Tratando o Problema de Corte Utilizando Geração de Colunas e Algoritmos Genéticos

MS777 - Orientador: Professor Antonio Carlos Moretti
Aluna: Ana Flavia Lima (093370)

Resumo

Esse projeto trata a resolução de um problema clássico de programação inteira: o problema de corte unidimensional (Figura 1). Como já é conhecido de problemas inteiros, geralmente achar sua solução exata pode ser demorado. No caso de problemas reais, não é incomum constatar-se que existem milhões de padrões possíveis, de forma que enumerá-los e encontrar a solução ótima torna-se uma tarefa complicada. Portanto, às vezes é mais vantajoso utilizar heurística, que encontram uma boa solução em relativamente curto tempo. Assim, a intenção é utilizar um método não-usual para encontrar uma boa solução: dois tipos de heurísticas, a geração de colunas (algoritmo de Gilmore-Gomory) e um tipo de algoritmo genético. Combinados, esses dois métodos podem produzir soluções ainda melhores que se aplicados separadamente, as quais serão analisadas posteriormente
.

Introdução

No problema de corte unidimensional, tem-se uma viga de comprimento L, e vários pedidos de g_i peças de comprimento l_i cada uma, $i = 1, 2, ..., n$, isto é, existem vários pedidos, e cada pedido é de um comprimento, no qual, para cada um desses comprimentos, quer-se uma quantidade. As variáveis de decisão, nesse caso, são quantas vezes cada padrão de corte será utilizado, as quais, devido a restrições de maquinário, devem ser inteiras (ou seja, não se pode mudar o corte no meio do processo). Isso faz com que esse problema seja difícil de se resolver computacionalmente, pois o grande número de variáveis faz com que métodos de enumeração sejam inviáveis.

Para resolver problemas como esse, existem várias heurísticas disponíveis, e no escopo desse trabalho, são procedimentos que procuram soluções razoáveis para problemas cujas

soluções ótimas são difíceis de se encontrar.

Problemas de programação inteira e mista são os mais encontrados hoje em situações práticas: quais rotas utilizar dados vários caminhos entre cidades, alocação de mão-de-obra em uma empresa, etc. Porém, esses tipos de problemas normalmente têm algo da ordem de milhares de variáveis, o que torna ainda mais importante, em inúmeros casos, resolvê-los de modo rápido e prático. Segundo (1), na indústria de papel da época, a diminuição de 1 por cento no desperdício poderia significar uma economia de cerca de 1 milhão de dólares ao ano.

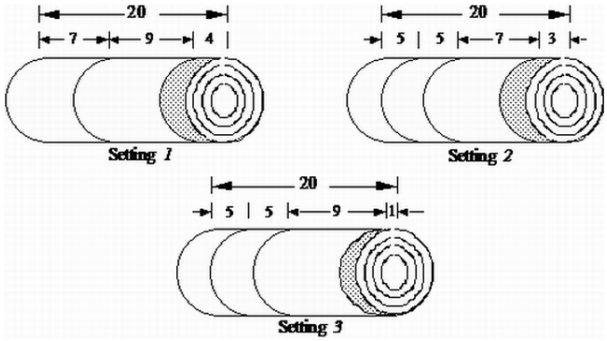

Figura 1: Exemplos de cortes unidimensionais.

Objetivos

Como foi dito anteriormente, qualquer melhora nos resultados obtidos por heurísticas é muito valorizada: com uma leve redução no desperdício, pode-se economizar muito dinheiro. Utilizando isso como motivação, o trabalho apresentado aqui une duas heurísticas (geração de colunas e algoritmos genéticos), na tentativa de encontrar uma solução melhor do que as já conhecidas, comparando os resultados obtidos com aquele encontrado quando a geração de colunas é utilizada separadamente.

O Problema de Corte

Modelagem

Sabendo que L é o comprimento do rolo mestre que deve ser cortado em peças de comprimento l_i a satisfazer uma demanda conhecida g_i, e representando o conjunto dos padrões de corte como uma matriz P_{ij} , onde cada coluna é um padrão diferente (ou seja, o elemento na coluna k e linha m responde por quantas vezes o comprimento l_m é utilizado no padrão de corte k), a modelagem fica:

$$\begin{aligned} \text{Minimizar} \quad & z = \sum_i x_i \\ \text{s.a.} \quad & P_{ij} x_i \geq g_i \\ & x_i \geq 0 \\ & x_i \in \mathcal{Z}^+ \end{aligned}$$

Onde x_i é o número de vezes que o padrão de corte i é utilizado, e $i = 1, 2, ..., n$, n sendo o número de pedidos.

Heurísticas

Para encontrar uma boa solução para o problema de corte, o algoritmo de Gilmore-Gomory será utilizado para gerar soluções (as quais dificilmente serão ótimas) do problema original. Nem todas essas soluções serão inteiras, então um método para torná-las inteiras será utilizado, e assim a população inicial do algoritmo genético será gerada.

Já no algoritmo genético, nao serão gerados novos padrões, apenas serão procuradas novas soluções utilizando as operações usuais de mutação, crossover e torneio.
Assim, as soluções de cada iteração no algoritmo de Gilmore-Gomory serão guardadas em uma matriz, onde cada linha será um gene da população inicial do Algoritmo Genético. Como parte das soluções não é inteira, os elementos dessa matriz são arredondados para cima (para que os rolos cortados continuem a satisfazer a demanda).

Geração de Colunas

Utilizando a modelagem descrita anteriormente, esse método encontra, a cada iteração, um padrão novo de corte, adicionando uma nova coluna à matriz do problema. Inicialmente, tem-se padrões homogêneos (isto é, são compostos de apenas um comprimento), e a matriz do problema será uma matriz quadrada diagonal, onde os elementos não-nulos correspondem ao número máximo de vezes que cada comprimento pode ser cortado no rolo mestre.

Para encontrar novos padrões de corte, um subproblema é resolvido chamado Problema da Mochila.

O Problema da Mochila

Esse o subproblema é utilizado na Geração de Colunas para a obtenção de novos padrões de corte. Existem várias formulações a respeito das variáveis (contínuas, binárias,

etc), porém nesse caso elas serão inteiras. Originalmente, a modelagem foi formulada para resolver o problema de maximizar a utilidade de certos objetos que serão colocados em uma mochila, onde existe um grupo de candidatos dos quais deve ser escolhido um subgrupo cuja soma dos pesos de seus integrantes respeita a capacidade da mochila, e que dá a solução ótima.

Se tomarmos v_i como as variáveis de decisão, que indicam quantas unidades de cada item i serão utilizadas ($i = 1, 2, ..., n$), sendo que cada um desses itens tem uma utilidade u_i e um peso p_i correspondentes, e P como a capacidade máxima da mochila, a formulação é:

$$\begin{aligned} \text{Maximizar} \quad & z = \textstyle\sum_i u_i v_i \\ \text{s.a.} \quad & p_i v_i \leq P \\ & x_i \geq 0 \\ & x_i \in \mathcal{Z}^+ \end{aligned}$$

No algoritmo de Gilmore-Gomory, utiliza-se o preço-sombra das variáveis na função objetivo do problema da mochila, e os tamanhos l_i como o "peso"na restrição ($i = 1, ..., n$).

O Algoritmo da Geração de Colunas

A ideia utilizada aqui é muito simples: no Simplex, quando procura-se uma coluna que melhore a solução, em vez de utilizar-se de métodos do tipo da enumeração (os quais, na maioria dos casos, retornam um número muito grande de alternativas), procura-se uma alternativa resolvendo-se um problema auxiliar (ou seja, o Problema da Mochila).

É assumido que existe matéria-prima para atender a todos os pedidos. Um pedido baseia-se em N_i peças de l_i comprimentos, $i = 1, 2, ..., n$ (onde n é o número de pedidos). Quer-se saber quantas vezes cada padrão de corte será utilizado, então as variáveis de decisão serão x_j, onde j é o número de padrões de corte.

Analizando o problema, fica claro que, para cada comprimento l_i cortado pelos padrões, sua quantidade deve satisfazer à demanda. Assim, a formulação fica:

$$\begin{aligned} \text{Minimizar} \quad & z = \textstyle\sum_i u_i x_i \\ \text{s.a.} \quad & P_{ji} x_i \geq D_i \\ & x_i \geq 0 \\ & x_i \in \mathcal{Z}^+ \end{aligned}$$

Onde Pji é a matriz dos padrões (cada padrão de corte representa uma coluna), D_i é a demanda e c_j são os custos, que nesse caso são todos 1. Inicialmente, os padrões são homogêneos, o que significa que Pji começa como uma matriz quadrada diagonal, onde

cada elemento não-nulo Pii representa o número de vezes que o comprimento l_i cabe no Rolo-Mestre.

A cada iteração um Problema da Mochila é resolvido: os coeficientes da função objetivo são os preços-sombra do problema de corte, e os coeficientes da restrição são os tamanhos da demanda l_i. Assim, é como se os cortes que serão feitos no padrão tivessem que "caber"no comprimento do Rolo-Mestre, e ao resolver o problema, o resultado procurado fosse como fazer isso da melhor maneira (daí os preços-sombra na função objetivo):

$$\begin{aligned} \text{Maximizar} \quad & z = \textstyle\sum_i S_i v_i \\ \text{s.a.} \quad & l_i v_i \leq T \\ & v_i \leq 0 \\ & v_i \in \mathcal{Z}^+ \end{aligned}$$

Aqui, T é o tamanho do Rolo-Mestre, S_i são os preço-sombra e v_i são as variáveis de decisão, ou seja, quantas vezes o comprimento i estará no padrão de corte.

As iterações seguem, até que o preço-sombra no novo padrão de corte seja menor que $1 + \lambda$, onde λ é um valor pequeno (normalmente, 10^{-3}). Isso acontece pois, quando esse ponto da heurística é atingido, o problema da mochila não tem mais solução, caso tente ser resolvido.

Algoritmo Genético

Algoritmos Genéticos utilizam-se de operações como crossover e mutação para procurar novas soluções a partir de uma dada população inicial, e podem ser usados para propósitos diferentes, com tipos de seleção diferentes. O algoritmo tradicional é composto dos seguintes passos:

Algorithm 1 Algoritmo Genético

```
1: Gerar a população inicial
2: Avaliar cada indivíduo da população
3: while critério de para não for satisfeito do
4:     Selecionar os indivíduos mais aptos
5:     Criar novos indivíduos, utilizando crossover e mutação
6: end while
```

Nesse caso, a intenção é usar essa ferramenta na otimização. Para tal, será utilizada a técnica do torneio na seleção dos indivíduos mais aptos. O critério de parada é definido como sendo a tolerância da diferença entre as duas últimas soluções, o qual, junto as taxas de mutação e crossover, são parâmetros regulados de acordo com o resultado desejado, pois podem manipular a convergência da heurística.

Um algoritmo genético que converge muito rápido pode não levar a uma solução muito melhor do que a inicial, porém algo que demore muito para convergir pode significar desempenho computacional desnecessário.

Torneio

Nessa fase, escolhe-se aleatoriamente, para cada gene da população, outro gene com o qual é feito a comparação para encontrar a melhor aptidão, que é copiado naquele que tem a pior aptidão.

Crossover

Aqui, para cada gene, existe uma variável que assume valores contínuos aleatórios entre 0 e 1. Caso essa variável seja menor que a taxa de crossover, é escolhido aleatoriamente outro gene com o qual será feito o crossover.

Em relação ao ponto onde será feito o crossover, também é utilizada uma variável que pode assumir valores aleatórios, porém dessa vez esses valores são inteiros e o intervalo é de 0 até o número de casas do gene.

Nesse trabalho, a taxa de crossover escolhida foi de 70%.

Mutação

Escolhida a taxa de mutação (algo próximo de 1 por cento normalmente), para cada casa de cada gene é sorteado um número aleatório entre 0 e 1. Caso esse número seja menor ou igual à taxa escolhida, o valor do gene é mudado para um número aleatório, inteiro, entre 0 e MAX (onde MAX é o maior valor que qualquer casa de um gene pode ter).

No decorrer dos testes da heurística, percebeu-se uma convergência muito rápida. Então, resolveu-se aumentar um pouco a taxa de mutação, colocado-a em 5%. Como o leitor verá adiante, a demanda máxima para os comprimentos pedidos é de 288, por isso o número MAX escolhido foi de 300.

Critério de Parada

Para isso, é escolhida uma tolerância (nesse trabalho, de valor igual a 1), e caso o módulo da diferença entre a solução anterior com a melhor solução atual seja maior, o algoritmo é interrompido. Também foi estabelecido um mínimo de 30 iterações para evitar a convergência prematura.

Penalização

No decorrer da heurística, frequentemente podem ser geradas soluções que não atendem à demanda. Nesse caso, uma penalização foi feita: para cada tamanho da demanda não atendido, foi adicionada uma penalização na aptidão. Para calcular a penalização, calculou-se a difereça entre o número de tamanhos l_i que a solução dá, e o número de tamanhos da demanda. Esse valor foi adicionado à aptidão.

Aptidão

Depois de muita reflexão sobre como julgar uma solução melhor ou pior que outra, ficou-se em dúvida sobre dois critérios: o desperdício e o valor da função objetivo. De fato, uma solução que atende à demanda a um preço razoável, mas que gera muito desperdício não é tão boa quanto uma que apenas atende à demanda, com um desperdício mínimo. Então, a aptidão escolhida para esse caso foi:

$$P + D + 3FObj$$

Onde P é a penalização, D é o desperdício e $FObj$ é o valor da função objetivo daquela solução.

Materiais e Métodos

O problema foi resolvido em AIMMS, e foi retirado de (2), e foi resolvido em um HP Pavilion, com processador Intel Centrino.

Análise dos Resultados

O problema foi:

Tamanho do Rolo Mestre: 90 cm

Tamanhos	60 cm	30 cm	25,5 cm	20 cm	17,25 cm	15 cm	12,75 cm	10 cm
Demanda	3 unid.	21 unid.	94 unid.	50 unid.	288 unid.	178 unid.	112 unid.	144 unid.

Os padrões de corte gerados pela Geração de Colunas foram:

Os genes gerados inicialmente para a população inicial foram as soluções de cada iteração desse algoritmo:

Valor da função objetivo final na geração de colunas: 163
Valor do desperdício final na geração de colunas: 1,29
Aptidão dessa solução (de acordo com o AIMMS): 492

Tamanhos	60 cm	30 cm	25,5 cm	20 cm	17,25 cm	15 cm	12,75 cm	10 cm
Padrão 1	1							
Padrão 2		3						
Padrão 3			3					
Padrão 4				4				
Padrão 5					5			
Padrão 6						6		
Padrão 7							7	
Padrão 8								9
Padrão 9	1			1				1
Padrão 10			3				1	
Padrão 11				4				1
Padrão 12			1		3		1	
Padrão 13					3		3	

Padrões	1	2	3	4	5	6	7	8	9	10	11	12	13	
Solução 1	3	7	31,3	12,5	57,6	29,7	16	16						
Solução 2	3	7	31,3	12,5	57,6	29,7	16	16	3					
Solução 3		7	31,3	11,8	57,6	29,7	16	15,7	3					
Solução 4		7		11,8	57,6	29,7	16	15,7	3	31,3		11,8		
Solução 5		7			57,6	29,7	16	14,4	3	31,3		11,8	94	
Solução 6		7			1,2	29,7	2,57	14,4	3			11,8	94	2
Solução 7		7				30	2	15	3			12	94	2

Assim, a população inicial do Algoritmo Genético foi gerada arredondando-se para cima todas as soluções. Isso foi feito devido ao fato de que são necessárias soluções inteiras para a segunda parte de heurísticas, e deve-se atender à demanda em todos os genes (isto é, arredondando-se para cima dá a certeza de que a demanda ainda é satisfeita).

A partir daqui, são mostrados os dados finais do algoritmo genético:

Padrões	1	2	3	4	5	6	7	8	9	10	11	12	13
Solução 7	3	7	32	13	58	30	3	16	3				
Solução 6		7	32	13	58	30	3	16	3				
Solução 5		7	32		2	30	12	16	3				
Solução 4	3	7	32	13	58	30	3	16	3				
Solução 3		7	32		2	30	12	16	3				
Solução 2		7	32		2	30	12	16	3				
Solução 1		7	32		2	30	12	16	3				

Tabela 1: População final do algoritmo genético.

Como se pode ver, não foram geradas soluções com aptidões melhores. Uma justificativa para isso é o fato de que é muito fácil gerar-se uma solução que não atende à demanda no decorrer da heurística, devido aos crossovers, principalmente.

Outra justificativa pode ser o fato de que existem outros padrões de corte que podem levar a soluções melhores, porém não foram encontrados na geração de colunas.

Desperdício	
Solução 7	871.5
Solução 6	781.5
Solução 5	448.5
Solução 4	871.5
Solução 3	448.5
Solução 2	448.5
Solução 1	448.5

Tabela 2: Desperdício de cada solução.

Aptidão	
Solução 7	1457.5
Solução 6	1358.5
Solução 5	1107.5
Solução 4	1457.75
Solução 3	1107.5
Solução 2	1107.5
Solução 1	1107.5

Tabela 3: Aptidão final de cada solução.

Objetivo	
Solução 7	165
Solução 6	162
Solução 5	102
Solução 4	165
Solução 3	102
Solução 2	102
Solução 1	102

Tabela 4: Valor da função objetivo de cada solução final.

Tamanhos	60 cm	30 cm	25,5 cm	20 cm	17,25 cm	15 cm	12,75 cm	10 cm
Solução 1	3	21	96	3	10	180	84	147
Solução 2	3	21	96	3	10	180	84	147
Solução 3	3	21	96	3	10	180	84	147
Solução 4	6	21	96	55	290	180	84	147
Solução 5	3	21	96	3	10	180	84	147
Solução 6	6	21	96	55	290	180	84	147
Solução 7	6	21	96	55	290	180	84	147

Tabela 5: Quandidade dos comprimentos da demanda em cada solução final.

No entanto, foram encontradas, durante alguns testes, soluções cuja função objetivo era consideravelmente menor do que a solução da geração de colunas, mas que não produziam nenhum corte de um certo comprimento da demanda, por exemplo. Ou que até produziam, mas a quantidade era muito pequena em relação àquela pedida pela demanda.

Penalização	
Solução 7	91
Solução 6	91
Solução 5	353
Solução 4	91
Solução 3	353
Solução 2	353
Solução 1	353

Tabela 6: Penalização de cada solução final.

Nesses casos, talvez fosse interessante analisar-se, futuramente, como ficaria a solução caso fossem aumentados o número de vezes que o padrão homogêneo dessa quantidade é utilizado, e se a solução ficaria melhor assim.

Quando a demanda não é atendida, mas por muito pouco, seria interessante também fazer-se uma análise de sensibilidade, para saber-se como ficaria a função objetivo e o desperdício caso a solução fosse mudada.

Conclusões

Nesse trabalho, foi feito um experimento na tentativa de melhorar a solução dos problemas de corte unidimensionais, clássicos na programação inteira. Como foi visto na análise de dados, não foram encontradas soluções satistafórias, em termos de melhora da função objetivo, respeitando as restrições. Porém, os resultados poderiam ser úteis para serem analisados posteriormente, em busca de alguma melhora, pois encontraram-se soluções cuja função objetivo é menor, e seria interessante pesquisar sobre uma possível análise de sensibilidade que poderia fazer com que a solução atendesse a demanda, a um bom preço.

Referências Bibliográficas

(1) Gilmore, P. C.; Gomory, R.E; A Linear Programming Approach to the Cutting Stock Problem - Part II.

(2) Chvátal, V.; Linear Programming.