.1	0.01	0.01	0.01
Tamanho relativo maior	0.3	0.05	0.3	0.8
Media da demanda por tamanho	20	100	100	100
Seed dos numeros aleatórios	1994	1994	1994	1994
Número do problema gerado	3	3	1	3

A variedade da demanda é o número de tamanhos diferentes da demanda. Este, mais que qualquer outro, é o atributo que define o tamanho do problema. O tamanho mestre é de onde os outros tem que ser extraídos. Tamanho

relativo menor é a razão mínima permitida entre um tamanho da demanda e o rolo mestre, e o maior é análogo. A média da demanda por tamanho é a quantidade média da demanda por tamanho. Seed dos números aleatórios é parte de um gerador de números aleatórios mencionado no paper [2]. Número do problema gerado é qual dos problemas gerados foi utilizado neste projeto (geralmente, o programa gerador gera muitos exemplos de problemas a partir de cada input).

4.2 Quase perfeito...

Dos quatro exemplos apresentados neste relatório, três foram resolvidos muito satisfatoriamente pelo híbrido dos algoritmos Haessler e genético; o último e mais complexo, não. Me referirei a eles pela variedade de suas demandas, sendo os bem-sucedidos 14, 15, e 38. O 40, apesar de próximo do 38, foi resolvido apenas parcialmente. Como pode se ver, o 40 tem um tamanho máximo relativo de 0,8. Isso é, o tamanho máximo de sua demanda é de 0,8 vezes o tamanho mestre.

O tempo de execução do programa variou de exemplo para exemplo. Em todos os casos, foram necessários menos de quinze minutos, sendo os exemplos 14 e 15 executados em menos de cinco minutos. O programa não se comportou bem com o 40, e se tivesse rodado até o fim (considerando quanto do problema completou em quanto tempo), teria demorado mais de uma hora por uma resposta inaceitável.

Table 2: Padrões da Solução 14

Vezes utilizado	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Desperdício
16	2	0	0	0	0	0	0	0	0	1	0	0	0	2	0
11	0	0	0	2	0	1	1	0	0	0	0	0	1	0	9
10	0	2	0	0	0	0	1	0	0	0	0	0	2	0	1
7	0	0	0	1	1	0	0	0	1	0	0	0	1	2	0
5	0	0	1	1	0	1	0	1	0	0	0	0	0	1	0
5	1	0	0	0	0	0	1	0	0	0	0	0	0	4	6
2	0	0	0	0	1	0	0	0	0	0	0	2	0	4	5
1	2	1	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	1	2	3	15
1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	578

A tabela de soluções do problema 38 e 40 (parcial) seriam verdadeiramente gigantes, e por tanto não estão inclusas aqui. Para a resolução destes, devido ao tamanho, o critério $MAXTL$ foi relaxado para 5 porcento do

Table 3: Padrões da Solução 15

Vezes utilizado	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Desp.
23	5	4	1	1	1	2	0	1	0	5	0	4	5	0	1	0
18	0	0	1	0	1	2	5	6	0	5	4	2	7	1	0	0
8	0	0	2	1	4	0	13	0	0	3	3	0	4	0	1	0
3	6	0	2	0	3	0	0	3	1	8	2	2	0	1	1	11
3	2	0	0	0	11	0	2	0	2	0	12	0	0	1	4	3
2	0	0	0	1	11	0	0	3	7	0	1	1	1	1	13	0
2	12	0	0	3	1	0	0	1	0	0	0	5	0	0	12	0
1	5	0	0	3	13	0	0	0	1	0	0	3	0	2	1	0
1	4	0	0	3	1	1	0	0	15	0	1	4	0	0	2	0
1	10	0	0	9	0	0	0	0	0	0	0	6	1	0	1	0
1	13	0	1	2	0	0	0	0	0	0	0	5	0	0	8	62

tamanho mestre em vez de três.

Lembrando do propósito do Algoritmo Haessler, vemos aqui que seus objetivos foram satisfeitos com uma "busca genética". Como pode se ver aqui, o número de padrões é bem limitado, e no caso do 38, foram utilizados apenas quinze padrões um total de seisentas e vinte-três vezes, com apenas 4 porcento de desperdício. Para o 40 foram encontrados 34 padrões de cortes diferentes até que o programa entrou num processo lento onde não eram encontradas soluções boas, apesar de que apenas 1500 dos estimados 6000 tamanhos mestres á serem processados já estavam definidos. Todos os padrões que eram encontrados depois disso só eram utilizáveis uma vez e tinham desperdício inaceitável para um estágio tão cedo do processo, e cada busca dessa demorava uma quantidade excessiva de tempo.

5 Conclusões

Começando pelo pior, discutiremos o problema 40. Fazendo o uso judicioso de printf's, foi possível diagnosticar o problema de tamanhos maiores relativos demasiado grandes, e este é inerente ao Algoritmo Haessler. O processo de achar apenas um padrão por vez e utilizá-lo ao máximo impede planejamento futuro. Quando há um tamanho grande, relativo ao tamanho mestre, não há muitas maneiras de combiná-lo tal que o desperdício seja inferior ao $MAXTL$, e o critério $MINR$ praticamente garante que mais tamanhos grandes que pequenos sobrarão para o final, e sem a versatilidade destes últimos, o problema se torna impossível.

Menciono aqui que meu colega Samuel Martins Barbosa Neto, também

bolsista PIBIC trabalhando em um projeto semelhante ao meu, comentou que tamanhos relativos grandes são a especialidade de seu método: uma heurística para programação linear inteira. Logo, o Haessler-Genético poderia ser bem complementado por este.

Porém, conclúo que o método Haessler-Genético foi, em si, um sucesso. Através deste projeto percebi que o algoritmo genético é uma ferramenta tremenda de busca por espaços complicados, como é o caso do espaço de numeros inteiros do problema da mochila. Por si só, seria talvez ineficiente para resolver o problema de corte, pois o espaço de soluções possíveis seria imenso demais. Com os critérios de Haessler restringindo a região de busca, ele não só se torna efetivo mas também eficiente. A exceção é quando o tamanho relativo maior é próximo demais de 1, e o Algoritmo Haessler se torna ineficiente.

References

[1] Haessler, R. 1975, Controlling CUtting Pattern Changes in One-Dimensional Trim Problems, Operations Research 23, 483-493.

[2] Gau, T. and Wascher, G. (1995). CUTGEN1: A Problem Generator for the Standard One-dimensional Cutting Stock Problem, European Journal of operational Research 84, 572-579.

6 Apoio

Este projeto foi financiado pela SAE por um período de um ano, e não teria sido possível sem tal.