Planejamento de produção através do dimensionamento de lotes de itens únicos

Pedro Henrique Simões de Oliveira

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Carlos Eduardo Ferreira

São Paulo, Janeiro de 2011

Planejamento de produção
através do
dimensionamento de lotes
de itens únicos

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Pedro Henrique Simões de Oliveira
e aprovada pela Comissão Julgadora.

Banca Examinadora:

- Prof. Dr. Carlos Eduardo Ferreira (orientador) - IME-USP.
- Prof. Dr. José Coelho de Pina Júnior- IME-USP.
- Profa. Dra. Maria do Socorro Nogueira Rangel - UNESP.

Dedico aos meus pais.

Agradecimentos

Agradeço à minha família pelo apoio incondicional dado durante o tempo de elaboração deste trabalho. Ao meu orientador, pela enorme paciência e compreensão durante todo o processo, fica o meu muito obrigado.

E, a todas as pessoas que, de alguma forma, me estimularam ou desestimularam durante a realização do meu mestrado, agradeço também, pois sem elas não teria conseguido terminá-lo.

Resumo

Este texto trata de um dos temas fundamentais no planejamento de produção, o problema de dimensionamento de lotes de um único item. Uma descrição sucinta e informal do problema segue abaixo.

Considere um intervalo de tempo dividido em períodos e que a cada período de tempo está associada a demanda de um item. Dados os custos e as eventuais restrições na produção e no armazenamento, determine os períodos em que se produzirá e em que quantidade para que as demandas sejam atendidas com o menor custo possível, respeitando as restrições impostas.

Apresentamos aqui resultados sobre a estrutura ótima do problema, sobre complexidade e algoritmos para os casos básicos do problema.

Abstract

This text studies one of the core subjects in production planning, the single-item lot-sizing problem. A brief and informal description of this problem follows below.

Considering a time interval split into time periods and that there is a demand of an item associated with each time period. Given production and holding costs and possibly production and holding restrictions, determine in which periods the production must occur and in which quantity, in order to attend the demands with a minimum cost, without violate any restriction.

Here, it will be shown some results about the optimal structure of the problem, about the complexity and algorithms for the simpler cases.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A produção de bens de consumo está intimamente ligada à nossa vida cotidiana. Em grande medida, a alta disponibilidade e o relativo baixo custo de itens de necessidade básica tornam nossas vidas muito mais fáceis do que a de nossos antepassados. Para verificarmos isso, basta olharmos o crescente aumento na expectativa de vida mundial ao longo das décadas.

Essa facilidade de obtenção de produtos a baixo custo se deve, principalmente, a avanços técnicos nos processos de produção e também a um melhor entendimento das grandezas envolvidas e das interrelações existentes na cadeia de produção de um determinado item.

O preço dos produtos está intrinsecamente ligado ao custo de produzi-los e, portanto, minimizar o custo de produção aumenta a competitividade de quem os produz. E para isso é importante entender os diversos fatores que afetam o custo total de produção, tais como custos de armazenamento e custos fixos associados a atividades, tais como limpeza, preparo das máquinas, etc.

Podemos definir um *planejamento de produção* de diversas formas. Uma delas é como a determinação de quanto se deve produzir de cada item, em cada momento, de forma que a demanda dos itens seja atendida. A essa quantidade a ser produzida, denominaremos *lote* e ao problema de se determinar o planejamento de produção de custo mínimo, denominaremos de *problema de dimensio-*

namento de lotes (em inglês, *Economic Lot-Sizing*). Nesse trabalho nos preocuparemos apenas com a produção de itens de um único tipo (*Single-item*).

Essa abordagem sistemática que visa à minimização do custo de produção tem sua origem em um problema correlato denominado *Economic Order Quantity*, apresentado em um artigo do início do século passado [Har13], que se preocupa não em produzir quantidades que atendam às demandas, mas sim em realizar compras de produtos de forma que se possa atender à demanda dos mesmos, minimizando o custo total de compra e armazenamento dos itens.

Vamos agora descrever formalmente o problema de dimensionamento de lotes de itens únicos, na seção seguinte.

1.1 Descrição do problema

O nosso objetivo final é minimizar o custo total de produção de forma a atender as demandas existentes e, para isso, precisamos identificar quais são as variáveis envolvidas e quais restrições devem ser respeitadas.

Primeiramente, vamos sempre, a partir de agora, considerar um intervalo de tempo fixo sobre o qual nos concentraremos em resolver o problema. Esse intervalo é usualmente denominado *horizonte de planejamento* e é dividido em n *períodos de tempo*. Esses períodos podem ser dias, semanas, turnos de trabalho ou qualquer outra divisão que faça sentido para o problema específico que esteja sendo analisado.

Em cada período de tempo t, teremos que atender uma demanda D_t. Vamos considerar que existe um custo p'_t de se produzir uma unidade do item de interesse no período t, um custo h_t de se armazenar uma unidade e um custo fixo f_t de se produzir naquele período de tempo (por exemplo, preparação de máquinas, limpeza, etc). Além disso, vamos definir as variáveis x_t, que indicam quantas unidades foram produzidas, s_t que indicam o total de unidades armazenadas e $y_t \in \{0, 1\}$

que indicam se pode ou não haver produção naquele período de tempo.

Com isso em mãos, podemos começar a modelar o problema como um programa linear inteiro-misto (*MIP*).

Por conveniência utilizaremos a partir de agora a seguinte notação:

$$D_{i,j} = \sum_{t=i}^{j} D_t.$$

Considerando o que foi dito até então, nossa função objetivo é:

$$\min \sum_{t=1}^{n} (p'_t x_t + h_t s_t + f_t y_t). \tag{1.1}$$

Vamos agora analisar as restrições do problema. Uma condição que queremos é $s_0 = s_n = 0$, isto é, consideramos que não temos nada antes do nosso horizonte de planejamento e não guardaremos nada após nosso horizonte de planejamento. Da mesma forma, uma restrição desejada é

$$\sum_{t=1}^{n} x_t = D_{1,n}, \tag{1.2}$$

que nos garante que não produziremos nada além do que foi demandado. E além disso, ao longo do horizonte de planejamento, o que está sendo produzido tem que ser sempre suficiente para suprir a demanda até o presente momento. Isso pode ser expresso da seguinte forma:

$$\sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n. \tag{1.3}$$

Outra restrição que queremos garantir é que em todos os períodos de tempo em que houver produção e somente nesses períodos, seja computado o custo fixo. Uma forma de garantir isso é

utilizando a seguinte restrição:

$$x_t \leq y_t D_{1,n}, \text{ para } t = 1, \ldots, n. \tag{1.4}$$

1.1.1 Interpretando o problema como um fluxo de custo mínimo

Vamos apresentar aqui uma representação do problema de planejamento de produção como um fluxo de custo mínimo em uma rede. Essa representação se mostrará útil para obtermos uma série de resultados relativos à estrutura do problema.

Vamos considerar um grafo dirigido $G(V, A)$ cujos vértices são os períodos de tempo $t = 1, \ldots, n$ com uma demanda D_t, além de um vértice artificial o com demanda $-D_{1,n}$.

O conjunto de arcos A é composto por arcos (o, t), para $t = 1, \ldots, n$ e $(t, t+1)$, para $t = 1, \ldots, n-1$. O fluxo $s_t = \chi(t, t+1)$ nos arcos $(t, t+1)$ representa a quantidade armazenada ao final do período t e, além disso, esse fluxo terá custo igual a $h_t s_t$. Já o fluxo $x_t = \chi(o, t)$ nos arcos (o, t) representa a quantidade produzida no período t. Se $x_t > 0$ essa aresta terá custo igual a $f_t + p_t x_t$.

A figura 1.1 ilustra a representação do fluxo, propriamente dito, sem a presença da função custo nos arcos.

Usando a interpretação do planejamento como um fluxo em uma rede, como ilustra a figura 1.1 e observando que não há acúmulo de fluxo nos nós de 1 até n, derivamos a seguinte restrição

$$s_{t-1} + x_t = s_t + D_t, \text{ para } t = 1, \ldots, n. \tag{1.5}$$

Utilizando as restrições que mostramos até aqui, a modelagem do nosso problema como um *MIP*

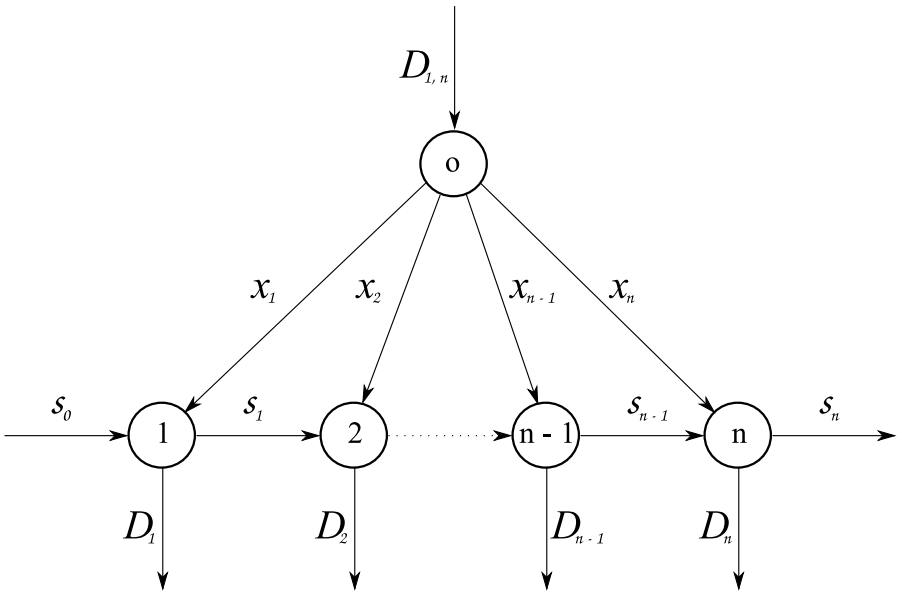


Figura 1.1: Representação do planejamento como fluxo em uma rede.

fica:

$$\min \sum_{t=1}^{n} (p'_t x_t + h_t s_t + f_t y_t) \tag{1.6}$$

sujeito a:

$$s_{t-1} + x_t = s_t + D_t, \text{ para } t = 1, \ldots, n \tag{1.7}$$

$$\sum_{t=1}^{n} x_t = D_{1,n} \tag{1.8}$$

$$\sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n \tag{1.9}$$

$$x_t \leq y_t D_{1,n}, \text{ para } t = 1, \ldots, n \tag{1.10}$$

$$x_t \geq 0, s_t \geq 0, \text{ para } t = 1, \ldots, n \tag{1.11}$$

$$s_0 = s_n = 0 \tag{1.12}$$

$$x_t \in \mathbb{R}, s_t \in \mathbb{R}, y_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{1.13}$$

Como consideramos os custos unitários não negativos a restrição (1.8) se torna redundante na formulação do problema, pois será gantida por (1.9) e (1.7) e pelo fato de $s_0 = s_n = 0$, mas foi mantida por clareza.

1.1.2 Eliminação de variáveis na modelagem do problema

Para facilitar o tratamento do problema, mostraremos que é possível formulá-lo com um número menor de variáveis.

Abaixo segue como podemos eliminar as variáveis x_t ou s_t da formulação do nosso problema.

- Eliminação das variáveis x_t

  Reescrevendo (1.7) isolando x_t, temos

$$x_t = D_t + s_t - s_{t-1},\ t : 1 \leq t \leq n. \tag{1.14}$$

  Substituindo na função objetivo (1.1), temos

$$\sum_{t=1}^{n} p'_t x_t + \sum_{t=1}^{n} h_t s_t + \sum_{t=1}^{n} f_t y_t =$$

$$\sum_{t=1}^{n} p'_t (D_t + s_t - s_{t-1}) + \sum_{t=1}^{n} h_t s_t + \sum_{t=1}^{n} f_t y_t =$$

$$\sum_{t=1}^{n} p'_t D_t + \sum_{t=1}^{n} p'_t (s_t - s_{t-1}) + \sum_{t=1}^{n} h_t s_t + \sum_{t=1}^{n} f_t y_t =$$

$$\sum_{t=1}^{n} p'_t D_t + \sum_{t=1}^{n} p'_t s_t - \sum_{t=1}^{n-1} p'_{t+1} s_t + \sum_{t=1}^{n} h_t s_t + \sum_{t=1}^{n} f_t y_t =$$

$$\sum_{t=1}^{n} p'_t D_t + \sum_{t=1}^{n-1} (p'_t - p'_{t+1} + h_t) s_t + (p'_n + h_n) s_n + \sum_{t=1}^{n} f_t y_t.$$

Se chamarmos de $h_t^{'} = p_t' - p_{t+1}' + h_t$, para $t = 1, \ldots, n-1$ e $h_n^{'} = p_n' + h_n$ e, além disso, se observarmos que $\sum_{t=1}^{n} p_t' D_t$ é uma constante, podemos remodelar o problema sem as variáveis x_t da seguinte forma:

$$\min \sum_{t=1}^{n} (h_t^{'} s_t + f_t y_t) \tag{1.15}$$

$$\textbf{sujeito a:} \quad s_t - s_{t-1} \leq y_t D_{1,n} - D_t, \text{ para } t = 1, \ldots, n \tag{1.16}$$

$$s_t \geq 0, \text{ para } t = 1, \ldots, n \tag{1.17}$$

$$s_0 = s_n = 0 \tag{1.18}$$

$$s_t \in \mathbb{R}, y_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{1.19}$$

- Eliminação das variáveis s_t

  Observando que $\sum_{t=1}^{n} (s_t - s_{t-1}) = \sum_{t=1}^{n} (x_t - D_t)$ é uma série telescópica, temos que

$$s_t = \sum_{i=1}^{t} x_i - \sum_{i=1}^{t} D_i$$

  Substituindo na função de custo $\sum_{t=1}^{n} p_t' x_t + \sum_{t=1}^{n} h_t s_t$, obtemos

$$\sum_{t=1}^{n} p_t' x_t + \sum_{t=1}^{n} h_t (\sum_{i=1}^{t} x_i - \sum_{i=1}^{t} D_i) =$$

$$\sum_{t=1}^{n} p_t' x_t + \sum_{t=1}^{n} h_t \sum_{i=1}^{t} x_i - \sum_{t=1}^{n} h_t \sum_{i=1}^{t} D_i =$$

$$\sum_{t=1}^{n}(p'_t + \sum_{i=t}^{n} h_i)x_t - \sum_{t=1}^{n} h_t \sum_{i=1}^{t} D_i$$

Nomeando $p_t = p'_t + \sum_{i=t}^{n} h_i$, e observando que $-\sum_{t=1}^{n} h_t \sum_{i=1}^{t} D_i$ é constante, o problema pode ser reformulado do seguinte modo:

$$\min \sum_{t=1}^{n}(p_t x_t + f_t y_t) \tag{1.20}$$

sujeito a:

$$\sum_{t=1}^{n} x_t = D_{1,n} \tag{1.21}$$

$$\sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n \tag{1.22}$$

$$x_t \leq y_t D_{1,n}, \text{ para } t = 1, \ldots, n \tag{1.23}$$

$$x_t \geq 0, \text{ para } t = 1, \ldots, n \tag{1.24}$$

$$x_t \in \mathbb{R}, y_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{1.25}$$

A formulação acima será usada nos próximos capítulos e será denominada *problema irrestrito* (em inglês *Uncapacitated*), pois desconsideramos a existência de qualquer restrição na capacidade de produção ou de armazenamento. De agora em diante, vamos nos referir a esse problema como *LS-I*.

1.1.3 Adicionando restrições ao problema

O problema, da forma como foi descrito até aqui, não leva em conta restrições de ordem prática com os quais os responsáveis pelo planejamento de produção têm que lidar, tais como restrições na capacidade de armazenamento, por exemplo, por limitação de espaço físico e restrições na quantidade

que pode ser produzida (em inglês, normalmente referenciado como *Capacitated*), em um determinado período de tempo, por exemplo, por limitação das máquinas e/ou das pessoas envolvidas, que são inerentes ao processo.

Para que nosso modelo seja consistente com problemas reais encontrados no planejamento de produção, temos que considerar as restrições acima.

Nesta seção, vamos tomar a liberdade de denotarmos o p'_t definido anteriormente por p_t.

Vamos agora considerar restrições de armazenamento na modelagem. Chamemos de u_t a quantidade máxima que pode ser armazenada ao final de um período de tempo t. A restrição que nos diz que a quantidade armazenada em cada instante deve respeitar a capacidade máxima de armazenamento segue de maneira trivial e é apresentada abaixo:

$$s_t \leq u_t, \text{ para } t = 1, \ldots, n. \tag{1.26}$$

Vale observar que essa restrição pode ser trivialmente incorporada à representação do problema como fluxo de custo mínimo. A modelagem do problema como um *MIP*, considerando as restrições

de armazenamento, fica

$$\min \sum_{t=1}^{n} (p_t x_t + h_t s_t + f_t y_t) \tag{1.27}$$

sujeito a:

$$s_{t-1} + x_t = s_t + D_t, \text{ para } t = 1, \ldots, n \tag{1.28}$$

$$\sum_{t=1}^{n} x_t = D_{1,n} \tag{1.29}$$

$$\sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n \tag{1.30}$$

$$x_t \leq y_t D_{1,n}, \text{ para } t = 1, \ldots, n \tag{1.31}$$

$$s_t \leq u_t, \text{ para } t = 1, \ldots, n \tag{1.32}$$

$$x_t \geq 0, s_t \geq 0, \text{ para } t = 1, \ldots, n \tag{1.33}$$

$$s_0 = s_n = 0 \tag{1.34}$$

$$x_t \in \mathbb{R}, s_t \in \mathbb{R}, y_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{1.35}$$

Chamaremos essa modelagem, daqui por diante, de *LS-A*.

Vamos agora proceder de maneira análoga com as restrições de capacidade. Chamemos de c_t a quantidade máxima que pode ser produzida em um período de tempo t. A restrição que nos diz que a quantidade produzida em cada instante deve respeitar a quantidade máxima que pode ser produzida é a seguinte:

$$x_t \leq c_t, \text{ para } t = 1, \ldots, n. \tag{1.36}$$

Novamente, aqui também podemos incorporar essa restrição à representação do problema como um fluxo de custo mínimo. A modelagem do problema como um *MIP*, considerando as restrições na capacidade produtiva:

$$\min \sum_{t=1}^{n} (p_t x_t + h_t s_t + f_t y_t) \tag{1.37}$$

sujeito a:

$$s_{t-1} + x_t = s_t + D_t, \text{ para } t = 1, \ldots, n \tag{1.38}$$

$$\sum_{t=1}^{n} x_t = D_{1,n} \tag{1.39}$$

$$\sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n \tag{1.40}$$

$$x_t \leq c_t y_t, \text{ para } t = 1, \ldots, n \tag{1.41}$$

$$x_t \geq 0, s_t \geq 0, \text{ para } t = 1, \ldots, n \tag{1.42}$$

$$s_0 = s_n = 0 \tag{1.43}$$

$$x_t \in \mathbb{R}, s_t \in \mathbb{R}, y_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{1.44}$$

Vamos nos referir, daqui por diante, a essa modelagem como *LS-C*.

Obviamente, podemos combinar ambas as restrições no nosso modelo, como a seguir:

$$\min \sum_{t=1}^{n} (p_t x_t + h_t s_t + f_t y_t) \tag{1.45}$$

sujeito a:

$$s_{t-1} + x_t = s_t + D_t, \text{ para } t = 1, \ldots, n \tag{1.46}$$

$$\sum_{t=1}^{n} x_t = D_{1,n} \tag{1.47}$$

$$\sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n \tag{1.48}$$

$$x_t \leq c_t y_t, \text{ para } t = 1, \ldots, n \tag{1.49}$$

$$s_t \leq u_t, \text{ para } t = 1, \ldots, n \tag{1.50}$$

$$x_t \geq 0, s_t \geq 0, \text{ para } t = 1, \ldots, n \tag{1.51}$$

$$s_0 = s_n = 0 \tag{1.52}$$

$$x_t \in \mathbb{R}, s_t \in \mathbb{R}, y_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{1.53}$$

A essa modelagem, vamos nos referir por *LS-AC*.

1.2 Resumo do capítulo

Neste capítulo apresentamos os problemas básicos de dimensionamento de lotes, suas respectivas formulações como *programas inteiro-mistos* e a interpretação como um problema de fluxo víavel de custo mínimo em uma rede.

Na tabela 1.2 mostramos um resumo dos problemas que serão tratados ao longo do trabalho e suas respectivas restrições.

Nos próximos capítulos analisaremos a estrutura de uma solução ótima, a complexidade e mostraremos algoritmos para a resolução dos problemas.

Problema	Restrição de Armazenamento	Restrição de Produção
LS-I	não	não
LS-A	sim	não
LS-C	não	sim
LS-AC	sim	sim

Tabela 1.1: Restrições dos problemas tratados

Capítulo 2

Problemas básicos

O objetivo desse capítulo é apresentar alguns dos resultados fundamentais no tratamento do problema e algoritmos obtidos a partir desses resultados.

Os subproblemas a serem tratados são:

1. O problema irrestrito (LS-I);
2. O problema com restrições no armazenamento (LS-A);
3. O problema com restrições na produção (LS-C).
4. O problema com restrições na produção e no armazenamento (LS-AC).

2.1 O Problemas irrestrito (LS-I)

Nesta seção, vamos tratar do problema no caso em que não existe limitação na quantidade que pode ser produzida (*Uncapacitated*) ou armazenada e a única restrição existente é que as demandas devem ser atendidas.

Um dos primeiros artigos a tratar do problema de dimensionamento de lotes se deve a *Wagner e Within* [WW58], de 1958, onde foi mostrado um algoritmo $O(n^2)$ para resolver o problema no caso específico em que os custos de produção são constantes em todos os períodos do horizonte de planejamento. Nos anos seguintes, diversos trabalhos apresentaram melhoras no algoritmo proposto por eles, porém, nenhuma delas havia apresentado uma melhora significativa na complexidade do algoritmo.

Um dos primeiros artigos que apresentou uma melhora significativa na complexidade do problema de dimensionamento de lotes se deve a *van Hoesel et al* [KHW92] e [Hoe91], de 1991. Os autores fizeram uma interpretação geométrica da recorrência que aparece no problema, o que permitiu uma redução na complexidade do algoritmo.

Vamos agora apresentar o algoritmo proposto em [KHW92].

2.1.1 Algoritmos para *LS-I*

Os principais algoritmos desenvolvidos para tratar esse problema são baseados em recorrências. Essas recorrências se fundamentam em uma propriedade que foi mostrada primeiramente em [WW58] para o caso em que os custos são não crescentes, isto é, $p_{t+1} \leq p_t$ para $t = 1, \ldots, n-1$, e posteriormente complementada por *Veinot* [Jr64] e *Zangwill* [Zan68] para o caso geral.

A seguir mostraremos um lema que será utilizado na demostração da propriedade que acabamos de citar. O lema utiliza a seguinte definição:

Definição 2.1. Consideremos uma rede $G = (V, A)$ e vértices $v_1, v_2, \ldots, v_n \in V$. Denotaremos a sequência de vértices $C = v_1, v_2, \ldots, v_{n-1}, v_n, v_1$ por *pseudo-ciclo* se (v_i, v_{i+1}) ou (v_{i+1}, v_i) pertencem a A e tem fluxo positivo, para todo i da forma $i = (t \mod n) + 1$ em que t assume os valores $t = 1, \ldots, n$. Ainda diremos que $(v_i, v_{i+1}) \in C$ se v_i, v_{i+1} é uma subsequencia de C.

Lema 2.1. No problema de obtenção de um fluxo viável de custo mínimo, se o custo dos arcos é função côncava do fluxo então existe uma solução ótima em que não existe nenhum *pseudo-ciclo*.

Demonstração. Consideremos uma rede $G = (V, A)$ e um fluxo $\chi(i, j)$ para todo $(i, j) \in A$ que seja um fluxo viável. Suponha que exista um *pseudo-ciclo* $C = v_1, v_2, \ldots, v_{n-1}, v_n, v_1$ na rede G. Denotemos por $C_d = \{(v_i, v_j) \in A \bigcap C\}$ e por $C_i = \{(v_i, v_j) \in A \text{ e } (v_j, v_i) \in C\}$. Observamos que $C = C_d \bigcup C_i$ e chamaremos todo $(i, j) \in C_d$ de arco direto e todo $(i, j) \in C_i$ de arco inverso. O custo de transportar $\chi(i, j)$ unidades de fluxo em um arco (i, j) é determinado por uma função côncava $c_{i,j}(\chi(i, j))$ e, portanto, o custo do *pseudo-ciclo*, que denotaremos por $c(C)$ é

$$c(C) = \sum_{(i,j)\in C_d} c_{i,j}(\chi(i, j)) + \sum_{(i,j)\in C_i} c_{i,j}(\chi(i, j)). \tag{2.1}$$

Denotemos por ϵ_d o menor volume de fluxo nos arcos diretos, e por ϵ_i o menor volume de fluxo nos arcos inversos, isto é,

$$\epsilon_d = \min\{\chi(i, j) : (i, j) \in C_d\}, \tag{2.2}$$

$$\epsilon_i = \min\{\chi(i, j) : (i, j) \in C_i\}. \tag{2.3}$$

Consideremos duas possibilidades. Uma em que aumentamos em ϵ_i unidades o fluxo no *pseudo-ciclo* no sentido direto e outra em que aumentamos em ϵ_d unidades o fluxo no sentido oposto. Observamos que em ambos os casos, o *pseudo-ciclo* é eliminado. Tomemos um $\epsilon \leq \min\{\epsilon_d, \epsilon_i\}$. Denotaremos o fluxo dos arcos pertencentes ao *pseudo-ciclo* após o acréscimo de ϵ unidades no sentido direto por C_+ e por C_- o fluxo após o acréscimo de ϵ no sentido inverso, com os respectivos

custos:

$$c(C_+) = \sum_{(i,j)\in C_d} c_{i,j}(\chi(i,j)+\epsilon) + \sum_{(i,j)\in C_i} c_{i,j}(\chi(i,j)-\epsilon) \tag{2.4}$$

$$c(C_-) = \sum_{(i,j)\in C_d} c_{i,j}(\chi(i,j)-\epsilon) + \sum_{(i,j)\in C_i} c_{i,j}(\chi(i,j)+\epsilon) \tag{2.5}$$

Consideremos as seguintes diferenças

$$\text{(i)}\ \ c(C_+) - c(C) = \sum_{(i,j)\in C_d} [c_{i,j}(\chi(i,j)+\epsilon) - c_{i,j}(\chi(i,j))] + \sum_{(i,j)\in C_i} [c_{i,j}(\chi(i,j)-\epsilon) - c_{i,j}(\chi(i,j))]$$

$$\text{(ii)}\ \ c(C_-) - c(C) = \sum_{(i,j)\in C_d} [c_{i,j}(\chi(i,j)-\epsilon) - c_{i,j}(\chi(i,j))] + \sum_{(i,j)\in C_i} [c_{i,j}(\chi(i,j)+\epsilon) - c_{i,j}(\chi(i,j))]$$

e suponhamos que $c(C_+) \geq c(C)$ e, portanto,

$$\sum_{(i,j)\in C_d} [c_{i,j}(\chi(i,j)+\epsilon) - c_{i,j}(\chi(i,j))] \geq \sum_{(i,j)\in C_i} [c_{i,j}(\chi(i,j)) - c_{i,j}(\chi(i,j)-\epsilon)]. \tag{2.6}$$

Como a função de custo nos arcos é côncava, temos que

$$2c_{i,j}(\chi(i,j)) \geq c_{i,j}(\chi(i,j)+\epsilon) + c_{i,j}(\chi(i,j)-\epsilon), \quad \forall (i,j) \in A \tag{2.7}$$

e portanto, usando esse fato e (2.6), substituindo em (ii), temos que

$$c(C_-) - c(C) \leq \sum_{(i,j)\in C_d} c_{i,j}(\chi(i,j)-\epsilon) + c_{i,j}(\chi(i,j)+\epsilon) - 2c_{i,j}(\chi(i,j)) \leq 0. \tag{2.8}$$

Agora suponhamos que $c(C_-) \geq c(C)$

$$\sum_{(i,j)\in C_i} [c_{i,j}(\chi(i,j)+\epsilon) - c_{i,j}(\chi(i,j))] \geq \sum_{(i,j)\in C_d} [c_{i,j}(\chi(i,j)) - c_{i,j}(\chi(i,j)-\epsilon)]. \tag{2.9}$$

Aplicando (2.7) e (2.9) em (i) temos que

$$c(C_+) - c(C) \leq \sum_{(i,j)\in C_i} c_{i,j}(\chi(i,j) - \epsilon) + c_{i,j}(\chi(i,j) + \epsilon) - 2c_{i,j}(\chi(i,j)) \leq 0. \tag{2.10}$$

Logo, concluímos que sempre existirá uma forma de aumentar o fluxo do *pseudo-ciclo* em um determinado sentido de maneira que o custo não aumente e, portanto, podemos escolher ϵ_d se for o sentido inverso e ϵ_i se for o sentido direto, de forma que obtemos uma solução na qual o custo não aumenta e que não contém um *pseudo-ciclo*. □

Abaixo segue a proposição com a seguinte propriedade conforme mostrado em [Zan68].

Proposição 2.1. Existe uma solução ótima para o problema de planejamento de lotes irrestrito onde para $t = 1, \ldots, n$ a seguinte igualdade vale: $s_{t-1}x_t = 0$. Tal propriedade é chamada, por vezes, de *princípio de estoque zero*, por indicar que só será produzido novamente quando tudo o que estiver armazenado for utilizado para suprir a demanda.

Demonstração. Conforme discutimos no capítulo anterior, o problema pode ser interpretado como um problema de fluxo de custo mínimo. Pelo lema (2.1) sabemos que existe uma solução ótima que não contém um *pseudo-ciclo*. Mostraremos a seguir que, nesse caso, a condição $s_{t-1}x_t = 0$ vale.

Suponhamos por contradição que existe uma solução ótima sem *pseudo-ciclos* em que temos um período t no qual x_t é maior do que zero e s_{t-1} também é maior do que zero. Como s_{t-1} é maior que zero, existe um período i em que x_i é maior que zero que origina o fluxo s_{t-1}. Tomemos $i = \max\{j : 1 \leq j < t \text{ e } x_j > 0\}$. Como i é máximo, temos que $s_i, s_{i+1}, \ldots, s_{t-1}$ são maiores do que zero e, portanto, $C = o, i, i+1, \ldots, t-1, t, o$ é um *pseudo-ciclo*, nos levando a uma contradição. □

Vamos agora apresentar algumas definições que serão úteis na construção do algoritmo.

Definição 2.2. Chamaremos de *ponto de regeneração* um período de tempo t em que temos $s_t = 0$.

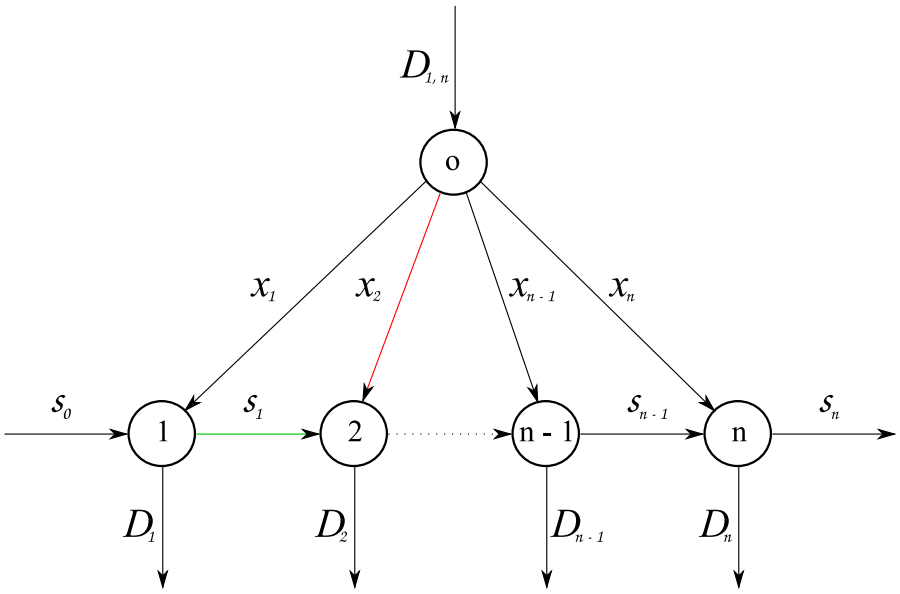


Figura 2.1: Princípio do estoque zero: $s_{t-1}x_t = 0$, para $t = 1, \ldots, n$. Para exemplificar, ou a aresta s_1 ou a aresta x_2 será igual a zero.

Definição 2.3. Chamaremos de *intervalo de regeneração* um intervalo de tempo $[t, l]$ onde, $t-1$ e l são *pontos de regeneração* consecutivos. Isto é, $s_r > 0$, para $t \leq r < l$.

Com base na última definição, podemos formular o seguinte:

Proposição 2.2. princípio de decomposição de estoque: um horizonte de planejamento pode ser quebrado em sucessivos intervalos de regeneração.

Demonstração. De fato, como o valor do estoque inicialmente é zero e no final do período n também é zero, caso não haja nenhum período i no horizonte de planejamento, tal que $1 \leq i < n$ onde $s_i = 0$, o intervalo $[1, n]$ é um *intervalo de regeneração.* Se houver um conjunto de períodos de tempo $I = \{i_a, i_{a+1}, \ldots, i_b\}$ tais que, no planejamento de produção nós tivermos que $s_{i_j} = 0$ para $j = a, \ldots, b$, podemos considerar o planejamento de produção de 1 até n como a sequência de *intervalos de regeneração* $[1, i_a], [i_a + 1, i_{a+1}], \ldots, [i_b, n]$. □

Aqui vale observar que as duas últimas proposições nos dizem que existe um planejamento ótimo que pode ser quebrado em intervalos de regeneração, e que dentro de cada intervalo de regeneração se produz apenas uma única vez.

Com base nisso, vamos agora descrever duas recorrências que utilizam funções $H(.)$ e $G(.)$, mostradas a seguir, que descrevem a solução do problema de maneira direta e inversa, respectivamente. Isto é, o custo ótimo de produção é dado por $H(n)$ e $G(0)$.

Consideremos a função $H(i)$ que nos dá o custo ótimo de produção do período 1 até o período i. Vamos considerar $H(0) = 0$.

A recorrência que descreve a solução do problema é:

$$H(i) = \begin{cases} 0 & , \text{ se } i = 0 \\ \min\limits_{1 \leq t \leq i} f_t + p_t D_{t,i} + H(t-1) & , \text{ se } i > 0 \end{cases} \tag{2.11}$$

A seguir apresentamos um algoritmo que resolve o problema em $O(n^2)$.

Algoritmo *LS-I-DIRETO*$(p,\ f,\ D)$

01 $H[0] \leftarrow 0$

02 $D_{n+1,n} \leftarrow 0$

03 **para** $i = n, \ldots, 1$ **faça:**

04 $\quad D_{i,n} \leftarrow D_{i+1,n} + D_i$

05 **para** $i = 1, \ldots, n$ **faça:**

06 $\quad x_i \leftarrow 0$

07 $\quad minH \leftarrow \infty$

08 $\quad$ **para** $t = 1, \ldots, i$ **faça:**

09 $\quad\quad$ **se** $minH > f_t + p_t(D_{t,n} - D_{i+1,n}) + H[t-1]$ **então**

10 $\quad\quad\quad minH \leftarrow f_t + p_t(D_{t,n} - D_{i+1,n}) + H[t-1]$

11 $\quad\quad\quad minT[i] \leftarrow t$

12 $i \leftarrow n$

13 **enquanto** $i \geq 1$ **faça:**

14 $\quad$ **se** $H[i] \neq H[i-1]$ **então**

15 $\quad\quad x_{minT[i]} \leftarrow D_{minT[i],n} - D_{i+1,n}$

16 $\quad\quad i \leftarrow minT[i] - 1$

17 devolva x e $H[n]$

Notemos aqui, que a recorrência $H(n)$ pode ser interpretada como um problema de caminho mais curto, em um grafo com vértices $V = \{0, \ldots, n\}$ e arcos $A = \{(i, j)$, para todos i e $j \in \{0, \ldots n\}$ tais que $i < j\}$. Cada arco (i, j) tem um custo $c(i, j)$ igual ao custo do intervalo de regeneração $[i + 1, j]$, que é $f_{i+1} + p_{i+1}D_{i+1,j}$. Observemos que $|A| = O(n^2)$. Nesse grafo dirigido, um caminho mais curto de 0 até n é a sequência de intervalos de regeneração que representam o planejamento de produção de menor custo.

A seguir, a figura 2.2 ilustra o grafo que acabamos de descrever.

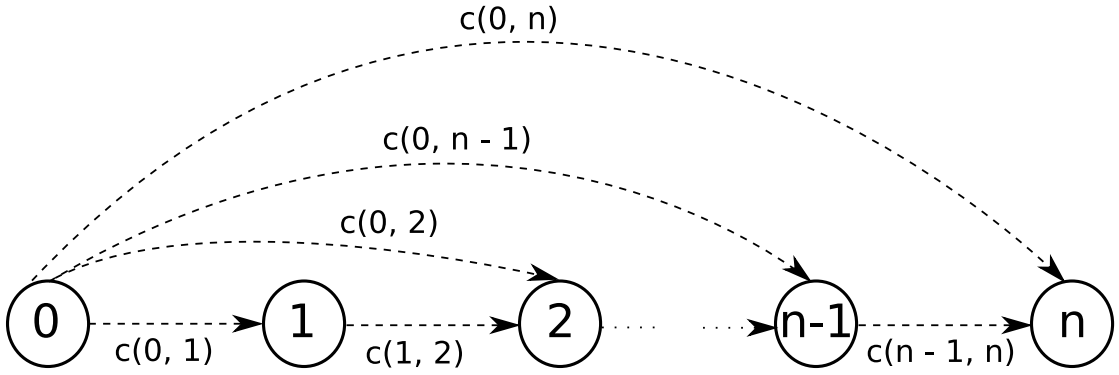


Figura 2.2: Representação do *LS-I* como um problema de caminho mais curto

Agora consideremos uma função $G(i)$ que nos dá o custo ótimo de produção a partir do período i até o período n. Além disso, vamos assumir por conveniência que $G(n+1) = 0$.

Uma recorrência alternativa para o cálculo do custo ótimo do problema *LS-I*, utilizando a função $G(.)$:

$$G(i) = \begin{cases} 0 & , \text{ se } i = n+1 \\ \min\limits_{i<t\leq n+1} f_i + p_i D_{i,t-1} + G(t) & , \text{ se } D_i > 0 \\ \min\{G(i+1), \min\limits_{i<t\leq n+1} f_i + p_i D_{i,t-1} + G(t)\} & , \text{ se } D_i = 0 \end{cases} \tag{2.12}$$

Uma solução para esse problema através de um algoritmo de complexidade $O(n^2)$ segue de maneira bastante natural da recorrência acima, que mostraremos a seguir:

Algoritmo *LS-I-INVERSO*(p, f, D)

01 $G[n+1] \leftarrow 0$

02 $D_{n+1,n} \leftarrow 0$

03 **para** $i = n, \ldots, 1$ **faça:**

04 $\quad x_i \leftarrow 0$

05 $\quad D_{i,n} \leftarrow D_{i+1,n} + D_i$

06 $\quad minG \leftarrow \infty$

07 $\quad$ **para** $t = i, \ldots, n$ **faça:**

08 $\quad\quad$ **se** $minG > f_i + p_i(D_{i,n} - D_{t,n}) + G[t]$ **então**

09 $\quad\quad\quad minG \leftarrow f_i + p_i(D_{i,n} - D_{t,n}) + G[t]$

10 $\quad\quad\quad minT[i] \leftarrow t$

11 $\quad$ **se** $D_i > 0$ **ou** $f_i + p_i(D_{i,n} - D_{minT[i],n}) + G[minT[i]] < G[i+1]$ **então**

12 $\quad\quad G[i] \leftarrow minG$

13 $\quad$ **senão**

14 $\quad\quad G[i] \leftarrow G[i+1]$

15 $i \leftarrow 1$

16 **enquanto** $i \leq n$ **faça:**

17 $\quad$ **se** $G[i] \neq G[i+1]$ **então**

18 $\quad\quad x_i \leftarrow D_{i,n} - D_{minT[i],n}$

19 $\quad\quad i \leftarrow minT[i]$

20 devolva x e $G[1]$

O que será mostrado a seguir é que é possível resolver o subproblema

$$\min_{i<t\leq n+1} p_i D_{i,t-1} + G(t) \tag{2.13}$$

em $O(log(n))$ e, portanto, o procedimento de cálculo de $G(i)$ entre as linhas *07* e *14* que custa $O(n)$ pode ter seu custo reduzido e o algoritmo *LS-I-INVERSO* pode ser reimplementado com custo $O(nlog(n))$.

Primeiramente vamos assumir que $D_{n+1,n} = 0$, por conveniência. Consideremos agora o conjunto de pontos $P = \{(D_{t,n}, G(t)),$ para $t = i+1, \ldots, n+1\}$ e o fecho convexo de P, C_P. Vamos definir a função $g(z) = w$, tal que $z \in [D_{n+1,n}, D_{i+1,n}]$ e $(z, w) \in C_P$.

Chamaremos $z \in (D_{n+1,n}, D_{i+1,n})$ de *breakpoint* se a inclinação de g muda. Além desses, consideraremos $D_{n+1,n}$ e $D_{i+1,n}$ como sendo *breakpoints*.

Suponha que temos r períodos de tempo t onde $D_{t,n}$ são *breakpoints*. Vamos considerar a função $e(p)$, $p = 1, \ldots, r$ e $i+1 = e(1) < e(2) < \ldots < e(r) = n+1$ que nos dá o período de tempo do associado ao p-ésimo *breakpoint*.

A partir de agora, vamos denotar o conjunto de pontos $e(p)$, para $p = 1, \ldots, r$ de períodos *eficientes*, pois mostraremos a seguir que na solução do subproblema, precisamos considerar apenas

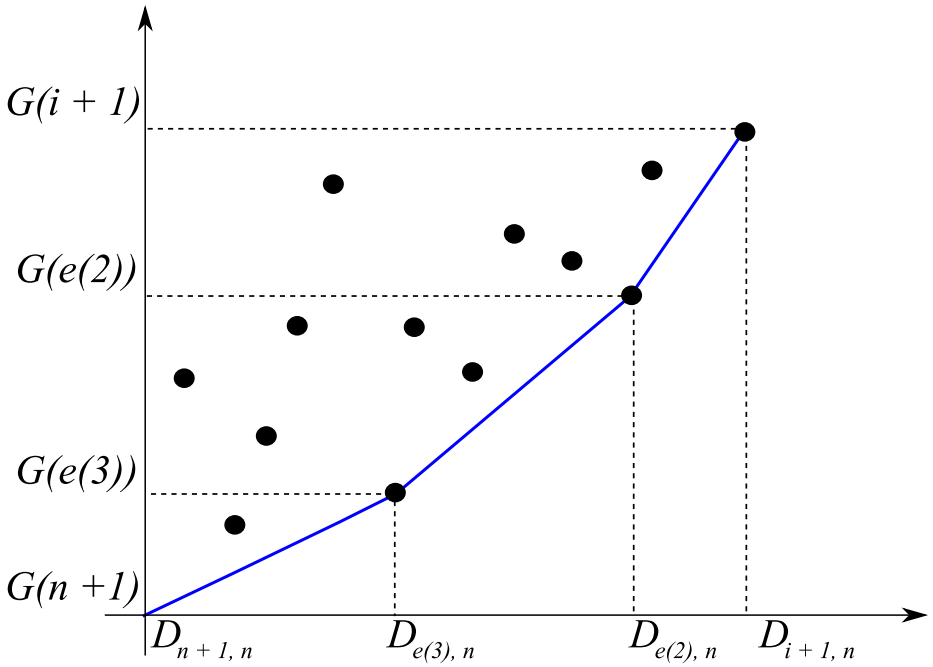


Figura 2.3: Figura ilustrando a função $g(.)$ e os respectivos breakpoints.

esses períodos de tempo.

Proposição 2.3. Sejam $e(p)$, $p = 1, \ldots, r$ os períodos eficientes, então temos que

$$\min_{i<t\leq n+1} p_i D_{i,t-1} + G(t) = \min_{i<p\leq r} p_i D_{i,e(p)} + G(e(p)).$$

Demonstração. Seja um período k, $i+1 < k < n+1$ que não é um período eficiente e considere dois períodos eficientes consecutivos j e l tais que $j < k < l$.

Consideremos a função $g(z)$ definida no intervalo $[D_{l,n}, D_{j,n}]$:

$$g(D_{k,n}) = G(l) + \frac{G(j) - G(l)}{D_{j,l-1}} D_{k,l-1}, \tag{2.14}$$

e observando que, pela definição de g, temos

$$g(D_{k,n}) \leq G(k). \tag{2.15}$$

Suponhamos que

$$p_i \geq \frac{G(j) - G(l)}{D_{j,l-1}}. \tag{2.16}$$

Por (2.15) temos que $\quad p_i D_{i,k-1} + G(k) \geq p_i D_{i,j-1} + p_i D_{j,k-1} + g(D_{k,n})$

e aplicando (2.14) e (2.16) temos $\quad p_i D_{i,k-1} + G(k) \geq p_i D_{i,j-1} + G(j).$

Suponhamos agora o contrário, que

$$p_i < \frac{G(j) - G(l)}{D_{j,l-1}}. \tag{2.17}$$

Por (2.15) temos que $\quad p_i D_{i,k-1} + G(k) \geq p_i D_{i,l-1} - p_i D_{k,l-1} + g(D_{k,n})$

e aplicando (2.14) e (2.17) temos $\quad p_i D_{i,k-1} + G(k) \geq p_i D_{i,l-1} + G(l).$

Com isso, mostramos que podemos desconsiderar os períodos entre dois períodos eficientes consecutivos e portanto, podemos desconsiderar todos os períodos que não são períodos eficientes. □

A próxima proposição que apresentaremos a seguir mostra que podemos utilizar a inclinação de $g(.)$ entre dois períodos eficientes consecutivos para comparar dois valores distintos em (2.13).

Proposição 2.4. Sejam j e l dois períodos eficientes consecutivos. Se

$$\frac{G(j) - G(l)}{D_{j,l-1}} < p_i,$$

então $p_i D_{i,j-1} + G(j) < p_i D_{i,l-1} + G(l)$, senão se

$$\frac{G(j) - G(l)}{D_{j,l-1}} \geq p_i,$$

temos que $p_i D_{i,j-1} + G(j) \geq p_i D_{i,l-1} + G(l)$

Demonstração.

$$\begin{aligned} \frac{G(j) - G(l)}{D_{j,l-1}} < p_i \quad & \iff \\ G(j) < p_i D_{j,l-1} + G(l), \quad & \text{somando } p_i D_{i,j-1} \text{ de ambos os lados, temos} \\ G(j) + p_i D_{i,j-1} < p_i D_{i,j-1} + p_i D_{j,l-1} + G(l) \quad & \iff \\ G(j) + p_i D_{i,j-1} < p_i D_{i,l-1} + G(l). & \end{aligned}$$

Considerando agora o caso contrário:

$$\begin{aligned} \frac{G(j) - G(l)}{D_{j,l-1}} \geq p_i \quad & \iff \\ G(j) \geq p_i D_{j,l-1} + G(l), \quad & \text{somando } p_i D_{i,j-1} \text{ de ambos os lados, temos} \\ G(j) + p_i D_{i,j-1} \geq p_i D_{i,j-1} + p_i D_{j,l-1} + G(l) \quad & \iff \\ G(j) + p_i D_{i,j-1} \geq p_i D_{i,l-1} + G(l). & \end{aligned}$$

□

Vamos agora retomar o subproblema que, pela proposição 2.3, é equivalente a

$$\min_{i<t\leq n+1} p_i D_{i,t-1} + G(t) = \min_{i<p\leq r} p_i D_{i,e(p)} + G(e(p)). \tag{2.18}$$

Usando a proposição 2.4 que acabamos de mostrar e o fato da função g ser convexa, temos que o período eficiente $e(q)$ que minimiza (2.18) é:

$$q = \min\{r, \min\{p : 1 \leq p < r \text{ e } \frac{G(e(p)) - G(e(p+1))}{D_{e(p),e(p+1)-1}} < p_i\} \tag{2.19}$$

pois, para $p = 1, \ldots, q-1$ temos $p_i D_{i,e(p)-1} + G(e(p)) \geq p_i D_{i,e(p+1)-1} + G(e(p+1))$ e para $p = q, \ldots, r$ temos que $p_i D_{i,e(p)-1} + G(e(p)) < p_i D_{i,e(p+1)-1} + G(e(p+1))$.

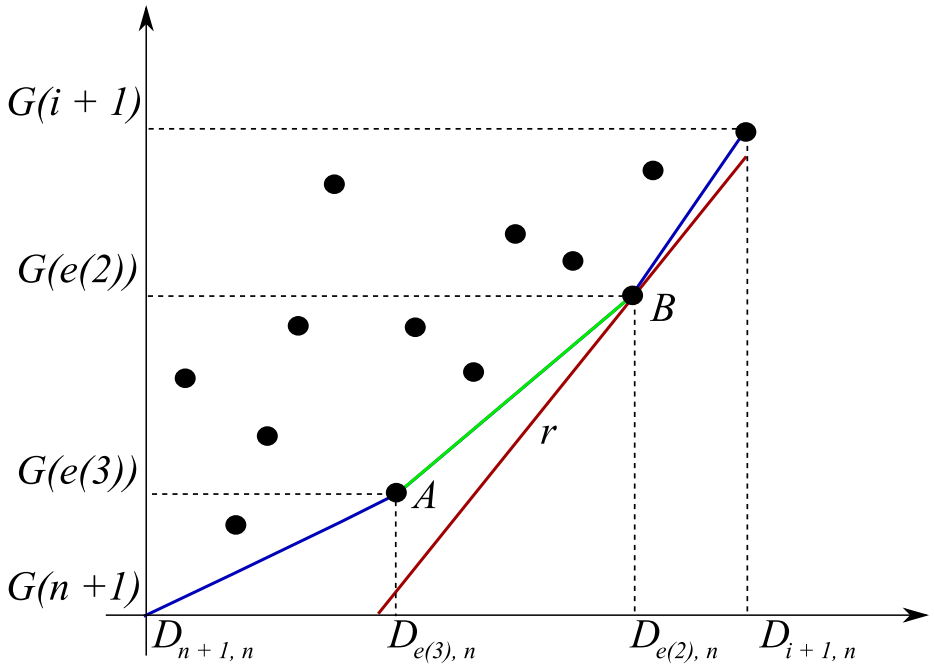


Figura 2.4: Neste exemplo, o período de tempo que minimiza (2.18) é $q = e(2)$. A reta r tem coeficiente angular p_i maior que o coeficiente angular do segmento $\overline{AB}$.

Dada a forma da solução de q, podemos calculá-la através de uma busca binária, em tempo $O(log(n))$, se tivermos os breakpoints armazenados de maneira ordenada.

Como conseguimos resolver o subproblema (2.18), conseguimos calcular o valor de $G(i)$. Para conseguirmos garantir que o algoritmo possa ser resolvido em tempo $O(nlog(n))$, precisamos conseguir atualizar o fecho convexo de maneira eficiente, ao adicionarmos o ponto $Q = (D_{i,n}, G(i))$. Para isso, procederemos de maneira análoga à do algoritmo de *Graham* para calcular o fecho convexo (uma apresentação bastante didática do algoritmo pode ser encontrada em [CLRS]).

Consideremos os pontos P_k, $k = 1, \ldots, K = |C_{P_K}|$, onde $P_k \in C_{P_K}$, ordenados da esquerda para direita. Tomemos agora o conjunto $C_{P_K} \bigcup \{Q\}$ e denotaremos o seu respectivo fecho convexo de $C_{P_K \bigcup Q}$. Se, por acaso tivermos $C_{P_K \bigcup \{Q\}} = C_{P_K} \bigcup \{Q\}$, então, trivialmente, já conseguimos atualizar o fecho convexo.

Suponhamos agora o contrário, que $C_{P_K \bigcup \{Q\}} \neq C_{P_K} \bigcup \{Q\}$. Isso nos diz que os vetores $\vec{u} = \overrightarrow{P_{K-1}P_K}$ e $\vec{v} = \overrightarrow{P_K Q}$ estão em sentido horário, isto é, formam um trecho côncavo e, portanto, o ponto P_K não pertence ao fecho convexo .

Uma maneira fácil de verificar se eles formam um trecho côncavo é através da orientação do vetor resultante do produto vetorial $\vec{u} \times \vec{v}$. No caso, basta verificarmos se

$$det \begin{vmatrix} \overrightarrow{u_x} & \overrightarrow{u_y} \\ \overrightarrow{v_x} & \overrightarrow{v_y} \end{vmatrix} \text{ é } < 0.$$

A partir daí, basta procedermos da mesma maneira com $C_{P_{K-1}} \bigcup \{Q\}$ e assim, sucessivamente, até encontrarmos um $w, 1 < w \leq K$ onde $C_{P_w \bigcup \{Q\}} = C_{P_w} \bigcup \{Q\}$.

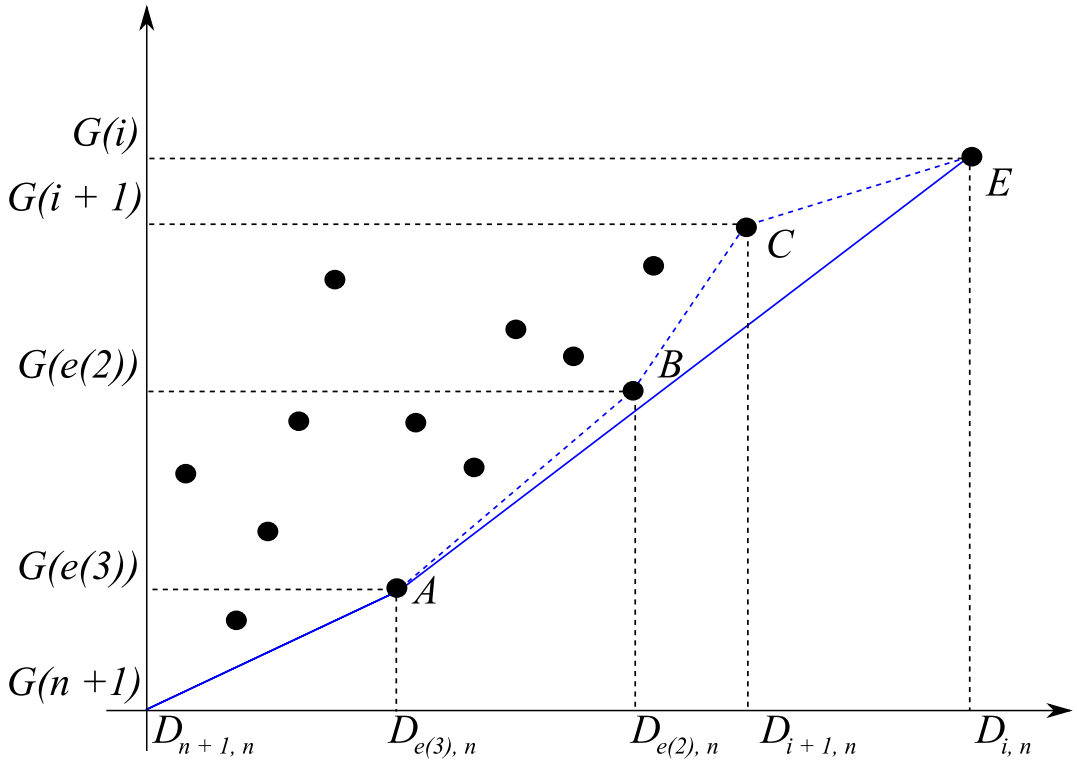


Figura 2.5: Aqui, com a adição do ponto E, os pontos B e C deixam de fazer parte do fecho convexo.

Com o que foi dito até aqui, vamos descrever o algoritmo que resolve o problema *LS-I* no caso geral em $O(nlog(n))$.

Para isso utilizaremos um vetor L e sobre ele definiremos as seguintes operações de pilha, que podem ser implementadas em $O(1)$:

- $empilha(L, i)$ - Empilha o elemento i no final de L;
- $desempilha(L)$ - Remove o último elemento de L;
- $topo(L)$ - Devolve o elemento no final da pilha;
- $prox(L, i)$ - Devolve o elemento que se encontra imediatamente acima do elemento i na pilha L;
- $ant(L, i)$ - Devolve o elemento que se encontra imediatamente abaixo do elemento i na pilha L.

Algoritmo *LS-I-HKW*$(p,\ f,\ D)$

01 $empilha(L, n+1)$

02 $G[n+1] \leftarrow 0$

03 $D_{n+1,n} \leftarrow 0$

04 $q(n+1) \leftarrow 0$

05 **para** $i = n, \ldots, 1$ **faça:**

06 $\quad D_{i,n} \leftarrow D_{i+1,n} + D_i$

07 **para** $i = n, \ldots, 1$ **faça:**

08 $\quad empilha(L, i)$

09 $\quad q(i) \leftarrow \min\{n+1, \min\{t \in L : t < n+1 \textbf{ e } \frac{G[t]-G[ant(L,t)]}{D_{t,n}-D_{ant(L,t),n}} < p_i\}\}$

10 $\quad$ **se** $D_i > 0$ **ou** $f_i + p_i(D_{1,n} - D_{q(i),n}) + G[q(i)] \leq G[i+1]$ **então**

11 $G[i] \leftarrow f_i + p_i(D_{1,n} - D_{q(i),n}) + G[q(i)]$

12 **senão**

13 $G[i] \leftarrow G[i+1]$

14 **se** $|L| \geq 3$ **então**

15 **faça:**

16 $desempilha(L)$

17 $t \leftarrow topo(L)$

18 $P_K \leftarrow (D_{t,n}, G[t])$

19 $P_{K-1} \leftarrow (D_{ant(L,t),n}, G[ant(L,t)])$

20 $Q \leftarrow (D_{i,n}, G[i])$

21 $\vec{u} \leftarrow \overrightarrow{P_{K-1}P_K}$

22 $\vec{v} \leftarrow \overrightarrow{P_KQ}$

23 **enquanto** $\vec{u} \times \vec{v} \leq 0$

24 $empilha(L, i)$

25 **enquanto** $i \leq n$ **faça:**

26 **se** $D_i = 0$ **e** $G[i] = G[i+1]$ **então**

27 $i \leftarrow i + 1$

28 **senão**

29 $x_i \leftarrow D_{i,n} - D_{q(i),n}$

30 $i \leftarrow q(i)$

31 devolva x e $G[1]$

Como L armazena os vértices de um fecho convexo, a linha *09* do algoritmo descrito acima

pode ser calculada através de uma busca binária e, portanto, consumindo um tempo proporcional a $O(log(n))$.

Contudo se por acaso tivéssemos uma implementação na qual o tempo amortizado de cada execução da linha *09* consumisse um tempo proporcional a $O(1)$ o algoritmo teria complexidade $O(n)$. O que mostraremos agora é que no caso originalmente tratado por Wagner e Within em [WW58], no qual temos que

$$p_1 \geq p_2 \geq \ldots \geq p_n, \tag{2.20}$$

o problema *LS-I* pode ser resolvido em $O(n)$. A condição descrita pela equação (2.20) é frequentemente chamada de condição de Wagner-Within.

Quando essa condição é satisfeita, podemos calcular a linha *09* a partir do ponto $q(i+1)$, pois como $p_{i+1} \leq p_i$, a desigualdade é válida para t maior ou igual a $q(i+1)$. Além disso, observamos que se, por acaso, o elemento $q(i+1)$ foi removido, o elemento que acabou de ser inserido será o elemento escolhido, dado que o coeficiente angular será menor.

Apresentaremos a seguir um algoritmo para o cálculo de $q(i)$.

Algoritmo *calculo-q*(i)

01 $t \leftarrow q(i+1)$

02 **se** $t \notin L$ **então**

03 $q(i) \leftarrow i$

04 **senão**

05 **enquanto** $\frac{G[t]-G[ant(L,t)]}{D_{t,n}-D_{ant(L,t),n}} < p_i$ **faça:**

06 $t \leftarrow prox(L,t)$

07 $q(i) \leftarrow t$

Como em cada iteração nunca olhamos para um período anterior a $q(i+1)$, o custo nas n iterações será $O(n)$ e, portanto, o custo amortizado por iteração será $O(1)$, o que nos dá nesse caso específico um algoritmo $O(n)$ para o problema *LS-I*.

2.2 O problema com restrições de capacidade de armazenamento (*LS-A*)

O problema irrestrito (*LS-I*), que discutimos na seção anterior, trata de um caso que, na maior parte das vezes, é bastante irreal, pois por questões de natureza física e orçamentárias, quase sempre estaremos lidando com algum tipo de restrição. Contudo, o problema contém aspectos estruturais bastante semelhantes ao de problemas mais complexos, como o *LS-A*.

2.2.1 A estrutura do problema

Para apresentarmos a solução do problema, vamos redefinir alguns conceitos que foram definidos na seção anterior, mas adequando-os com a necessidade de respeitar a restrição

$$s_t \leq u_t, \text{ para } t = 1, \ldots, n \tag{2.21}$$

Definição 2.4. Chamaremos de *ponto de regeneração* um período de tempo t no qual $s_t \in \{0, u_t\}$.

Definição 2.5. Chamaremos de *intervalo de regeneração* um intervalo de tempo $[t, l]$ onde, $t-1$ e l são *pontos de regeneração* consecutivos. Isto é, $0 < s_r < u_r$, para todo r tal que $t \leq r < l$.

Análogo ao que foi feito na seção anterior, faremos a seguinte proposição:

Proposição 2.5. princípio de decomposição de estoque: um horizonte de planejamento pode ser quebrado em sucessivos intervalos de regeneração.

Demonstração. De fato, como o valor do estoque inicialmente é zero e no final do período n também é zero, caso não haja nenhum período i no planejamento ótimo, tal que $1 \leq i \leq n$ onde $s_i \in \{0, u_i\}$, o intervalo $[1, n]$ é um intervalo de regeneração. Se houver um conjunto de períodos de tempo $I = \{i_a, i_{a+1}, \ldots, i_b\}$ tais que, no planejamento de produção nós tivermos que $s_{i_j} \in \{0, u_{i_j}\}$ para $j = a, \ldots, b$, podemos considerar o planejamento de produção de 1 até n como a sequência de intervalos de regeneração $[1, i_a], [i_a + 1, i_{a+1}], \ldots, [i_b, n]$. □

Vamos agora apresentar o resultado que caracteriza a solução ótima do problema:

Proposição 2.6. Existe uma solução ótima para o problema *LS-A*, se ele for viável, em que se produz, no máximo, apenas uma única vez dentro de um intervalo de regeneração.

Demonstração. Conforme vínhamos fazendo, vamos novamente interpretar o problema como um fluxo em uma rede, que é a mesma do problema anterior. Contudo, para facilitar a demonstração, vamos utilizar uma redução, de forma a transformar o problema com restrições de armazenamento, em um problema irrestrito equivalente (Conforme mostrado no capítulo 4 do livro [CCPS]).

Para isso, vamos eliminar os arcos $(i, i + 1)$ e adicionar vértices q_i e r_i com demandas u_i e $-u_i$, respectivamente. Ao conjunto de arcos, adicionaremos (i, q_i), (r_i, q_i) e $(r_i, i+1)$, pelos quais passarão $s_i, u_i - s_i$ e s_i unidades de fluxo, respectivamente. O custo por unidade de fluxo que passa no arco (i, q_i) é h_i e zero nos arcos (r_i, q_i) e $(r_i, i + 1)$.

Uma solução ótima nessa rede construída, também será uma solução ótima no problema original.

Tomemos um fluxo ótimo sem pseudo-ciclos no qual o intervalo $[j, l]$ é um intervalo de regeneração. Isso implica que para todo $t \in [j, l - 1]$, temos que $0 < s_i < u_i$, por definição e, portanto, o fluxo nos arcos (i, q_i), (r_i, q_i) e $(r_i, i + 1)$ é positivo.

Suponhamos, por contradição, que existam dois períodos a e b pertencentes a $[j, l]$ onde x_a e x_b são maiores do que zero. Se isso fosse verdade, teríamos um pseudo-ciclo $C = o, a, q_a, r_a, a+1, \ldots, b-1, q_{b-1}, r_{b-1}, b, o$, e isso contraria a hipótese de termos um fluxo sem pseudo-ciclos.

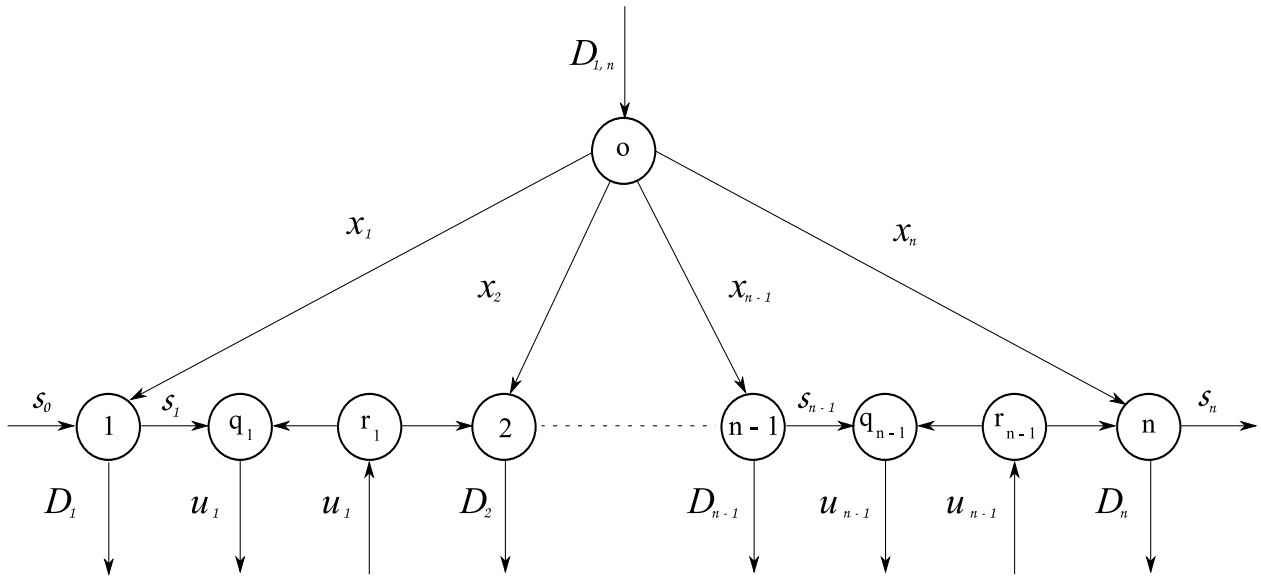


Figura 2.6: Rede equivalente com substituição da capacidade de armazenamento por vértices com demandas.

Portanto, existe uma solução ótima na qual, dentro de um intervalo de regeneração, se produz no máximo uma única vez, o que prova a proposição. □

Com base nos fatos acima mostrados, em 1973 Love [Lov73] apresentou uma solução para o problema. Vamos agora mostrá-la.

Consideremos a seguinte função $[C(i,j,k)]_{\alpha_i}^{\beta_l}$, onde $C(i,j,l)$ é o custo de se produzir dentro de um *intervalo de regeneração* $[i,l]$ no período j e $\alpha_i \in \{0, u_{i-1}\}$, $\beta_l \in \{0, u_l\}$ são os valores do estoque, no início dos período i e no final do período l, respectivamente.

$$[C(i,j,l)]_{\alpha_i}^{\beta_l} = \begin{cases} \infty & , \text{ se for inviável} \\ \sum_{t=i}^{j-1} h_t(\alpha_i - D_{i,t}) + \sum_{t=j}^{l} h_t(D_{t+1,l} + \beta_l) & , \text{ se } D_{i,l} - \alpha_i + \beta_l = 0 \\ f_j + p_j(D_{i,l} - \alpha_i + \beta_l) + \sum_{t=i}^{j-1} h_t(\alpha_i - D_{i,t}) + \sum_{t=j}^{l} h_t(D_{t+1,l} + \beta_l) & , \text{ caso contrário} \end{cases} \tag{2.22}$$

onde as condições de viabilidade são:

$$D_{i,l} - \alpha_i + \beta_l \geq 0 \tag{2.23}$$

$$0 < \alpha_i - D_{i,t} < u_t, \text{ para } t \in [i, j-1] \tag{2.24}$$

$$0 < D_{t+1,l} - \beta_j < u_t, \text{ para } t \in [j, l) \tag{2.25}$$

Dado o acima, é natural considerarmos a função

$$[G(i,l)]_{\alpha_i}^{\beta_l} = \min_{j:i \leq j \leq l} \{[C(i,j,l)]_{\alpha_i}^{\beta_l}\} \tag{2.26}$$

onde $[G(i,l)]_{\alpha_i}^{\beta_l}$ é o menor custo de se produzir no *intervalo de regeneração* $[i,l]$ tendo o estoque inicial α_i e o final β_l.

Consideremos agora a função $F(i, \alpha_i)$ que nos dá o menor custo de se produzir dos instantes i até n dado que tinha α_i no estoque. Essa função pode ser descrita, usando $G(.)$, pela recorrência abaixo:

$$F(i,\alpha_i) = \begin{cases} \min\limits_{\substack{i<l\leq n \\ \beta_l \in \{0,u_l\}}} \{[G(i,l)]_{\alpha_i}^{\beta_l} + F(l+1,\beta_l)\} & \text{, se } 1 \leq i \leq n-1 \\ 0 & \text{, se } i = n, \alpha_i = 0 \\ \infty & \text{, caso contrário} \end{cases} \tag{2.27}$$

Dado isso, o valor de $F(1,0)$ nos dá a solução do problema.

A equação (2.27) pode ser interpretada como um problema de caminho mais curto, com vértices $i \in [0,n]$ que representam períodos de tempo onde o estoque está vazio ao final do período e $j' \in [1,n-1]$ representam os períodos de tempo onde o estoque estará cheio ao final. O custo do arco (i,j') é determinado pelo custo do intervalo de regeneração (i,j'), se for viável, ou infinito, caso contrário.

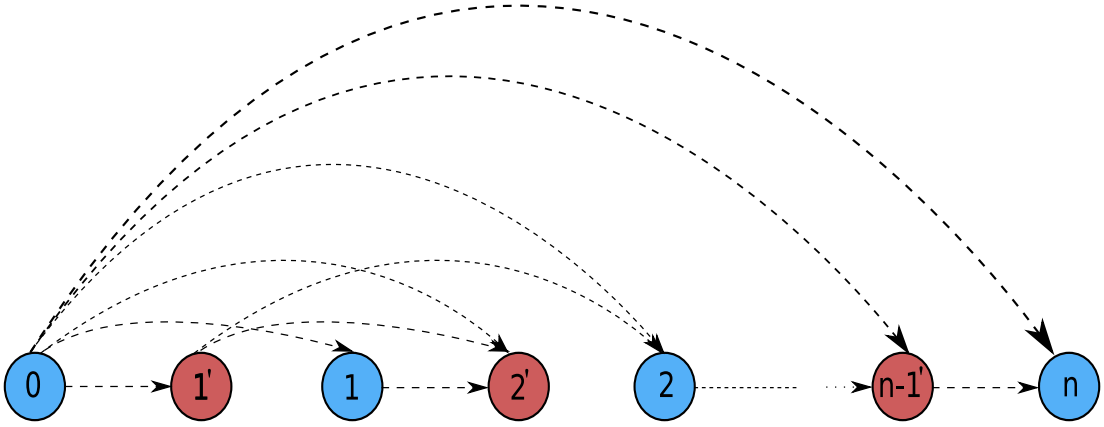


Figura 2.7: Representação do *LS-A* como um problema de caminho mais curto

Podemos, observando que podemos calcular $[C(i,j,l)]_{\alpha_i}^{\beta_l}$ a partir de $[C(i,j,l-1)]_{\alpha_i}^{\beta_{l-1}}$ em tempo $O(1)$, implementar um algoritmo utilizando programação dinâmica de complexidade $O(n^3)$ que nos dê a solução.

2.3 O Problemas com restrições na produção (*LS-C*)

Restrições na capacidade de produção aparecem de modo bastante natural na maioria dos casos. Diferentemente dos problemas mostrados até aqui, que podem ser resolvidos em tempo polinomial, o problema com restrições na capacidade, onde as capacidades c_t não obedecem a nenhum padrão específico é um problema NP-difícil. Para mostrar isso, vamos reduzir o *Problema da Mochila* ao problema de dimensionamento de lotes com restrições na capacidade de produção. A redução original pode ser encontrada em [FLK80].

Abaixo reapresentamos a formulação do problema *LS-C*

$$\min \sum_{t=1}^{n} (p_t x_t + h_t s_t + f_t y_t) \tag{2.28}$$

sujeito a:

$$s_{t-1} + x_t = s_t + D_t, \text{ para } t = 1, \ldots, n \tag{2.29}$$

$$\sum_{t=1}^{n} x_t = D_{1,n} \tag{2.30}$$

$$\sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n \tag{2.31}$$

$$x_t \leq c_t y_t, \text{ para } t = 1, \ldots, n \tag{2.32}$$

$$x_t \geq 0, s_t \geq 0, \text{ para } t = 1, \ldots, n \tag{2.33}$$

$$s_0 = s_n = 0 \tag{2.34}$$

$$x_t \in \mathbb{R}, s_t \in \mathbb{R}, y_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{2.35}$$

E agora apresentamos a modelagem clássica do *Problema da Mochila*

$$\max \sum_{t=1}^{n} v_t z_t \tag{2.36}$$

$$\textbf{sujeito a:} \quad \sum_{t=1}^{n} w_t z_t \leq M \tag{2.37}$$

$$z_t \in \{0,1\}, \text{ para } t = 1, \ldots, n. \tag{2.38}$$

onde v_t representa o valor do item t, w_t é o peso do item t e M é a capacidade da mochila.

Consideremos uma instância do *Problema da Mochila*. Vamos agora construir uma instância do problema *LS-C* com os seguintes valores:

- $p_t = 0$, para $t = 1, \ldots, n$;
- $h_t = 0$, para $t = 1, \ldots, n$;
- $c_t = w_t$, para $t = 1, \ldots, n$;
- $f_t = v_t$, para $t = 1, \ldots, n$;
- $y_t = 1 - z_t$, onde $z_t \in \{0,1\}$, para $t = 1, \ldots, n$;
- $D_j = 0$, para $j = 1, \ldots, n-1$;
- $D_n = \sum_{t=1}^{n} w_t - M$.

O problema fica:

$$\min \sum_{t=1}^{n} v_t(1 - z_t) \tag{2.39}$$

$$\textbf{sujeito a:} \quad \sum_{t=1}^{n} x_t = \sum_{t=1}^{n} w_t - M \tag{2.40}$$

$$x_t \leq w_t(1 - z_t), \text{ para } t = 1, \ldots, n \tag{2.41}$$

$$x_t \geq 0, \text{ para } t = 1, \ldots, n \tag{2.42}$$

$$x_t \in \mathbb{R}, z_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{2.43}$$

unindo (2.40) e (2.41) podemos reescrever o problema da seguinte forma:

$$\min \sum_{t=1}^{n} v_t - \sum_{t=1}^{n} v_t z_t \tag{2.44}$$

$$\textbf{sujeito a:} \quad \sum_{t=1}^{n} w_t - \sum_{t=1}^{n} w_t z_t \geq \sum_{t=1}^{n} w_t - M \tag{2.45}$$

$$z_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{2.46}$$

Simplificando o problema e observando que $\sum_{t=1}^{n} v_t$ é uma constante, o problema se torna

$$\max \sum_{t=1}^{n} v_t z_t \tag{2.47}$$

$$\textbf{sujeito a:} \quad \sum_{t=1}^{n} w_t z_t \leq M \tag{2.48}$$

$$z_t \in \{0, 1\}, \text{ para } t = 1, \ldots, n, \tag{2.49}$$

que é o *Problema da Mochila*. Portanto, se soubéssemos resolver *LS-C* em tempo polinomial, saberíamos resolver o *Problema da Mochila*, que é NP-difícil, em tempo polinomial. Logo o problema

LS-C também é NP-difícil.

Uma análise mais aprofundada da complexidade de problemas de dimensionamento de lotes pode ser encontrada em [BY82] de *Bitran e Yanasse*. Além disso, algoritmos *pseudo-polinomiais* podem ser encontrados em [CHL94] e [SW98].

2.3.1 A estrutura da solução ótima de LS-C

Embora tenhamos mostrado que o problema é NP-difícil na seção anterior, nós podemos caracterizar a solução ótima do problema.

Para isso, vamos utilizar a definição de *intervalo de regeneração* que apresentamos na seção 2.1 para mostrarmos a seguinte propriedade, que foi primeiramente mostrada por *Florian e Klein* [FK71]:

Proposição 2.7. Existe uma solução ótima em que, dentro de um intervalo de regeneração $[i, l]$, existe no máximo um período de tempo j onde $0 < x_j < c_j$. Para todos os outros períodos de tempo t, $t \in [i, l]$ e $t \neq j$, temos que ou $x_t = 0$ ou $x_t = c_t$.

Demonstração. De modo análogo ao que fizemos na seção anterior, vamos reduzir a rede com restrições na capacidade de produção a uma rede onde a solução é equivalente.

Para isso, vamos eliminar os arcos (o, i) e adicionar vértices w_i e z_i com demandas c_i e $-c_i$, respectivamente. Ao conjunto de arcos, adicionaremos (o, w_i), (z_i, w_i) e (z_i, i), pelos quais passarão $x_i, c_i - x_i$ e x_i unidades de fluxo, respectivamente. O fluxo no arco (o, w_i) tem custo $p_i x_i + f_i$ e nos arcos (z_i, w_i) e (z_i, i) o custo é zero.

Uma solução ótima nessa rede construída, também será uma solução ótima no problema original.

Consideramos um fluxo sem pseudo-ciclos, que é uma solução ótima para o problema e um intervalo de regeneração $[j, l]$ e, portanto, para todo $t \in [j, l-1]$, temos que s_t é maior do que zero, por definição.

Suponhamos, por contradição, que existam dois períodos a e b pertencentes a $[j, l]$ nos quais

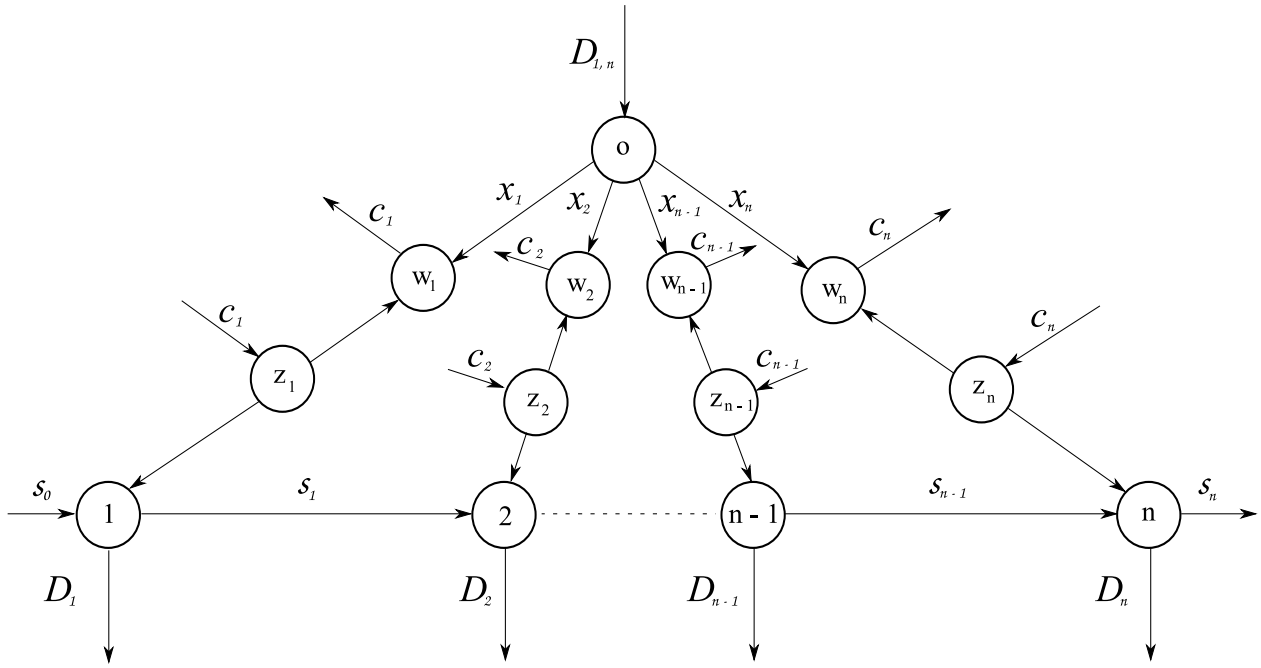


Figura 2.8: Rede equivalente com substituição da capacidade de produção por vértices com demandas.

temos que $0 < x_a < c_a$ e $0 < x_b < c_b$. Mas, se isso fosse verdade, teríamos um pseudo-ciclo $C = o, a, a+1, \ldots, b-1, , b, o$, e isso contraria a hipótese de termos um fluxo sem pseudo-ciclos.

Portanto, existe uma solução ótima em que no máximo um único período de tempo t, dentro de um intervalo de regeneração em que $0 < x_t < c_t$, o que prova a proposição. □

Embora o problema seja *NP-difícil* no seu caso geral, existe uma solução polinomial para o caso onde a capacidade é constante em todos os períodos de tempo, conforme foi mostrado em [FK71] e posteriormente aprimorado em [HW96].

A seguir, estudemos esse caso especial.

2.3.2 Algoritmo para *LS-C* com capacidade constante

Conforme vimos na seção 2.1, quando o problema pode ser quebrado em *intervalos de regeneração*, ele pode ser interpretado como um problema de caminho mais curto, onde o custo de um arco $(i-1, l)$ é o custo de produção de um intervalo de regeneração $[i, l]$. Também observamos que existem $O(n^2)$ intervalos de regeneração.

Vamos considerar aqui, sem perda de generalidade, que em todos os períodos de tempo a demanda é menor do que a capacidade, isto é,

$$D_t \leq C \text{ para } t = 1, \ldots, n. \tag{2.50}$$

Caso, para uma instância do problema isso não seja verdade para um dado período t, podemos tomar um problema equivalente, repassando a demanda excedente para o período anterior. Ou seja, $D'_t = C$ e $D'_{t-1} = D_{t-1} + (C - D_t)$. Se o procedimento se seguir até t igual a 1 e tivermos $D_1 > C$ é porque o problema é inviável.

No caso em que consideramos que a capacidade de produção é limitada, pela proposição 2.7, existe uma solução ótima em que, dentro de um intervalo de regeneração, existe no máximo um período em que se produz, mas não na capacidade máxima. Portanto, dentro de um intervalo de regeneração, a seguinte igualdade é válida:

$$D_{i,l} = KC + f, \text{ onde } K \in \mathbb{Z}_+,\ 0 \leq f < C. \tag{2.51}$$

Onde K é o total de vezes em que se deve produzir na capacidade máxima dentro do intervalo de regeneração $[i, l]$ e f é a quantidade fracionária (isto é, uma fração da capacidade de produção) que deve ser produzida para atender a demanda do intervalo.

Denotemos por $[t_{min}, t_{max}] \subseteq [i, l]$ o intervalo onde a quantidade de produção fracionária deve

ser produzida, para termos um planejamento viável e para que $[i, l]$ continue sendo um intervalo de regeneração. Dado isso, podemos determinar t_{min} e t_{max} como:

$$t_{min} = \min\{t \in [i, l] : D_{i,t} < (t - i)C + f\}, \tag{2.52}$$

$$t_{max} = \max\{t \in [i, l] : D_{t,l} \geq f\}. \tag{2.53}$$

Consideremos agora o problema de obter um planejamento de custo ótimo de um intervalo de regeneração $[i, l]$, dado que fixamos um período $t \in [t_{min}, t_{max}]$ no qual será produzida a quantidade fracionária. Chamemos esse problema auxiliar de $P(t)$. A um problema ligeiramente diferente, onde ganhamos f unidades no instante t sem nenhum custo adicional e sem ocupar nenhuma capacidade de produção no instante t, ao invés de produzirmos f, chamaremos de $P(t)'$.

A seguir, mostraremos em linhas gerais o algoritmo que determina o planejamento de menor custo, dentro de um intervalo de regeneração, usando os subproblemas que acabamos de definir. As etapas envolvidas são:

- Obtenha a solução de $P(t_{max})'$;
- Calcule o custo de $P(t_{max})$ a partir de $P(t_{max})'$;
- Para cada $t \in [t_{max} - 1, t_{min}]$, obtenha a solução de $P(t)'$ a partir da solução $P(t+1)'$ e calcule o custo de $P(t)$ a partir de $P(t)'$;
- O problema $P(t)$, $t \in [t_{min}, t_{max}]$ que tem o menor custo é a resposta.

Vamos mostrar como esse algoritmo que acabamos de mostrar pode ser implementado em tempo amortizado $O(n)$. Além disso, como existem $O(n^2)$ intervalos de regeneração, o custo de todos os intervalos podem ser calculados em $O(n^3)$. Com todos os custos dos intervalos calculados, podemos

resolver o problema de caminho mais curto de $n+1$ vértices (ilustrado pela figura 2.2) em $O(nlog(n))$ e, portanto, o problema como um todo em $O(n^3)$.

Consideremos a função $A(t,j)$ para $t,j \in [i,l]$ que nos dá um limitante inferior para o número de períodos onde teremos uma produção a plena capacidade, dentro do intervalo $[i,j]$, no problema $P(t)'$.

$$A(t,j) = \begin{cases} \left\lceil \dfrac{D_{i,j}}{C} \right\rceil & \text{, se } j < t, \\ \left\lceil \dfrac{D_{i,j} - f}{C} \right\rceil & \text{, se } j \geq t. \end{cases} \tag{2.54}$$

Vamos definir w_k, para todo $k = 1, \ldots, K$ tal que $w_k = \min\{j \in [i,l] \text{ para o qual } A(t,j) = k\}$. O período w_k será denominado *período de escolha*, pois nos força a escolher k períodos dentro do intervalo $[i, w_k]$.

Apresentaremos agora uma estratégia para a escolha dos K períodos onde se produz a toda capacidade no problema $P(t)'$.

Algoritmo *K-Períodos*(i, l, t)

01 $K \leftarrow \left\lfloor \dfrac{D_{i,l}}{C} \right\rfloor$

02 $B \leftarrow \emptyset$

03 $k \leftarrow 0$

04 **para** $j = i, \ldots, l$ **faça:**

05 **se** $A(t,j) = k+1$ **então**

06 $w_{k+1} \leftarrow j$

07 $k \leftarrow k+1$

08 **para** $k = 1, \ldots, K$ **faça:**

09 $b_k \leftarrow \min\{j : j \in W = \{i, \ldots, w_k\} \backslash B$ **e** $p'_j C + q_j = \min_{z \in W} p'_z C + q_z\}$

10 $B \leftarrow B \bigcup \{b_k\}$

11 devolva B

Observe que, para esse algoritmo servir para calcular $P(t)$, basta excluirmos o t, juntamente com os períodos já escolhidos, na linha *09*, que ficaria da seguinte forma:

$$b_k \leftarrow \min\{j : j \in W = \{i, \ldots, w_k\} \backslash B \bigcup \{t\} \text{ e } p'_j C + q_j = \min_{z \in W} p'_z C + q_z\}. \tag{2.55}$$

Vamos agora mostrar a validade da solução obtida.

Lema 2.2. A solução de $P(t)'$ obtida pelo algoritmo descrito é ótima.

Demonstração. Consideremos um conjunto de períodos de tempo $E = \{e_1, \ldots, e_K\}$, que seja uma solução ótima de $P(t)'$, onde as escolhas ocorrem o mais cedo possível. Vamos supor por contradição que a solução $B = \{b_1, \ldots, b_K\}$ obtida pelo algoritmo *K-Períodos* é diferente da solução ótima E.

Suponhamos que o k-ésimo período seja o primeiro período de tempo que B e E não compartilhem. Observamos que sempre existe tal período, por hipótese, pois se não existisse teríamos $E = B$. Pela definição de w_k, sabemos que tanto e_k quanto b_k pertencem ao intervalo $[i, w_k]$. Por hipótese, ou $e_k < b_k$ ou $p'_{e_k} C + q_{e_k} < p'_{b_k} C + q_{b_k}$, pois E é uma solução ótima em as escolhas ocorrem o mais cedo possível. Porém, pela escolha feita na linha *09* do algoritmo *K-Períodos*, isso é uma contradição. Portanto a solução B obtida pelo algoritmo é igual a solução ótima E. □

Vamos agora fazer uma análise de complexidade do algoritmo *K-Períodos*. Por facilidade vamos definir $T = l - i$. Uma análise direta, nos dá que o algoritmo consome tempo $O(T^2)$. Porém, será mostrado a seguir, que, se calculado para todos os intervalos de regeneração, o custo amortizado será de $O(n)$ por intervalo.

Lema 2.3. Sejam i e l tais $1 \leq i \leq l \leq n-1$, consideremos os intervalos de regeneração $[i,l]$ e $[i,l+1]$ e chamaremos de t_{max} e t'_{max} o último período no qual a quantidade fracionária pode ser produzida, para os intervalos $[i,l]$ e $[i,l+1]$, respectivamente. A solução ótima do problema $P(t_{max})'$, no intervalo $[i,l]$ é um subconjunto da solução ótima do problema $P(t'_{max})'$, definido no intervalo $[i,l+1]$.

Demonstração. Note que as funções $A(t_{max},j)$ e $A(t'_{max},j)$ que determinam os períodos de escolha têm os mesmos valores para $j = i,\ldots,t_{max}-1$. Se $A(t_{max},t_{max}) = A(t'_{max},t_{max})$ então obviamente a solução de $P(t_{max})'$ é um subconjunto da solução de $P(t'_{max})'$. Vamos analisar agora o caso onde são diferentes. Primeiro, lembremos que $D_{i,l} = KC + f$, para algum K inteiro e algum f tal que $0 \leq f < C$. Como $A(t_{max},t_{max}) \neq A(t'_{max},t_{max})$ temos que $\left\lceil \frac{KC}{C} \right\rceil \neq \left\lceil \frac{KC+f}{C} \right\rceil$ e, portanto, $A(t_{max},t_{max}) = K$, $A(t'_{max},t_{max}) = K+1$ e, além disso, $f > 0$. Como $0 \leq D_{l-1} \leq C$, temos que $A(t_{max},t_{max}-1) \geq \left\lceil \frac{(K-1)C+f}{C} \right\rceil = K$.

Isso mostra que todos os K períodos de escolha do problema $P(t_{max})'$ são períodos de escolha do problema $P(t'_{max})'$ e, portanto, a solução de $P(t_{max})'$ é um subconjunto da solução de $P(t'_{max})'$, como queríamos demonstrar. □

Com o resultado desse lema, sabemos que para todo i fixo, as soluções dos intervalos de regeneração $[i,l]$ são subconjuntos das soluções de $[i,n]$. Portanto, podemos calcular todos os intervalos $[i,l]$, $i < l \leq n$, com o custo do intervalo $[i,n]$. Como o custo do algoritmo *K-Períodos*, para o intervalo $[i,n]$ é $O((n-i)^2)$, o custo de calcular a solução de todos os intervalos de regeneração é proporcional a $\sum_{i=0}^{n-1}(n-i)^2$, que é $O(n^3)$. Como existem $O(n^2)$ intervalos de regeneração, o custo amortizado por intervalo de regeneração será $O(n)$.

Dado que mostramos podemos calcular $P(t_{max})'$ em tempo amortizado $O(n)$, vamos apresentar

o lema que nos mostra como obter as demais soluções iterativamente.

Lema 2.4. Existe no máximo um período diferente na solução dos problemas $P(t+1)'$ e $P(t)'$ pelo algoritmo apresentado anteriormente. E além disso, no caso de haver diferença, podemos obter a solução de $P(t)'$ através da solução $P(t+1)'$ movendo a produção de um período $[i,t]$ para $[t+1,l]$.

Demonstração. Primeiramente observamos que os períodos onde se produzem a plena capacidade são determinados pelos períodos de escolha. Logo, só haverá diferença entre os períodos de produção, se houver diferença nos períodos de escolha. Vamos agora mostrar que ou os períodos de escolha são idênticos, nos problemas $P(t)'$ e $P(t+1)'$ ou t é um período de escolha no problema $P(t+1)'$ mas não no problema $P(t)'$.

Notemos que $A(t+1,j) = A(t,j)$ para $j \in \{i,\ldots,l\}\backslash\{t\}$. E, além disso, $A(t+1,t)-1 \leq A(t,t) \leq A(t+1,t)$, pela definição da função $A(.,.)$. Como a função $A(.,.)$ só pode assumir valores inteiros, pela definição, ou temos que $A(t,t) = A(t+1,t)$ e, portanto, $A(t+1,j) = A(t,j)$, para todo $j \in [i,l]$ e, consequentemente, os K períodos de escolha são idênticos e a solução é a mesma para $P(t)$ e $P(t+1)$, ou $A(t,t)-1 = A(t+1,t)$. Se o caso for o último, então o período t será um período de escolha no problema $P(t+1)$, mas não no problema $P(t)$, pois $A(t+1,t-1) = A(t,t-1) \leq A(t,t) = A(t+1,t)+1$ e $A(t+1,t-1) \leq A(t+1,t) \leq A(t+1,t-1)+1$ implicam que $A(t+1,t) = A(t+1,t-1)+1$ e que $A(t,t) = A(t,t-1)$.

Consideremos que t é o k-ésimo período de escolha do problema $P(t+1)'$. Como t não é período de escolha do problema $P(t)'$ e todos os outros períodos de escolha são comuns aos dois problemas, o k-ésimo período de escolha de $P(t)'$ é um período u, tal que $t < u < w_{k+1}$.

Vamos, a partir de agora denotar por $b_{k,t}$ a k-ésima escolha do algoritmo para o problema $P(t)'$. Observe que se $b_{k,t} = b_{k,t+1}$, os problemas $P(t)'$ e $P(t+1)'$ têm ambos a mesma solução.

Vamos nos restringir ao caso em que $b_{k,t} \neq b_{k,t+1}$, que só ocorre se $b_{k,t} \in [t+1,u]$ e analisar a escolha feita na iteração $k+1$ para os problemas.

Na iteração, duas coisas podem ocorrer:

(i) Se tivermos $b_{k+1,t} = b_{k,t+1}$, trivialmente também teremos que $b_{k,t+1} = b_{k+1,t}$ e, novamente, a solução de ambos os problemas será a mesma.

(ii) Senão, teremos $b_{k+1,t} \in [t+1, w_{k+1}]$ e

$$b_{k+1,t+1} = \begin{cases} b_{k,t} & \text{, se } p'_{b_{k,t}} C + q_{b_{k,t}} \leq p'_{b_{k+1,t}} C + q_{b_{k+1,t}}, \\ b_{k+1,t} & \text{, caso contrário.} \end{cases} \tag{2.56}$$

Observamos aqui que, até o momento existe apenas um período, no caso $b_{k,t+1} \in [i, t]$ que não parte da solução de $P(t)'$ e apenas $r_t \in \{b_{k,t+1}, b_{k+1,t}\}$ que não faz parte da solução $P(t+1)$.

Para as iterações $k+2$ até K, podemos proceder da mesma maneira como fizemos em (i) e (ii), o que nos leva a concluir que existe no máximo um período diferente nas soluções para $P(t)'$ e $P(t+1)'$. □

Como no nosso problema precisamos obter a solução de $P(t)$ a partir de $P(t)'$, abaixo segue o lema que nos mostra como obter uma solução a partir da outra.

Lema 2.5. Consideremos uma solução de $P(t)'$, $t \in [t_{min}, t_{max}]$, tal que t é escolhido como um período de produção. Podemos obter uma solução ótima de $P(t)$, a partir de uma solução ótima de $P(t)'$, apenas realocando a produção do período t para um outro período.

Demonstração. Consideremos as soluções $B' = \{b'_1, \ldots, b'_K\}$ e $B = \{b_1, \ldots, b_K\}$ obtidas pelo algoritmo *K-Períodos* para os problemas problemas $P(t)'$ e $P(t)$, respectivamente, tal que $b'_k = t$.

Se $b_k \notin B'$, então basta realocarmos a produção do período t para o período b_k. Caso contrário, existe um período $b'_j = b_k$, tal que $j > k$. Pela forma como o algoritmo escolhe os períodos dentro

do intervalo $[i, w_{j+1}]$ temos que

$$b'_{j+1} = \begin{cases} b_j & \text{, se } p'_{b_j}C + q_{b_j} \leq p'_{b_{j+1}}C + q_{b_{j+1}}, \\ b_{j+1} & \text{, caso contrário.} \end{cases} \tag{2.57}$$

e denotaremos

$$r_{j+1} = \begin{cases} b_j & \text{, se } p'_{b_j}C + q_{b_j} > p'_{b_{j+1}}C + q_{b_{j+1}}, \\ b_{j+1} & \text{, caso contrário.} \end{cases} \tag{2.58}$$

como o período rejeitado do planejamento B nas $j+1$ primeiras escolhas. De forma análoga, observamos que

$$b'_{j+z} = \begin{cases} r_{j+z-1} & \text{, se } p'_{r_{j+z-1}}C + q_{b_{j+z-1}} \leq p'_{b_{j+z}}C + q_{b_{j+z}}, \\ b_{j+z} & \text{, caso contrário.} \end{cases} \tag{2.59}$$

e

$$r_{j+z} = \begin{cases} r_{j+z-1} & \text{, se } p'_{r_{j+z-1}}C + q_{b_{j+z-1}} > p'_{b_{j+z}}C + q_{b_{j+z}}, \\ b_{j+z} & \text{, caso contrário.} \end{cases} \tag{2.60}$$

para $z = 2, \ldots, K - j$. De onde concluímos que r_K é o único período que pertence a B mas não a B'. Logo, podemos obter uma solução ótima de $P(t)$ apenas trocando b'_k por r_K. □

Conforme foi mostrado no lema 2.4, a solução de $P(t)'$ pode ser obtida a partir da solução de $P(t+1)'$ trocando apenas um período de produção. O que será mostrado a seguir é como escolher tal período de produção.

Para decidirmos qual período de tempo deve ser trocado para termos uma solução ótima para o problema $P(t)'$, precisamos antes identificar quais trocas mantêm a solução do problema viável.

Tomemos

$$\tau = \min\{j \in [t, l] : s_j < C\} \text{ e} \tag{2.61}$$

$$\zeta = \max\{j \in [i, t] : s_{j-1} < C\}. \tag{2.62}$$

Note que sempre existem tais τ e ζ, pois $[i, l]$ é um intervalo de regeneração e, portanto, $s_{i-1} = s_l = 0$. Como queremos mover C unidades de produção, para $[i, l]$ continuar sendo um intervalo de regeneração, a troca, necessariamente, deve ser feita de um período em $[\zeta, t]$ para um período em $[t + 1, \tau]$. Agora, basta ver quais períodos nesses intervalos fazem com que a solução obtida seja ótima. Sejam

$$\gamma_t = \{j : j \in [\zeta, t], j \in B_{t+1} \text{ e } p'_j C + q_j = \max_{z \in [\zeta, t]} p'_z C + q_z\} \text{ e} \tag{2.63}$$

$$\delta_\tau = \min\{j : j \in [i, \tau], j \notin B_{t+1} \text{ e } p'_j C + q_j = \min_{z \in [i, \tau]} p'_z C + q_z\}. \tag{2.64}$$

Se $p'_{\delta_\tau} C + q_{\delta_\tau} < p'_{\gamma_t} C + q_{\delta_\tau} q$ então a troca deve ser feita, caso contrário, a solução de $P(t + 1)'$ é igual a solução de $P(t)'$. No caso em que a troca é feita, observe que $\gamma_t > t$, pois as soluções B_t e B_{t+1} diferem pelo fato de que em B_{t+1} há uma período de produção no intervalo $[i, t]$ a mais do que na solução B_t. Se $\gamma_t \leq t$, essa propriedade seria violada.

Para obtermos $P(t)$ a partir de $P(t)'$, primeiros observamos se t foi escolhido como um período de produção. Se não, então $P(t) = P(t)'$. Caso tenha sido escolhido, então basta realocar a produção para um período no intervalo $[i, \tau]$, que seja o mais barato possível. Ou seja, basta substituir t por δ_τ que obtemos a solução de $P(t)$ a partir da solução de $P(t)'$.

Será mostrado a seguir como evitar cálculos desnecessários, para conseguirmos a complexidade desejada de $O(n)$.

Lema 2.6. Se, para obtermos a solução de $P(t)'$ a partir da solução de $P(t+1)'$ trocamos um período

de produção no intervalo $[\zeta, t]$ por um outro período de produção pertencente ao intervalo $[t+1, \tau]$ então, não é necessário realizar mais nenhuma troca para obtermos as soluções dos problemas $P(j)'$, $j = s, \ldots, t-1$.

Demonstração. Primeiro, vamos mostrar que, quando a troca é realizada para obtermos a solução de $P(t)'$ a partir de $P(t+1)'$, o valor do estoque s_t fica menor que C. Na solução de $P(t+1)'$ temos que

$$s_t = \sum_{j=i}^{t} x_j - D_{i,t} \tag{2.65}$$

que pelas escolhas feitas pelo algoritmo *K-Períodos* é equivalente a

$$s_t = \left\lceil \frac{D_{i,t}}{C} \right\rceil C - D_{i,t} \Longrightarrow \tag{2.66}$$

$$s_t \leq \left(\frac{D_{i,t}}{C} + 1 \right) C - D_{i,t} \Longrightarrow \tag{2.67}$$

$$s_t \leq C. \tag{2.68}$$

Como para obtermos a solução de $P(t)'$ a partir de $P(t+1)'$ movemos C unidades para o intervalo $[t+1, \tau]$ e ganhamos f unidades no período t, pela definição de $P(t)'$, temos que o novo valor de s_t após a troca, que chamaremos de s'_t é

$$s'_t = s_t - C + f \leq f < C. \tag{2.69}$$

Suponha, por contradição, que exista um período $\sigma \in [\zeta, t]$ em que seja vantajoso fazer uma troca de um período em $[\zeta, \sigma]$ por um período em $[\sigma+1, t]$. Podemos nos limitar a t, ao invés de τ pelo que acabamos de mostrar e pela definição de τ.

Como essa escolha já era viável em $P(t+1)'$, dizer que a troca é vantajosa implica em dizer que a solução de $P(t+1)'$ não era ótima, o que é uma contradição. □

Com os resultados mostrados até aqui, já podemos construir o nosso algoritmo. Antes, mostraremos um algoritmo auxiliar que calcula todos os δ's no intervalo $[i, \tau]$.

Algoritmo *CalculaDelta*(i, τ, x)

01 $inicio \leftarrow min\{k : k \in [i,\tau]$ **e** $x_k < C\}$

02 $\delta_{inicio} \leftarrow inicio$

03 **para** $v = inicio + 1, \ldots, \tau$ **faça:**

04 **se** $x_v < C$ **e** $p'_v C + q_v < p'_{\delta_{v-1}} C + q_{\delta_{v-1}}$ **então**

05 $\delta_v \leftarrow v$

06 **senão**

07 $\delta_v \leftarrow \delta_{v-1}$

08 devolva $\delta = \{\delta_{inicio}, \ldots, \delta_\tau\}$

E análogo ao que acabou de ser feito, segue o algoritmo para calcular todos os γ's no intervalo $[\zeta, t]$.

Algoritmo *CalculaGamma*(ζ, t, x)

01 $inicio \leftarrow min\{k : k \in [\zeta,t]$ **e** $x_k \geq C\}$

02 $\gamma_{inicio} \leftarrow inicio$

03 **para** $r = inicio + 1, \ldots, t$ **faça:**

04 **se** $x_r \geq C$ **e** $p'_r C + q_r > p'_{\gamma_{r-1}} C + q_{\gamma_{r-1}}$ **então**

05 $\gamma_r \leftarrow r$

06 **senão**

07 $\quad\quad \gamma_r \leftarrow \gamma_{r-1}$

08 devolva $\gamma = \{\gamma_{inicio}, \ldots, \gamma_t\}$

O algoritmo que calcula o planejamento ótimo do intervalo de regeneração $[i, l]$ é o seguinte:

Algoritmo *Intervalo-Custo-Mínimo*$(i,\ l)$

01 $K \leftarrow \left\lfloor \dfrac{D_{i,l}}{C} \right\rfloor$

02 $f \leftarrow D_{i,l} - KC$

03 $t_{min} \leftarrow \min\{t \in [i, l] : D_{i,t} < (t - i)C + f\}$

04 $t_{max} \leftarrow \max\{t \in [i, l] : D_{t,l} \geq f\}$

05 $x_j \leftarrow 0, \forall j \in [i, l]$

06 $B \leftarrow$ *K-Períodos*$(i,\ l,\ t_{max})$

07 $x_b \leftarrow C, \forall b \in B$

08 $x_{t_{max}} \leftarrow x_{t_{max}} + f$

09 $s_{i-1} \leftarrow 0$

10 **para** $j = i, \ldots, j$ **faça:**

11 $\quad\quad s_j \leftarrow s_{j-1} + x_j - D_j$

12 $\zeta \leftarrow \max\{j \in [i, t_{max}] : s_{j-1} < C\}$

13 $\tau \leftarrow t_{max}$

14 $\gamma \leftarrow$ *CalculaGamma*$(\zeta,\ t_{max},\ x)$

15 $\delta \leftarrow$ *CalculaDelta*$(i,\ \tau,\ x)$

16 $custo'[t_{max}] \leftarrow \sum_{\forall b \in B} p'_b C + q_b$

17 $y \leftarrow 0$

18 **se** $f > 0$ **então**

19 $y = 1$

20 **se** $x_{t_{max}} > C$ **então**

21 $custo[t_{max}] \leftarrow custo'[t_{max}] + p'_{\delta_{t_{max}}} f + q_{\delta_{t_{max}}} y$

22 **senão**

23 $custo[t_{max}] \leftarrow custo'[t_{max}] + p'_{t_{max}} f + q_{t_{max}} y$

24 troca-feita $\leftarrow falso$

25 **para** $t = t_{max} - 1, \ldots, t_{min}$ **faça:**

26 $x_{t+1} \leftarrow x_{t+1} - f$

27 $x_t \leftarrow x_t + f$

28 $s_t \leftarrow s_t + f$

29 **se** $s_t < C$ **então**

30 $\tau \leftarrow t$

31 **se** $t < \zeta$ **então**

32 troca-feita $\leftarrow falso$

33 $\zeta \leftarrow \max\{j \in [i, t] : s_{j-1} < C\}$

34 $\gamma \leftarrow$ *CalculaGamma*$(\zeta,\ t,\ x)$

35 **se** troca-feita $= falso$ **então**

36 **se** $p'_{\delta_\tau} C + q_{\delta_\tau} < p'_{\gamma_t} C + q_{\gamma_t}$ **então**

37 troca-feita $\leftarrow verdadeiro$

38 $x_{\gamma_t} \leftarrow x_{\gamma_t} - C$

39 $x_{\delta_\tau} \leftarrow x_{\delta_\tau} + C$

40 **para** $j = \gamma_t, \ldots, \delta_\tau - 1$ **faça:**

41 $s_j \leftarrow s_j - C$

42 $custo'[t] \leftarrow custo'[t+1] + (p'_{\delta_\tau} C + q_{\delta_\tau}) - (p'_{\gamma_t} C + q_{\gamma_t})$

43 $\tau \leftarrow t$

44 $\delta \leftarrow$ *CalculaDelta*$(\gamma_t,\ \tau,\ x)$

45 **senão**

46 $custo'[t] \leftarrow custo'[t+1]$

47 **se** $x_t > C$ **então**

48 $custo[t] \leftarrow custo'[t] + p'_{\delta_t} f + q_{\delta_t} y$

49 **senão**

50 $custo[t] \leftarrow custo'[t] + p'_t f + q_t y$

51 $t_{sol} \leftarrow \{t : custo[t] = \min_{z \in [t_{min}, t_{max}]} custo[z]\}$

52 $x_j \leftarrow 0, \forall j \in [i, l]$

53 $x_b \leftarrow C, \forall b \in B$

56 $x_{t_{sol}} \leftarrow x_{t_{sol}} + f$

57 $s_{i-1} \leftarrow 0$

58 **para** $j = i, \ldots, j$ **faça:**

59 $s_j \leftarrow s_{j-1} + x_j - D_j$

60 $\tau \leftarrow \min\{j \in [t, l] : s_j < C\}$

61 $\delta \leftarrow$ *CalculaDelta*$(i,\ \tau,\ x)$

62 **se** $x_{t_{sol}} > C$ **então**

63 $x_{\delta_\tau} \leftarrow C$

64 $x_{t_{sol}} \leftarrow x_{t_{sol}} - C$

65 devolva $x = \{x_i, \ldots, x_l\}$

Fora do laço que compreende as linhas *25* a *50*, todas as operações são no máximo $O(n)$. Dentro do laço os únicos pontos que não são $O(1)$ são as linhas *33-34* onde ζ é atualizado e os γ's no intervalo $[\zeta, t]$ são recalculados, as linhas *40-41* nas quais os valores do estoque são recalculados no intervalo $[\gamma_t, \delta_\tau]$ e a linha *44* onde o valor dos δ's são recalculados no intervalo $[\gamma_t, \tau]$.

Como, pelo lema 2.6 um período não precisa ser reconsiderado, o consumo de tempo total vai ser $O(n)$ e, portanto, o algoritmo como um todo tem custo $O(n)$.

2.4 O Problema com restrições de produção e armazenamento (*LS-AC*)

Conforme foi mostrado na seção anterior, o problema *LS-C* é *NP-difícil*, em sua formulação geral, e portanto o problema *LS-AC* também é. A resolução de problemas com essa caracterização se dá através de métodos de solução de *MIP*. Contudo, dado o até aqui exposto, segue de maneira bastante natural uma caracterização de soluções ótimas do problema.

Proposição 2.8. Existe uma solução ótima em que, dentro de um intervalo de regeneração $[i, l]$, existe no máximo um período de tempo j onde $0 < x_j < c_j$. Para todos os outros períodos de tempo t, $t \in [i, l]$ e $t \neq j$, temos que ou $x_t = 0$ ou $x_t = c_t$.

Demonstração. De modo análogo ao que fizemos na seção anterior, vamos reduzir a rede com restrições na capacidade de produção a uma rede onde a solução é a mesma.

Como fizemos para o problema *LS-C*, vamos eliminar os arcos (o, i) e adicionar vértices w_i e z_i com demandas c_i e $-c_i$, respectivamente. Ao conjunto de arcos, adicionaremos (o, w_i), (z_i, w_i) e (z_i, i) pelos quais passarão $x_i, c_i - x_i$ e x_i unidades de fluxo, respectivamente. O fluxo no arco (o, w_i)

tem custo $p_i x_i + f_i$ e o fluxo nos arcos (z_i, w_i) e (z_i, i) tem custo zero.

E análogo ao que fizemos para o problema *LS-A*, vamos eliminar os arcos $(i, i+1)$ e adicionar vértices q_i e r_i com demandas u_i e $-u_i$, respectivamente. Ao conjunto de arcos, adicionaremos (i, q_i), (r_i, q_i) e $(r_i, i+1)$, pelos quais passarão $s_i, u_i - s_i$ e s_i unidades de fluxo, respectivamente. O custo por unidade de fluxo em (i, q_i) é h_i e zero em (r_i, q_i) e $(r_i, i+1)$.

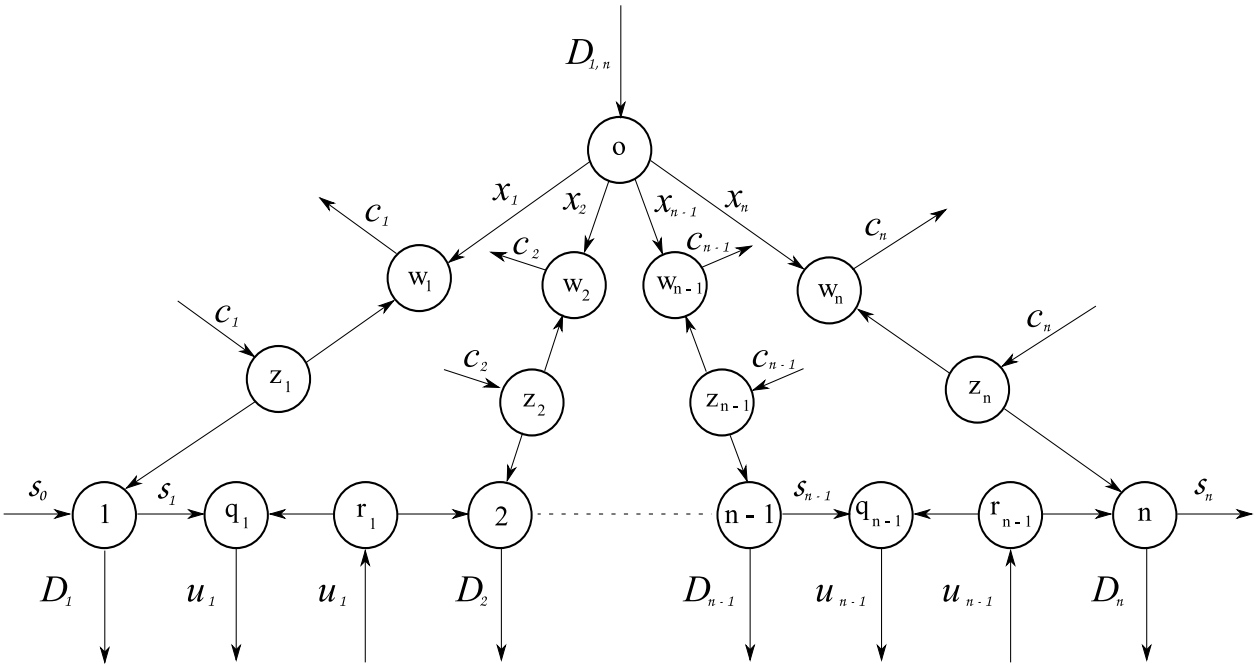


Figura 2.9: Rede equivalente com substituição da capacidade de armazenamento e de produção por vértices com demandas.

Uma solução ótima nessa rede construída, também será uma solução ótima no problema original.

Consideremos um fluxo ótimo sem pseudo-ciclos, que existe pela proposição (2.1), e um intervalo de regeneração $[j, l]$ que é parte da solução. Isso implica que para todo $t \in [j, l-1]$, temos que $0 < s_t < u_t$, por definição.

Suponhamos, por contradição, que existam dois períodos a e b pertencentes a $[j, l]$ tais que

$0 < x_a < c_a$ e $0 < x_b < c_b$. Se isso fosse verdade, teríamos um pseudo-ciclo $C = o, w_a, z_a, a, q_a, r_a, a+1, \ldots, b-1, q_{b-1}, r_{b-1}, b, z_b, w_b, o$, e isso contraria a hipótese de termos um fluxo ótimo sem pseudo-ciclos.

Portanto, existe uma solução ótima na qual, dentro de um intervalo de regeneração, se produz no máximo uma única vez, o que prova a proposição. □

2.5 Resumo do capítulo

Neste capítulo foi mostrada a importância do nível do estoque na construção de algoritmos e da caraterização de soluções ótimas. Mostramos que, na presença de restrições de capacidade de produção, de maneira geral, o problema se torna computacionalmente difícil de se resolver. Além disso, apresentamos algoritmos para os casos que não são computacionalmente difíceis.

A seguir apresentamos um resumo com os problemas e a respectiva complexidade dos problemas tratados.

Problema	Complexidade
LS-I	$O(nlogn)$
LS-A	$O(n^3)$
LS-C	$O(n^3)$ para capacidade constante *NP-difícil* no caso geral
LS-AC	*NP-difícil*

Tabela 2.1: Complexidade dos problemas tratados

Algumas abordagens para atacar o problema de dimensionamento de lotes nos casos que são *NP-difícil* serão mostradas no próximo capítulo.

Capítulo 3

Métodos de resolução do problema

Vimos no capítulo anterior que, quando consideramos restrições na capacidade de produção, de maneira geral, o problema é *NP-difícil*.

Neste capítulo, apresentaremos algumas reformulações do problema e algumas abordagens que são utilizadas para resolvê-lo nos casos em que o problema é *NP-difícil*.

3.1 Formulações do problema *LS-I*

Nesta seção vamos, primeiramente apresentar algumas definições que serão úteis para o resto do capítulo.

Consideremos a seguinte formulação geral de um *MIP*

$$\min \sum_{i=1}^{n} \alpha_i x_i + \sum_{j=1}^{m} \beta_j y_j \tag{3.1a}$$

sujeito a: (3.1b)

$$(x, y) \in X \tag{3.1c}$$

na qual X é o conjunto de pontos viáveis usualmente da forma $X = P \bigcap Y$. Onde $P \subset \mathbb{R}^{n+m}$ é um poliedro delimitado por um conjunto de restrições lineares utilizados na formulação e

$$Y = \{(x, y) : x \in \mathbb{R}^n, y \in \mathbb{Z}^m\}. \tag{3.2}$$

Uma formulação com o conjunto de restrições $P' \bigcap Y$ é considerada *equivalente* a uma formulação com o conjunto de restrições $P \bigcap Y$ se $P \nsubseteq P'$ e $P' \nsubseteq P$. Já uma formulação com o conjunto de restrições X' é dita *melhor* do que outra com o conjunto de restrições X se $P' \subset P$ e $X' = X$.

A formulação *ideal* para o problema é a combinação convexa dos pontos de X, que denotaremos a partir de agora por $conv(X)$, pois, para qualquer formulação com o conjunto de com restrições $P \bigcap Y$, temos que $conv(X) \subseteq P$.

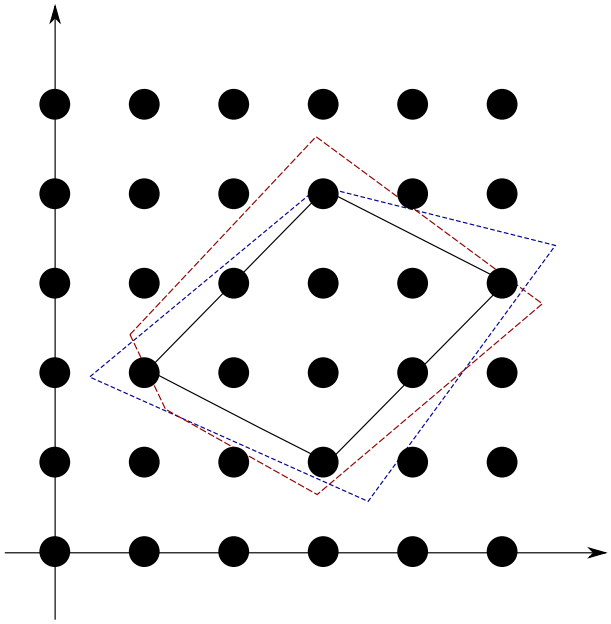


Figura 3.1: Representação de diversas formulações. As linhas tracejadas são formulações equivalentes e a linha cheia representa uma formulação ideal.

Consideremos duas formulações de um *MIP* distintas

$$\min \sum_{i=1}^{n} \alpha_i x_i + \sum_{j=1}^{n} \beta_j y_j \tag{3.3a}$$

$$\textbf{sujeito a:} \quad (x, y) \in X, \tag{3.3b}$$

e

$$\min \sum_{i=1}^{n} \gamma_i x_i + \sum_{j=1}^{n} \delta_j x_j \tag{3.4a}$$

$$\textbf{sujeito a:} \quad (x, y) \in X', \tag{3.4b}$$

dizemos que a formulação (3.4) é uma *relaxação* da formulação (3.3) se $X \subseteq X'$ e $\sum_{i=1}^{n} \gamma_i x_i + \sum_{j=1}^{m} \delta_j x_j \leq \sum_{i=1}^{n} \alpha_i x_i + \sum_{j=1}^{m} \beta_j x_j$.

Uma *relaxação linear* de um problema é uma formulação de um *MIP* na qual as restrições de algumas variáveis devam ter valores inteiros são removidas, tornando o *MIP* em um programa linear.

A grande vantagem de se ter uma formulação *ideal* é que, nesse caso específico, qualquer solução ótima da uma *relaxação linear* que seja um vértice é uma solução ótima do *MIP* original.

Conhecer apenas uma formulação ideal de um subconjunto das restrições já é interessante pois, para qualquer $X = X_1 \bigcap X_2$ temos que $X \supseteq conv(X_1) \bigcap X_2$ e, portanto, conseguimos obter uma formulação *melhor* para o problema.

Com esse fato em mãos, vamos apresentar algumas reformulações *ideais* do problema *LS-I*, que nos levará a uma formulação melhor dos demais problemas.

3.1.1 Estendendo a formulação original

Uma das formas de se obter uma formulação ideal é partir de uma formulação já existente e estendê-la, isto é, acrescentar um conjunto de restrições de maneira tal que a formulação estendida seja ideal.

A seguir vamos mostrar a validade de um conjunto de restrições que quando adicionadas à formulação já apresentada do problema *LS-I*, nos dá uma formulação ideal.

Para isso, utilizaremos o seguinte lema:

Lema 3.1. A desigualdade

$$x_t \leq D_{t,l} y_t + s_l, \quad \text{para } 1 \leq t, \leq l \leq n \tag{3.5}$$

vale sempre para o problema *LS-I*.

Demonstração. Pela conservação de fluxo, temos que a seguinte igualdade vale:

$$s_l = \sum_{i=t}^{l} x_i + \sum_{i=t-1}^{l-1} s_i - D_{t,l} \quad \text{para } 1 \leq t \leq l \leq n.$$

Essa mesma igualdade pode ser reescrita da seguinte forma:

$$x_t + \sum_{i=t+1}^{l} x_i + \sum_{i=t-1}^{l-1} s_i = D_{t,l} + s_l \quad \text{para } 1 \leq t \leq l \leq n.$$

Como $x_i, s_i \geq 0$ para $i \in \{1, \ldots, n\}$, temos que

$$x_t \leq D_{t,l} + s_l \quad \text{para } 1 \leq t \leq l \leq n.$$

Dado que a variável $y_t = 1$ se $x_t > 0$ e $y_t = 0$, caso contrário, pela desigualdade acima e pelo fato

de termos sempre $s_l \geq 0$ a proposição é sempre válida. □

Proposição 3.1. Sejam l, tal que $1 \leq l \leq n$, e S um conjunto tal que $S \subseteq \{1, \ldots, l\}$. Para o problema *LS-I* a desigualdade

$$\sum_{i \in S} x_i \leq \sum_{i \in S} D_{i,l} y_i + s_l \tag{3.6}$$

é sempre válida.

A demonstração da proposição é uma aplicação imediata do lema 3.1. As desigualdades (3.6) são comumente denominadas por *desigualdades-(l,S)*. *Barany et al.* [BRW84b] mostraram que se acrescentarmos o conjunto de desigualdades (3.6) à formulação original, teremos uma formulação ideal do problema *LS-I*.

O grande problema dessa reformulação é que existem $2^n - 1$ subconjuntos de $\{1, \ldots, n\}$ e, portanto, $2^n - 1$ restrições devem ser adicionadas ao problema original. Isso torna essa formulação inviável do ponto de vista prático, se tentarmos resolver o *MIP* diretamente.

Contudo, como foi mostrado em [GLS81], se existe um algoritmo polinomial capaz de dizer se uma solução respeita um conjunto exponencial de restrições e que, caso não respeite, devolve uma dessas restrições que não é respeitada, então o problema pode ser resolvido em tempo polinomial. Em [BRW84a] foi mostrado um algoritmo de separação polinomial para as *desigualdades-(l,S)*, que apresentaremos brevemente a seguir.

Primeiramente, vamos reescrever as desigualdades (3.6). Pela conservação de fluxo, sabemos que

$$s_l = \sum_{i \in L} x_i - D_{1,l} \quad \text{para } L = \{1, \ldots, l\} \text{ e } l = 1, \ldots, n. \tag{3.7}$$

Substituindo em (3.6) temos que

$$\sum_{i\in S} x_i \leq \sum_{i\in S} D_{i,l}y_i + \sum_{i\in L} x_i - D_{1,l} \quad \Rightarrow \tag{3.8}$$

$$\sum_{i\in L\setminus S} x_i + \sum_{i\in S} D_{i,l}y_i \geq D_{1,l}. \tag{3.9}$$

Consideremos uma solução (x^*, y^*) e observemos que a seguinte desigualdade vale:

$$\sum_{i\in L} \min(x_i^*, D_{i,l}y_i^*) \leq \sum_{i\in L\setminus S} x_i^* + \sum_{i\in S} D_{i,l}y_i^*. \tag{3.10}$$

Construindo um conjunto $S' = \{i \in L : x_i^* > D_{i,l}y_i^*\}$ temos que

$$\sum_{i\in L} \min(x_i^*, D_{i,l}y_i^*) = \sum_{i\in L\setminus S'} x_i^* + \sum_{i\in S'} D_{i,l}y_i^* \leq \sum_{i\in L\setminus S} x_i^* + \sum_{i\in S} D_{i,l}y_i^*. \tag{3.11}$$

Observando que se

$$\sum_{i\in L\setminus S'} x_i^* + \sum_{i\in S'} D_{i,l}y_i^* \geq D_{1,l}$$

então, para todo conjunto $S \subseteq L$, as *desigualdades-(l, S)* valem, por (3.11) e (3.9). Caso contrário, os conjuntos L e S' determinam um plano que corta a solução (x^*, y^*) para fora do conjunto de pontos viáveis.

Apresentamos agora o algoritmo de separação, que utiliza o que acabamos de descrever.

Algoritmo *Separação*$(x^*,\ y^*)$

01 $Cortes \leftarrow \emptyset$

01 **para** $l = 1, \ldots, n$ **faça:**

02 $opt_l \leftarrow 0$

03 **para** $j = 1, \ldots, l$ **faça:**

04 $opt_l \leftarrow opt_l + \min(x_j^*, D_{j,l} y_j^*)$

05 **se** $opt_l < D_{1,l}$ **então**

06 $S \leftarrow \emptyset$

07 **para** $j = 1, \ldots, l$ **faça:**

08 **se** $x_j^* < D_{j,l} y_j^*$ **então**

09 $S \leftarrow S \bigcup \{j\}$

10 $Cortes \leftarrow Cortes \bigcup \{(l, S)\}$

11 devolva $Cortes$

Foi mostrado tanto em [PW94] como em [PW01] que para o caso específico em os custos são tais que $p_1 \geq p_2 \geq \ldots \geq p_n$ (condição de *Wagner-Within*) apenas $O(n^2)$ *desigualdades-(l,S)* precisam ser adicionadas para se obter uma formulação estendida ideal.

3.1.2 Formulação como um problema de caminho mais curto

Vimos nas seção 2.1 que o problema *LS-I* pode ser interpretado como um problema de caminho mais curto, considerando o *princípio de decomposição do estoque*, de acordo com o qual, existe um planejamento ótimo em que o horizonte de planejamento pode ser quebrado em *intervalos de regeneração*.

A formulação como *MIP* da interpretação do problema *LS-I* como um problema de caminho mais curto se deve a Evans [Eva85]. Outras reformulações relacionadas podem ser encontradas em [EM87].

Quando interpretamos o problema *LS-I* como um problema de caminho mais curto, adotamos que cada aresta (i, j) do grafo representa um intervalo de regeneração $[i + 1, j]$, conforme ilustra a figura 2.2.

Para reformularmos o problema de caminho mais curto como um *MIP*, o interpretaremos como um problema de fluxo de custo mínimo em uma rede, no qual o vértice n tem demanda de uma unidade de fluxo e o vértice 0 é uma fonte que cederá também uma unidade de fluxo. Denotaremos por $Z_{i,j}$ o fluxo entre um arco (i, j), conforme ilustra a figura 3.2.

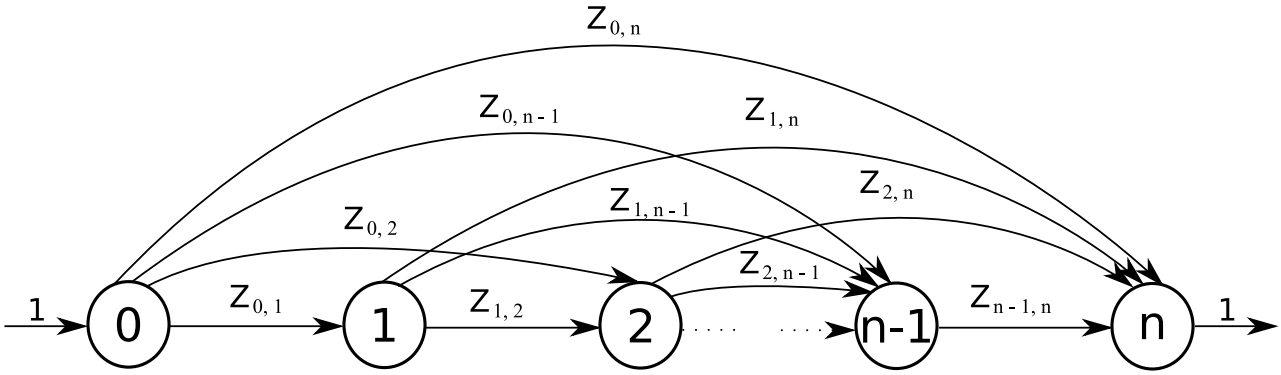


Figura 3.2: O problema de caminho mais curto *LS-I* como um problema de fluxo de custo mínimo.

A formulação desse problema de fluxo de custo mínimo como um *MIP* será apresentada a seguir:

$$\min \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} f_t y_t \tag{3.12a}$$

sujeito a: (3.12b)

$$\sum_{t=1}^{n} Z_{0,t} = 1, \tag{3.12c}$$

$$\sum_{i=0}^{t-1} Z_{i,t} = \sum_{i=t+1}^{n} Z_{t,i} \qquad \text{para } 1 \leq t \leq n, \tag{3.12d}$$

$$\sum_{i=0}^{n} Z_{i,n} = 1, \tag{3.12e}$$

$$\sum_{i=t}^{n} Z_{t-1,i} \leq y_t \qquad \text{para } 1 \leq t \leq n, \tag{3.12f}$$

$$\sum_{i=t}^{n} D_{t,i} Z_{t-1,i} = x_t \qquad \text{para } 1 \leq t \leq n, \tag{3.12g}$$

$$Z_{i,j} \in \mathbb{R}_+, \text{ para } 0 \leq i < j \leq n, \tag{3.12h}$$

$$x_t \in \mathbb{R}_+, y_t \in \{0,1\}, \text{ para } 0 \leq t \leq n. \tag{3.12i}$$

As restrições (3.12c)-(3.12e) garantem a conservação do fluxo, a restrição (3.12f) garante que $y_t = 1$ sempre que for produzido no instante t, para que o custo fixo seja computado, e a restrição (3.12g) reassocia o que foi produzido com a variável original do problema, x_t.

Pode ser mostrado que a matriz de restrições da formulação do problema acima é *totalmente unimodular* e, portanto, em toda solução básica do *MIP* (3.12), temos que ou $Z_{i,j} = 0$ ou $Z_{i,j} = 1$. Em particular, em uma solução ótima, todo $Z_{i,j}$ que pertence ao caminho mais curto será igual a 1 e os demais serão iguais a zero.

Esse fato nos traz uma informação importante. A de que a *relaxação linear* do problema (3.12) mantém y_t inteiro e, portanto, é uma solução para o *MIP*.

3.1.3 Reformulação baseada no problema de *Facility Location*

Uma reformulação importante do problema *LS-I* é baseada em um problema conhecido como *Uncapacitated Facility Location*(*UFL*), que descreveremos brevemente, a seguir.

Consideremos que temos M fornecedores de produtos e N clientes. Cada cliente j tem uma demanda d_j que precisa ser atendida e cada fornecedor é capaz de fornecer uma quantidade ilimitada de itens. O custo de se transportar um item do fornecedor i até o cliente j é $c_{i,j}$. Cada fornecedor i, caso esteja operando, tem um custo fixo de operação o_i. O objetivo do problema é atender às demandas dos clientes, determinando quanto cada fornecedor entregará para cada cliente, com o menor custo possível.

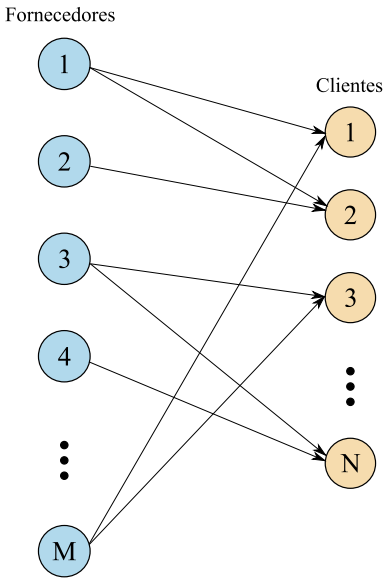


Figura 3.3: O conjunto de fornecedores e seus respectivos clientes no problema *UFL*.

Para modelarmos esse problema como um *MIP*, utilizaremos variáveis $e_{i,j}$ que representarão quantas unidades serão entregues ao cliente j pelo fornecedor i, variáveis binárias z_i que indicam se o fornecedor i vai ser utilizado na solução do problema ou não e denotaremos por $fornece(j)$ o

conjunto de fornecedores que podem entregar um produto ao cliente j. Com isso uma modelagem possível é a que segue:

$$\min \sum_{j=1}^{N} \sum_{i \in fornece(j)} c_{i,j} e_{i,j} + \sum_{i=1}^{M} o_i z_i \tag{3.13a}$$

sujeito a: (3.13b)

$$\sum_{i \in fornece(j)} e_{i,j} = d_j \qquad \text{para } 1 \leq j \leq M, \tag{3.13c}$$

$$e_{i,j} \leq d_j z_i \qquad \text{para } 1 \leq j \leq N, i \in fornece(j), \tag{3.13d}$$

$$e_{i,j} \in \mathbb{R}_+ \qquad \text{para } 1 \leq i \leq M, 1 \leq j \leq N, \tag{3.13e}$$

$$z_i \in \{0,1\} \text{ para } 1 \leq i \leq M, \tag{3.13f}$$

na qual a restrição (3.13c) garante que as demandas dos clientes serão atendidas e a restrição (3.13d) garante que se alguma entrega for realizada por um fornecedor então o custo de operação será computado.

Vamos mostrar agora que o problema *LS-I* pode ser interpretado como um problema *UFL*. Para isso, peguemos uma instância do problema *LS-I*. Consideremos agora os períodos $1, \ldots, n$ do horizonte de planejamento tanto como fornecedores como clientes. Um item entregue ao cliente j pelo fornecedor i, nesse caso significaria que um item foi produzido no período i para atender à demanda do período j.

Denotando por $x_{i,j}$ a variável que representa a quantidade de itens produzidos no período i para atender à demanda do período j. Atribuindo os valores dados no problema *LS-I*, da seguinte forma

- $fornece(t) = \{1, \ldots, t\}$, para $1 \leq t \leq n$;
- $c_{i,j} = p_i$, para $1 \leq i \leq j \leq n$;

- $e_{i,j} = x_{i,j}$, para $1 \leq i \leq j \leq n$;
- $z_t = y_t$, para $1 \leq t \leq n$;
- $d_t = D_t$, para $1 \leq t \leq n$;
- $o_t = f_t$, para $1 \leq t \leq n$;

e observando que $\sum_{t=1}^{n}\sum_{i=1}^{t} p_t x_{i,t} = \sum_{t=1}^{n} p_t \sum_{i=t}^{n} x_{t,i}$ e que podemos reassociar as variáveis originais do problema x_t através da igualdade $\sum_{i=t}^{n} x_{t,i} = x_t$, obtemos a seguinte formulação:

$$\min \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} f_t y_t \tag{3.14a}$$

sujeito a: (3.14b)

$$\sum_{i=1}^{t} x_{i,t} = D_t \qquad \text{para } 1 \leq t \leq n, \tag{3.14c}$$

$$x_{i,t} \leq D_t y_i \qquad \text{para } 1 \leq i \leq t \leq n, \tag{3.14d}$$

$$\sum_{t=i}^{n} x_{i,t} = x_i \qquad \text{para } 1 \leq i \leq n, \tag{3.14e}$$

$$x_{i,j} \in \mathbb{R}_+ \qquad \text{para } 1 \leq i \leq j \leq n, \tag{3.14f}$$

$$y_t \in \{0,1\} \text{ para } 0 \leq t \leq n. \tag{3.14g}$$

Assim como a *relaxação linear* da formulação 3.12 tem uma solução básica ótima na qual as variáveis y_t assumem valores inteiros, *Krarup e Bilde* [KB77] provaram que a matriz de restrições é *totalmente unimodular* e , portanto, a relaxação da formulação 3.13 também tem uma solução básica ótima onde as variáveis z_t assumem valores inteiros e, portanto é uma formulação compacta do problema. Como a formulação (3.14) é uma formulação do *LS-I* como um problema *UFL*, a formulação (3.14) também é uma formulação compacta para o problema *LS-I*.

3.2 Abordagens para resolução para os problemas difíceis

3.2.1 Branch-and-Bound

O método de *Branch-and-Bound* foi proposto em 1960, em [LD60] e serve de base ou trabalha em conjunto com diversos outros métodos de solução de *MIP*'s. Vamos mostrar a sua estrutura geral.

Se uma solução básica ótima da *relaxação linear* do *MIP* for solução do problema, então claramente ela será uma solução ótima. Porém, de maneira geral, em problemas mais complexos, a solução ótima do problema relaxado não será uma solução viável no problema original. Numa abordagem ingênua poderíamos querer arredondar a solução relaxada para termos uma solução inteira com um valor próximo do ótimo, cujo arredondamento nos levaria a uma solução inviável caso a solução do problema relaxado fosse o ponto (a) ou o ponto (b), mostrado no exemplo hipotético de solução para esse tipo de problema, apresentado na figura 3.4.

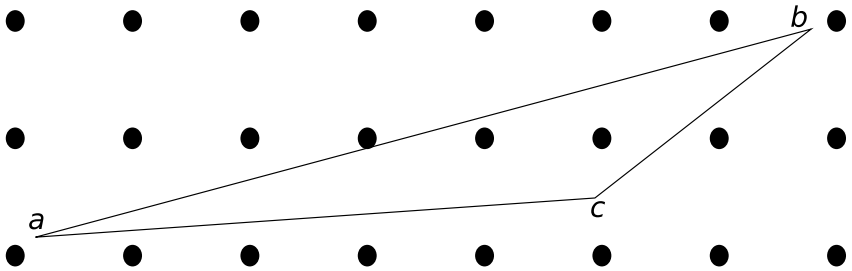


Figura 3.4: Exemplo de arredondamento para problema relaxado, levando a uma solução inviável.

Utilizando o fato de sabermos calcular eficientemente soluções de programas lineares e de que, se a solução do problema relaxado respeitar as condições de integralidade, temos a solução do problema original, podemos aplicar a seguinte estratégia de *divisão e conquista* para obtermos a solução do problema. Denotemos por $(x^*, y^*) \in \mathbb{R}^{n+m}$ uma solução básica ótima do problema relaxado. Se todas as coordenadas y_i^* são inteiras, então temos a solução ótima do problema. Senão, podemos

particionar o conjunto de restrições em duas partes, adicionando

$$y_i \leq \lfloor y_i^* \rfloor \text{ ou} \tag{3.15}$$

$$y_i \geq \lfloor y_i^* \rfloor + 1, \tag{3.16}$$

nos levando aos seguintes subproblemas:

(*Subproblema* 1):

$$\min \sum_{i=1}^{n} \alpha_i x_i + \sum_{j=1}^{m} \beta_j y_j \tag{3.17a}$$

$$\textbf{sujeito a:} \tag{3.17b}$$

$$(x, y) \in X, \tag{3.17c}$$

$$y_i \leq \lfloor y_i^* \rfloor. \tag{3.17d}$$

(*Subproblema* 2):

$$\min \sum_{i=1}^{n} \alpha_i x_i + \sum_{j=1}^{m} \beta_j y_j \tag{3.18a}$$

$$\textbf{sujeito a:} \tag{3.18b}$$

$$(x, y) \in X, \tag{3.18c}$$

$$y_i \geq \lfloor y_i^* \rfloor + 1. \tag{3.18d}$$

É fácil ver que a solução do problema original será a melhor solução entre os dois subproblemas. Para cada subproblema, podemos prosseguir da mesma maneira.

Esse processo de bissecção do espaço pode ser entendido como o processo de *ramificação* de uma árvore.

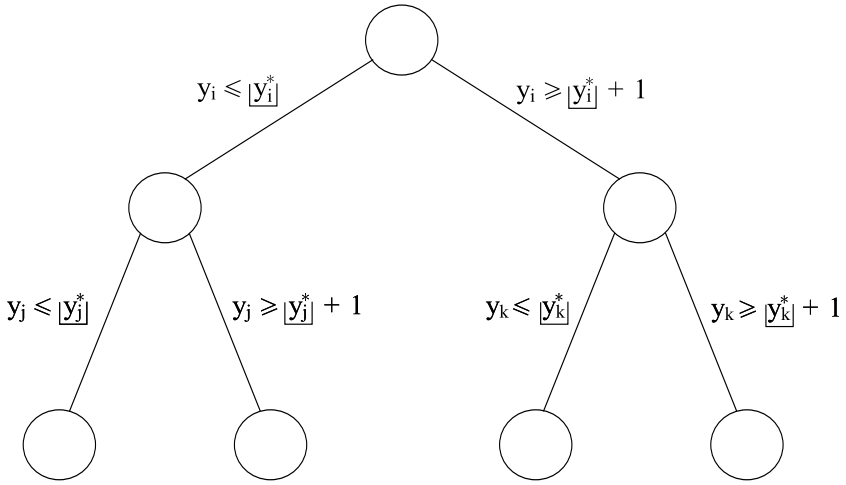


Figura 3.5: Figura que ilustra o processo de bissecção do espaço através de sucessivas divisões.

Como queremos sempre executar o menor número possível de operações para obtermos uma solução ótima, uma pergunta natural é saber se em um dado nó da árvore é realmente necessário continuar com o processo de ramificação ou se é possível podá-la.

Para podermos podar um determinado nó, temos que ter certeza que ele não contém uma solução ótima. Suponhamos que conhecemos um *limitante* superior $\mathcal{UB}$ e um limitante inferior $\mathcal{LB}$ para o problema. Vamos denotar a solução do problema relaxado no nó por (x', y'). Então temos que se

$$\sum_{i=1}^{n} \alpha_i x_i' + \sum_{j=1}^{m} \beta_j y_j' > \mathcal{UB} \tag{3.19}$$

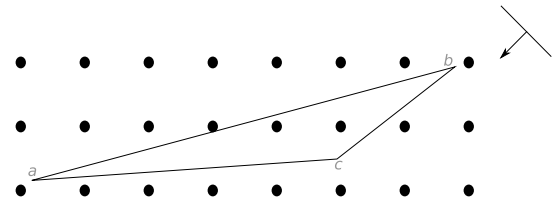
nenhuma solução ótima virá daquele nó.

Notemos que toda solução relaxada contendo um subconjunto das restrições do problema que está sendo analisado, serve como um limitante inferior para o problema e qualquer solução para o problema que respeite as restrições de integralidade serve como um limitante superior para a solução ótima.

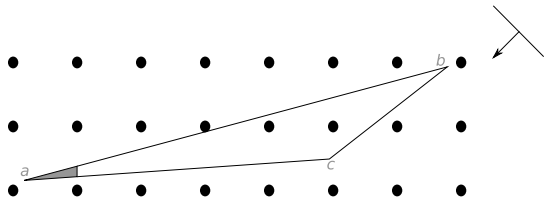
Com um limitante inferior e superior, podemos determinar um erro relativo de uma solução viável obtida até um determinado instante, pois sabemos que a porcentagem de erro entre a solução ótima e a solução obtida é inferior a $\epsilon_{rel} = \frac{\mathcal{UB}-\mathcal{LB}}{\mathcal{UB}}$.

Uma solução ótima é obtida quando mais nenhuma ramificação é possível e temos, considerando o menor limitante superior obtido e o maior limitante inferior, que $\epsilon_{rel} = 0$.

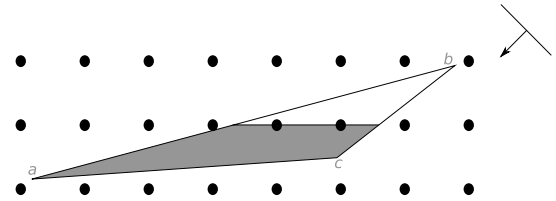
A seguir, a figura 3.6 mostra o funcionamento do método *Branch-and-Bound*.



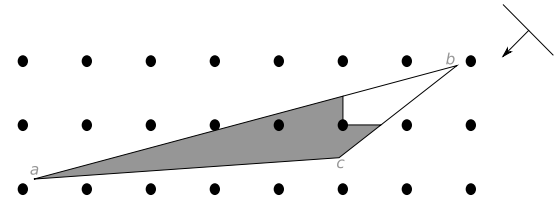
(a) O ponto a é a solução do problema relaxado, e suas coordenadas não são inteiras. A divisão do espaço será realizada.



(b) A área sombreada é a área que não mais pertence ao poliedro que está sendo considerado. Aqui, na solução obtida, ainda há coordenadas não inteiras, portanto, o poliedro será particionado novamente.



(c) Novamente, a solução não é inteira e o poliedro será novamente particionado.



(d) A solução do problema relaxado é inteira, e portanto uma solução viável.

Figura 3.6: Exemplo de funcionamento do método *Branch-and-Bound*.

Método de Branch-and-Bound aplicado ao *LS-C*

Conforme acabamos de mostrar, o método de *Branch-and-Bound* gera enumeração dos pontos inteiros viáveis, podando os que não tem um custo suficientemente bom ou que não vão gerar soluções viáveis.

Utilizando a interpretação do problema *LS-C* como um caminho mais curto, foi proposto em

[LY94] uma forma de se resolver, calculando o custo de cada intervalo de regeneração $[i, j]$ como um problema independente.

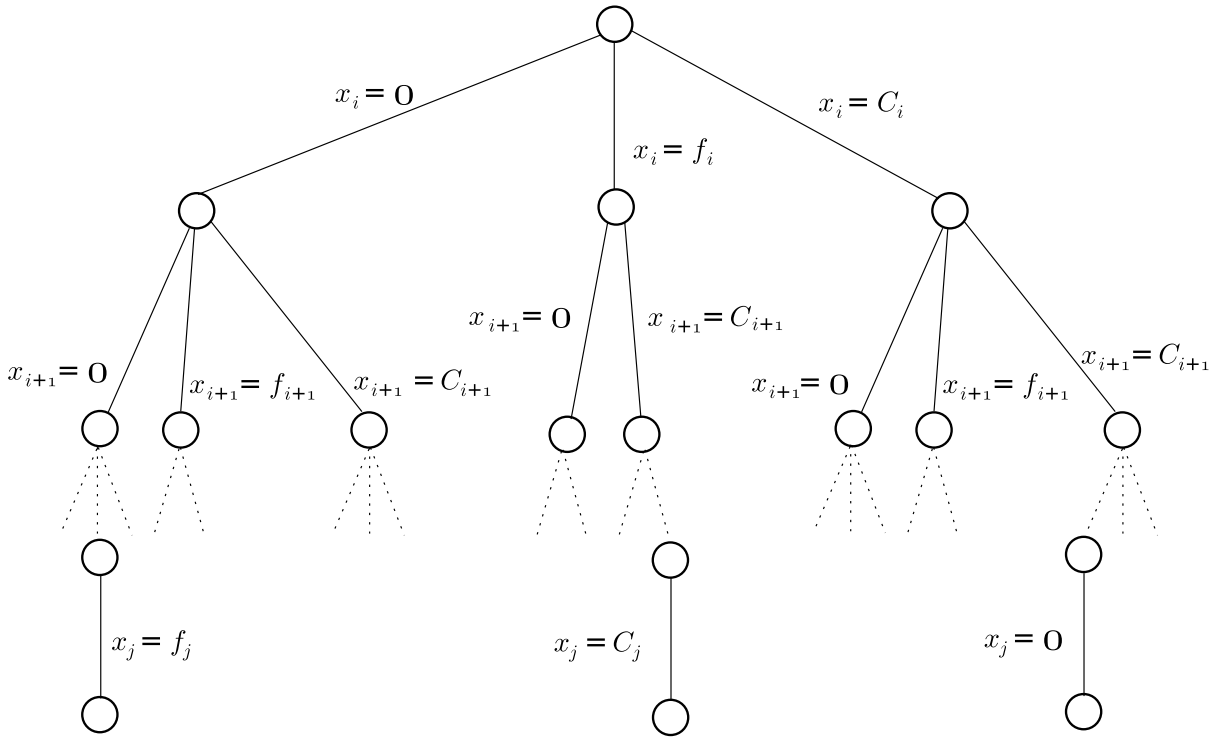


Figura 3.7: Exemplo de árvore de enumeração de possíveis soluções de um intervalo de regeneração $[i, j]$.

O processo de ramificação é feito da seguinte forma: consideremos os períodos $t \in [i, j]$ em ordem crescente. Pela proposição (2.7), existe uma solução ótima em que a quantidade produzida no período x_t é 0, fracionária em que a fração da capacidade total a ser produzida depende de todas as escolhas de planejamento dentro do intervalo, pois a capacidade varia de período para período, ou a plena capacidade, C_t. Como a proposição (2.7) nos diz que existe apenas um período de produção fracionária, uma vez que o período é escolhido, sobram apenas duas opções para os valores x_t futuros. A figura 3.7 ilustra o processo de ramificação da árvore.

O processo de poda da árvore consiste basicamente em verificar se a solução parcial obtida até um determinado período implica que as demandas não serão atendidas ou que a propriedade do intervalo $[i, l]$ ser um intervalo de regeneração será violada.

3.2.2 Relaxação Lagrangeana

Vimos, no método de *Branch-and-Bound*, a importância de se obter limitantes superiores e inferiores para o problema, pois em posse deles as podas podem ser realizadas e, possivelmente, um número menor de nós precisa ser processado.

Uma das formas de se obter bons limitantes inferiores é através do método *relaxação lagrangeana*. A aplicação do método para programas inteiros de forma geral foi apresentada em [Geo74] embora tenha sido anteriormente utilizada para resolver o problema do *caixeiro viajante* em [HK70].

Apresentaremos o método a seguir e exemplificaremos com a aplicação ao problema *LS-C*.

Consideremos a seguinte formulação geral de um *MIP*:

$$\min \sum_{i=1}^{n} \alpha_i z_i \tag{3.20a}$$

$$\textbf{sujeito a:} \quad Az \leq b, \tag{3.20b}$$

$$Dz \leq d, \tag{3.20c}$$

$$z_i \in \mathbb{Z}, \forall i \in I, \tag{3.20d}$$

$$z_i \in \mathbb{R}, \forall i \notin I \tag{3.20e}$$

onde o conjunto de restrições $Az \leq b$ é considerado um conjunto de restrições *fáceis*, isto é, que com a presença delas o problema ainda é fácil de resolver, $Dz \leq d$ é considerado um conjunto de restrições difíceis e I é um conjunto de índices que identificam as variáveis inteiras. Nos problemas de

dimensionamento de lotes, conforme vimos no capítulo anterior, as restrições que tornam o problema *NP-difícil* são as restrições

$$x_t - c_t y_t \leq 0 \quad \text{para } 1 \leq t \leq n. \tag{3.21}$$

A função lagrangeana de uma formulação geral de um *MIP* é a que segue:

$$\mathcal{L}(z, \lambda, \mu) = \alpha^{\mathrm{T}} z + \lambda^{\mathrm{T}}(Dz - d) + \mu^{\mathrm{T}}(Az - b). \tag{3.22}$$

Os vetores λ e μ, maiores ou iguais a zero, podem ser entendidos como o custo de se violar uma restrição ou um desconto por respeitá-la. Portanto, para λ e μ maiores que zero, o problema

$$LGR(\lambda, \mu) = \begin{cases} & \min \mathcal{L}(z, \lambda, \mu) \\ \textbf{sujeito a:} & z_i \in \mathbb{Z}, \forall i \in I \\ & z_i \in \mathbb{R}, \forall i \notin I \end{cases} \tag{3.23}$$

é uma relaxação para o *MIP* mais geral, pois para qualquer solução viável de (3.20) temos que, $\mathcal{L}(z, \lambda, \mu) \leq \alpha^{\mathrm{T}} z$. Além disso, o conjunto de restrições é um subconjunto das restrições da formulação geral do *MIP*.

Para diferentes valores de λ e μ, teremos diferentes valores para o problema de minimização da função lagrangeana. Como queremos obter o melhor limitante inferior possível (no caso, o maior), o problema no qual realmente estamos interessados é o seguinte:

$$\max LGR(\lambda, \mu) \tag{3.24a}$$

$$\textbf{sujeito a:} \quad \lambda \geq 0, \tag{3.24b}$$

$$\mu \geq 0. \tag{3.24c}$$

No método de *relaxação lagrangeana*, nós não utilizamos a função lagrangeana e sim uma simplificação da função, onde passamos apenas as restrições difíceis para a função objetivo. A versão relaxada do problema fica assim:

$$RLX(\lambda) = \begin{cases} & \min \bar{\mathcal{L}}(z,\lambda) = \alpha^{\mathrm{T}} z + \lambda^{\mathrm{T}}(Dz - d) \\ \textbf{sujeito a:} & Az \leq b, \\ & z_i \in \mathbb{Z}, \forall i \in I. \end{cases} \tag{3.25}$$

e o problema de obtenção do melhor limitante inferior se torna

$$MLB = \begin{cases} & \max RLX(\lambda) \\ \textbf{sujeito a:} & \lambda \geq 0. \end{cases} \tag{3.26}$$

Vamos agora mostrar a função $RLX(\lambda)$ para o problema *LS-C*, para que possamos calcular o melhor limitante inferior da relaxação lagrangeana.

$$RLX(\lambda) = \begin{cases} & \min \sum_{t=1}^{n}(p_t + \lambda_t)x_t + \sum_{t=1}^{n} h_t s_t + \sum_{t=1}^{n}(f_t - \lambda_t c_t)y_t \\ \textbf{sujeito a:} & s_{t-1} + x_t = s_t + D_t, \text{ para } t = 1, \ldots, n \\ & \sum_{t=1}^{n} x_t = D_{1,n} \\ & \sum_{t=1}^{j} x_t \geq D_{1,j}, \text{ para } j = 1, \ldots, n \\ & x_t \leq y_t D_{1,n}, \text{ para } t = 1, \ldots, n \\ & x_t \geq 0, s_t \geq 0, \text{ para } t = 1, \ldots, n \\ & s_0 = s_n = 0 \\ & x_t \in \mathbb{R}, s_t \in \mathbb{R}, y_t \in \{0,1\}, \text{ para } t = 1, \ldots, n. \end{cases}$$

Observemos que para um λ fixo, podemos determinar previamente os valores $\bar{p}_t = p_t + \lambda_t$ e $\bar{f}_t =$

$f_t - \lambda_t c_t$, tornando $RLX(\lambda)$ uma instância de um problema *LS-I*, que sabemos calcular eficientemente.

Dado que fixamos um λ, o problema de minimização (3.25) é um problema fácil de se resolver, por hipótese. Com esse fato em mãos, vamos construir um método de solução de (3.26).

Para isso, apresentaremos a seguir, um fato bem conhecido de cálculo multivariado.

Proposição 3.2. Uma direção d é chamada de direção de acréscimo, ou de subida de uma função $f(\lambda) : \mathbb{R}^m \to \mathbb{R}$ diferenciável, se existe um escalar ρ positivo tal que $f(\lambda + \rho d) > f(\lambda)$. Para qualquer direção de subida vale que $\nabla f(\lambda)^{\mathrm{T}} d > 0$.

Demonstração. Para $\rho > 0$ suficientemente pequeno, a expansão de Taylor para $f(\lambda + \rho d)$ vale e nos diz que

$$f(\lambda + \rho d) \approx f(\lambda) + \rho \nabla f(\lambda)^{\mathrm{T}} d, \tag{3.27}$$

o que implica que

$$f(\lambda + \rho d) - f(\lambda) \approx \rho \nabla f(\lambda)^{\mathrm{T}} d, \tag{3.28}$$

logo só teremos $f(\lambda + \rho d) - f(\lambda)$ maior que zero se $\nabla f(\lambda)^{\mathrm{T}} d$ também for maior que zero. □

Como o problema (3.26) é um problema em que a variável λ deve sempre ser maior ou igual a zero, apenas direções de subida que mantenham os valores de $\lambda \geq 0$ são permitidos.

A proposição a seguir mostra a forma que uma direção d deve ter para que seja uma direção de subida e que mantenha $\lambda' = \lambda + d$ maior ou igual a zero.

Proposição 3.3. Consideremos $f(\lambda) : \mathbb{R}^m \to \mathbb{R}$ uma função diferenciável restrita à $\lambda \geq 0$. Então $d = \max\{0, \lambda + \rho \nabla f(\lambda)\} - \lambda$ em que $\rho > 0$ é uma direção de subida.

Demonstração. Pela proposição 3.2 uma direção é de subida se $\nabla f(\lambda)^{\mathrm{T}} d > 0$. Chamemos de $M \subseteq$

$\{1, \ldots, m\}$ o conjunto de índices tal que para todo $i \in M$

$$\lambda_i + \rho \frac{\partial f(\lambda)}{\partial \lambda_i} > 0. \tag{3.29}$$

Portanto, pela definição de d e M, temos que as componentes do vetor d assumem a seguinte forma:

$$d_i = \rho \frac{\partial f(\lambda)}{\partial \lambda_i}, \forall i \in M, \tag{3.30}$$

$$d_i = -\lambda_i, \forall i \notin M. \tag{3.31}$$

Logo, temos que

$$\nabla f(\lambda)^{\mathrm{T}} d = \sum_{i \in M} \rho \left(\frac{\partial f(\lambda)}{\partial \lambda_i} \right)^2 + \sum_{i \notin M} -\lambda_i \frac{\partial f(\lambda)}{\partial \lambda_i} \tag{3.32}$$

A somatória com $i \in M$ é claramente positiva. Analisemos então a parte onde $i \notin M$. Pela própria definição de M e pelo fato de $\lambda_i \geq 0$ e $\rho > 0$, temos que

$$\frac{\partial f(\lambda)}{\partial \lambda_i} \leq -\frac{\lambda_i}{\rho} \leq 0, \forall i \notin M, \tag{3.33}$$

e , portanto,

$$-\lambda_i \frac{\partial f(\lambda)}{\partial \lambda_i} \geq 0, \forall i \notin M, \tag{3.34}$$

o que implica que $\nabla f(\lambda)^{\mathrm{T}} d > 0$. □

Consideremos agora que fixamos um λ^k, calculamos (3.25) e obtemos um vetor z^k que minimiza $\bar{\mathcal{L}}(z, \lambda^k)$. Pela proposição 3.2 sabemos que se tomarmos uma direção d, da forma da proposição 3.3,

para um ρ suficientemente pequeno, existe um λ^{k+1} tal que

$$\lambda^{k+1} = \lambda^k + d \tag{3.35}$$

no qual $\bar{\mathcal{L}}(z^k, \lambda^{k+1}) \geq \bar{\mathcal{L}}(z^k, \lambda^k)$, sempre que a função for diferenciável. Pode ser mostrado que a função $\mathcal{L}$ não é diferenciável em todo o domínio, porém a direção que acabamos de mostrar ser de subida, no problema específico da função lagrangiana do programa linear, $\max\{0, \lambda^k + \rho_k(Dz^k - d)\}$, sempre pode ser calculado. Esse vetor, independente de ser uma direção de subida ou não, é denominado *subgradiente*.

Repetindo iterativamente a ideia que acabou de ser mostrada, foi apresentado em [HK70] o algoritmo a seguir:

Algoritmo *Subgradiente*

01 $\lambda \leftarrow \lambda^0$

02 $k \leftarrow 0$

03 **enquanto** $RLX(\lambda^k)$ não converge **faça:**

04 Obtenha uma solução z^k de $RLX(\lambda^k)$

05 Escolha um ρ_k de maneira adequada

06 $\lambda^{k+1} \leftarrow \max\{0, \lambda^k + \rho_k(Dz^k - d)\}$

07 $k \leftarrow k + 1$

Algo que vale a pena ser salientado é a simplicidade deste algoritmo, que no caso do problema *LS-C* consiste em resolver na linha *03* uma instância do problema *LS-I* e na linha *06* atualizamos cada coordenada λ_t^{k+1} da seguinte forma: $\lambda_t^{k+1} = \lambda_t^k + \rho_k(x_t^k - c_t y_t^k)$.

Porém, a escolha, na linha *05*, de um ρ_k válido e a convergência do algoritmo em si não são claras.

Os resultados mostrados em [Gof77], [HWC74], [GSW72] e [Sha79] com relação a escolha de um ρ_k válido e a convergência do método podem ser sumarizados pelo seguinte teorema:

Teorema 3.1. No algoritmo *Subgradiente*, com relação a escolha de ρ_k, as seguintes condições valem:

(i) Se $\sum_{k=0}^{\infty} \rho_k \to \infty$ e para $k \to \infty$ temos que $\rho_k \to 0$ então $RLX(\lambda^k) \to MLB$.

(ii) Se para algum $\gamma < 1$ temos que $\rho_k = \rho_0 \gamma^k$ então $RLX(\lambda^k) \to MLB$ se ρ_0 e γ forem suficientemente grandes.

(iii) Se $\mathcal{LB} \leq MLB$ e $\rho_k = \epsilon_k \dfrac{(\mathcal{LB} - RLX(\lambda^k))}{||Dz^k - d||^2}$ tal que $0 < \epsilon_k < 2$ e z^k solução de $RLX(\lambda^k)$ então ou $RLX(\lambda^k) \to \mathcal{LB}$ ou então o algoritmo encontrará um λ^k em um número finito de passos tal que $\mathcal{LB} \leq RLX(\lambda^k) \leq MLB$ vale.

A condição (i) implica que os ρ_k formam uma série divergente, o que na prática implica que o método demorará muito tempo para convergir. Portanto, usualmente se usa ou (ii) ou (iii). Em particular, aplicado ao problema de dimensionamento de lotes, a regra (iii) é usada em [TW85] e [BDKZ92].

Outro aspecto indeterminado é o ponto de partida do algoritmo, λ^0. Uma escolha comum é a utilização da solução do problema dual da relaxação linear do problema como λ^0.

No início da seção dissemos que o método de *Relaxação Lagrangeana* é utilizado para obtenção de bom limitante inferior. A proposição a seguir, nos mostra que dentro de certas condições a solução do problema relaxado pode nos dar a solução ótima do problema original.

Proposição 3.4. Seja (z^*, λ^*) uma solução ótima de (3.26) e, portanto $\lambda^* \leq 0$. Se z^* satisfizer as condições

(i) $Dz^* \leq d$,

(ii) Se $(Dz^*)_i = d_i$ então $\lambda_i^* < 0$

então z^* é uma solução ótima de (3.20).

Demonstração. Se (i) vale, então a solução z^* é viável. Se (ii) vale, então sabemos que $\bar{\mathcal{L}}(z^*, \lambda^*) = a^{\mathrm{T}} z^*$. Mas como (3.26) é uma relaxação, sabemos que $\bar{\mathcal{L}}(z, \lambda) \leq a^{\mathrm{T}} z$ para quaisquer λ e z válidos e, portanto, z^* é uma solução ótima para (3.20). □

Observamos que o caso do problema *LS-AC* é análogo e o problema $RLX(\lambda)$ é uma instância do problema *LS-A*. Apesar de termos mostrados métodos bastante eficientes para a solução do problema *LS-I*, como o método de *relaxação lagrangeana* é frequentemente utilizado em conjunto com o método de *Branch-and-Bound* e, portanto, se tem um sistema de resolução de programas lineares, é prático se utilizar das formulações ideais (3.12) ou (3.14) e resolver o programa linear para obter a solução dos problemas *fáceis*.

3.3 Considerações finais

Do mesmo modo que foi mostrado na seção 3.1 para o problema *LS-I* também existem reformulações para alguns outros casos. Algumas delas podem ser encontradas em [LMV89], [LMW00] e [MNS00].

Uma apresentação mais detalhada do método de *Branch-and-Bound* pode ser encontrado em [Wol98] e [PS98] e uma também uma boa apresentação do método de *relaxação lagrangeana* pode ser encontrada em [Wol98].

Outras abordagens para a solução dos problemas *LS-C* e *LS-AC* têm ganho espaço, em especial o desenvolvimento de algoritmos *pseudo-polinomiais* com bons resultados computacionais, conforme mostrado em [CHL94] e [SW98]. Além disso, um esquema de aproximação polinomial foi mostrado em [HW01].

Capítulo 4

Conclusão

Tratamos nesse trabalho apenas os problemas de dimensionamento de lotes mais básicos. Porém, diversas outras restrições podem aparecer quando estamos tratando do problema de *dimensionamento de lotes de itens únicos*. A seguir, citaremos algumas.

- *Presença de* backlog: é comum que esteja previsto em contrato que se possa realizar a entrega de pedidos após a data estipulada mediante pagamento de multa ou, equivalentemente, a um desconto. A quantidade de itens que já deveriam ter sido entregues mas ainda não foram, denominamos *backlog*. Os principais artigos que tratam deste tipo de restrição são [Zan69] e [PW88].

- *Contabilização de* start-up: na modelagem que apresentamos ao longo do trabalho, contabilizamos custos relativos à atividade produtiva, mas não foi incorporado à modelagem o impacto de se iniciar uma sequência de períodos de produção. Devido aos preparativos, no período em que a produção é iniciada há uma natural redução na capacidade produtiva. Como boas modelagens que incorporam essa redução na capacidade são difíceis de serem obtidas, usualmente quando se quer contabilizar o impacto do início da produção (ou *start-up*) se associa um custo

ao período em que se inicia a produção. Alguns dos artigos que tratam do custo de *start-up* são [KKK87], [Wol89] e [Van98].

- *Janelas de tempo:* quando se produz bens perecíveis, a presença de janelas de tempo aparece de maneira bastante natural.

  Consideremos o prazo de validade PV de um produto e o período e no qual um determinado pedido deve ser entregue. Para que o produto entregue não esteja estragado é evidente que ele tem que ser produzido dentro do intervalo (janela) de tempo $(e - PV, e]$. Como referência para o leitor interessado, citamos os artigos [LÇW01], [Wol06] e [Hwa07].

Outro aspecto interessante sobre o problema de dimensionamento de lotes de itens únicos é que eles podem aparecer como subproblemas em problemas mais complexos. Em especial o caso em que vários tipos de itens devem ser produzidos.

Para exemplificar, consideremos uma pequena confecção que produz apenas dois tipos de itens: calças e jaquetas. Suponhamos que a capacidade máxima de produção de calças por hora seja de 10 unidades e que a capacidade máxima de produção de jaquetas por hora seja de 7 unidades. Além disso, consideremos que a confecção funciona 8 horas por dia e que não se produz calças e jaquetas ao mesmo tempo. Neste caso específico, temos uma restrição na capacidade produtiva da confecção que engloba todos os itens que pode ser descrita da seguinte forma:

$$\frac{1}{10}\text{calças} + \frac{1}{7}jaquetas \leq 8. \tag{4.1}$$

Considerando problemas com a mesma estrutura do exemplo que acabamos de descrever com T tipos distintos de itens a serem produzidos, e usando a mesma notação para as variáveis de modelagem que usamos no restante do trabalho, mas agora diferenciando a qual item a variável se refere (Exemplo: a variável $x_{i,t}$ se refere a quantidade de itens produzidos do tipo i no período t),

podemos modelar o problema da seguinte forma:

$$\min \sum_{i=1}^{T}\sum_{t=1}^{n}(p_{i,t}x_{i,t} + h_{i,t}s_{i,t} + f_{i,t}y_{i,t}) \tag{4.2a}$$

$$\textbf{sujeito a:} \quad s_{i,t-1} + x_{i,t} = s_{i,t} + D_{i,t}, \text{ para } t = 1, \ldots, n \text{ e } i = 1, \ldots, T \tag{4.2b}$$

$$\sum_{i=1}^{T}\sum_{t=1}^{n} x_{i,t} = D_{i,1,n} \tag{4.2c}$$

$$\sum_{i=1}^{T}\sum_{t=1}^{j} x_{i,t} \geq D_{i,1,j}, \text{ para } j = 1, \ldots, n \tag{4.2d}$$

$$\sum_{i=1}^{T} \alpha_i x_{i,t} \leq c_t \sum_{i=1}^{T} y_{i,t}, \text{ para } t = 1, \ldots, n \tag{4.2e}$$

$$\sum_{i=1}^{T} y_{i,t} \leq 1, \text{ para } t = 1, \ldots, n \tag{4.2f}$$

$$x_{i,t} \geq 0, s_{i,t} \geq 0, \text{ para } t = 1, \ldots, n \tag{4.2g}$$

$$s_{i,0} = s_{i,n} = 0 \tag{4.2h}$$

$$x_{i,t} \in \mathbb{R}, s_{i,t} \in \mathbb{R}, y_{i,t} \in \{0, 1\}, \text{ para } t = 1, \ldots, n. \tag{4.2i}$$

Se, para resolvermos o problema, utilizarmos o método de relaxação lagrangeana e passarmos para a função objetivo as restrições (4.2e) e (4.2f), teremos T problemas independentes do tipo *LS-I* como problemas fáceis de se resolver.

Em aplicações práticas há diversas dificuldades além das que mostramos aqui e, por isso, frequentemente são utilizadas heurísticas sem maiores garantias com relação a qualidade da solução obtida. Porém, com o aumento do poder computacional e com modelos que exploram melhor a estrutura dos problemas, métodos que buscam soluções exatas têm ganho espaço. Para o leitor que esteja interessado em bom guia sobre modelagem de diversos tipos de problemas de planejamento de produção, e suas respectivas aplicações, uma boa referência é o livro [PW06]. Para uma visão geral e rápida

sobre o problema específico de itens únicos uma boa referência é [BDPNA06].

Referências Bibliográficas

[BDKZ92] H. C. Bahl, M. Diaby, M.H. Karwan, and S. Zionts. Capacitated lot-sizing and scheduling by lagrangean relaxation. *European Journal of Operational Research, 59:444-458*, 1992.

[BDPNA06] N. Brahimi, S. Dauzère-Perès, N. M. Najid, and A.Nordli. Single item lot sizing problems. *European Journal of Operational Research, 168:1-16*, 2006.

[BRW84a] I. Barany, T.J. Van Roy, and L.A. Wolsey. Strong formulations for multi-item capacitated lot-sizing. *Management Science, Vol 30: 1255-1261*, 1984.

[BRW84b] I. Barany, T.J. Van Roy, and L.A. Wolsey. Uncapacitated lot sizing: The convex hull of solutions. *Mathematical Programming, 22:32-43*, 1984.

[BY82] G. R. Bitran and H. H. Yanasse. Computational complexity of the capacitated lot size problem. *Management Science, Vol 28: 1174-1186*, 1982.

[CCPS] W. J. Cook, W. H. Cunningham, W. L. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley-Interscience.

[CHL94] H. D. Chen, D. Hearn, and C. Y. Lee. A new dynamic programming algorithm for the single item capacitated dynamic lot size model. *Journal of Global Optmization, 4:285-300*, 1994.

[CLRS] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition.

[EM87] G. D. Eppen and R. K. Martin. Solving multi-item capacitaded lot-sizing problems using variable redefinition. *Operations Research, 35:832-848*, 1987.

[Eva85] J. R. Evans. An efficient implementation of the wagner-within algorithm for dynamic lot-sizing. *Journal of Operations Management, 5:229-235*, 1985.

[FK71] M. Florian and M. Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science, Vol 18: 12-20*, 1971.

[FLK80] M. Florian, J. K. Lenstra, and A. H. G Rinnooy Kan. Deterministic prodution planning: Algorithms and complexity. *Management Science, Vol 26, No. 7: 669-679*, 1980.

[Geo74] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study 2, 82-114*, 1974.

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica, 1:169-197*, 1981.

[Gof77] J. L. Goffin. On the convergence rates of subgradient optimization methods. *Mathematical Programming, 13:329-347*, 1977.

[GSW72] G. A. Gorrey, J. F. Shapiro, and L.A. Wolsey. Relaxation methods for pure and mixed integer programming problems. *Management Science, 18:229-239*, 1972.

[Har13] F. W. Harris. How many parts to make at once. *Factory, the Magazine of Management, Vol 18: 135-136, 152*, 1913.

[HK70] M. Held and R. M. Karp. The traveling salesman problem and minimun spanning trees. *Operations Research, 18:1138-1162*, 1970.

[Hoe91] Stan Van Hoesel. *Models And Algorithms for single-item Lot Sizing Problems.* PhD thesis, Erasmus University - Rotterdam, 1991.

[HW96] C. P. M. Van Hoesel and A.P.M Wagelmans. An $o(t^3)$ algorithm for the economic lot-sizing problem with constant capacities. *Management Science, Vol 42, No. 1: 143-150*, 1996.

[HW01] C. P. Van Hoesel and A. P. Wagelmans. Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. *Mathematics of Operations Research, 26:339-357*, 2001.

[Hwa07] H. Hwang. Dynamic lot-sizing model with production time windows. *Naval Research Logistics, Vol 54: 692-701*, 2007.

[HWC74] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming, 6:62-88*, 1974.

[Jr64] A. F. Veinott Jr. Production planning with convex costs: a parametric study. *Management Science, Vol 10: 441-460*, 1964.

[KB77] J. Krarup and O. Bilde. Plant location, set covering and economic lot size: An o(mn)-algorithm for structured problems. *International Series of Numerical Mathematics, Vol 36:155-186*, 1977.

[KHW92] A. Kolen, S. Van Hoesel, and A. Wagelmans. Economic lot sizing: An o(n log n) algorithm that runs in linear time in the wagner-whitin case. *Operations Research, 40, Supp. No 1. S145-S157*, 1992.

[KKK87] U. S. Karmarkar, S. Kekre, and S. Kekre. The dynamic lot-sizing problem with startup and reservation costs. *Operations Research, 35: 389-398*, 1987.

[LD60] A.H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica, Vol 28, No. 3: 497-520*, 1960.

[LMV89] J. M. Leung, T. L. Magnanti, and R. Vachani. Facets and algorithms for capacitated lot sizing. *Mathematical Programming, 45:331-359*, 1989.

[LMW00] M. Loparic, H. Marchand, and L. A. Wolsey. Facets and algorithms for capacitated lot sizing. *Technical Report, 47, Catholique de Louvain, Center for Operations Research and Economics*, 2000.

[Lov73] S. F. Love. Bounded production and inventory models with piecewise concave costs. *Management Science, Vol 20: 313-318*, 1973.

[LY94] V. Lofti and Y.S Yoon. An algorithm for the single-item capacitated lot-sizing problem with concave production and holding costs. *The Journal of Operational Research Society, Vol 45: 934-941*, 1994.

[LÇW01] C. Lee, S. Çetinkaya, and A. P. M. Wagelmans. A dynamic lot-sizing model with demand time windows. *Management Science, 47:1384-1395*, 2001.

[MNS00] A. J. Miller, G. L. Nemhauser, and M. W. Savelsbergh. On the capacitated lot-sizing and continuous 0-1 knapsack polyhedra. *European Journal of Operational Research, 125:298-315*, 2000.

[PS98] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization; Algorithms and Complexity*. Dover Publications, 1998.

[PW88] Y. Pochet and L. A. Wolsey. Lot-size models with backlogging: Strong reformulations and cutting planes. *Mathematical Programming, 40:317-335*, 1988.

[PW94] Y. Pochet and L.A. Wolsey. Polyhedra for lot-sizing with wagner-within costs. *Mathematical Programming, 67:297-324*, 1994.

[PW01] O. Pereira and L.A. Wolsey. On the wagner-within lot-sizing polyhedron. *Mathematics of Operations Research, 26:591-600*, 2001.

[PW06] Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming (Springer Series in Operations Research and Financial Engineering)*. Springer-Verlag New York, Inc., 2006.

[Sha79] J. F. Shapiro. A survey of lagrangean techniques for discrete optimization. *Annals of Discrete Mathematics 15, 113-118*, 1979.

[SW98] D. X. Shaw and A. P. Wagelmans. An algorithm for single-item capacitated economic lot-sizing with piecewise linear production costs and general holding costs. *Management Science, Vol 44:831-838*, 1998.

[TW85] J. M. Thizzy and L. N. Van Wassenhove. Capacitated lot-sizing and scheduling by lagrangean relaxation. *AIIE Transactions, 17:64-74*, 1985.

[Van98] F. Vanderbeck. Lot-sizing with start-up times. *Management Science, 44:1409-1425*, 1998.

[Wol89] L. A. Wolsey. Uncapacitated lot-sizing problems with start-up costs. *Operations Research, 37:741-747*, 1989.

[Wol98] L. A. Wolsey. *Integer Programming.* Wiley-Interscience, 1998.

[Wol06] L. A. Wolsey. Lot-sizing with production and delivery time windows. *Mathematical Programming, Vol 107: 471-489*, 2006.

[WW58] H. M. Wagner and T. M. Within. Dynamic version of the economic lot size model. *Management Science, Vol 15: 89-96*, 1958.

[Zan68] W. I. Zangwill. Minimum concave cost flows in certain networks. *Management Science, Vol 14: 429-450*, 1968.

[Zan69] W. I. Zangwill. A backlogging model and a multi-echelon model of a dynamic economic lot size production system-a network approach. *Management Science, 15:506-527*, 1969.