Sistemas de apoio à decisão

CAP5-Modelos e Métodos de Optimização

Índice

- Introdução
- Componentes
  - Índices
  - Dados
  - Variáveis
  - Restrições
  - Função objectivo
- Exemplo
- Estrutura
- Tipos especiais
- Métodos de optimização
  - Introdução aos métodos heurísticos
  - Métodos de construção
  - Pesquisa local
  - Pesquisa tabu
  - Pesquisa por arrefecimento simulado
- Bibliografia

Objectivos de aprendizagem

- Identificar diferentes tipos de modelos de Programação Matemática;
- Descrever os elementos destes modelos, e conhecer algumas instâncias clássicas destes elementos;
- Identificar a estrutura de um modelo de Programação Matemática;
- Conhecer abordagens de resolução alternativas aos métodos exactos;
- Aplicar métodos de pesquisa local na resolução de problemas;
- Conhecer, descrever e aplicar diferentes meta-heurísticas baseadas em pesquisa local.

Introdução

- Designações equivalentes | Casos considerados nesta UC
  - Optimização,
  - Optimização matemática,
  - Optimização algébrica,
  - Programação matemática

- Ferramenta de base: o modelo matemático
  - » Representa o conjunto das relações (matemáticas) entre os vários elementos de um problema;
  - » Exemplos
    - » Relações tecnológicas;
    - » Leis físicas;
    - » Restrições comerciais;
    - » Relações lógicas;
    - » ...

» Estrutura genérica »»»»

$$\begin{array}{lrcl} \min & f(x) & & \\ \text{s. a} & g(x) & \geq & 0 \\ & x & \in & \mathbb{R}^n \end{array}$$

Introdução

- Modelos de optimização | Elementos fundamentais
  - Dados (parâmetros ou constantes do modelo)
    - Custos;
    - Procuras;
    - Capacidades; ...
  - Variáveis (contínuas, semi-contínuas, inteiras, binárias) ou variáveis de decisão
    - Quantidade a produzir de um produto;
    - Produtos a armazenar e quantidades associadas; ...
  - Restrições (equações, inequações)
    - Restrições de capacidade;
    - Níveis mínimos de qualidade; ...
  - Função objectivo
    - Minimização de custos;
    - Minimização de desperdícios
    - Maximização de lucros.

Introdução

- Formulação equivalente e classes de problemas
  (*Fonte: J. Kallrath e T. Maindl, 2006*)
  - O caso mais geral: Programação Inteira Mista Não-Linear;
  - *Input:*
    - *Vector aumentado (variáveis):* $\mathbf{x}_{\oplus}^{\mathrm{T}} = \mathbf{x}^{\mathrm{T}} \oplus \mathbf{y}^{\mathrm{T}}$

    com $\mathbf{x}^{\mathrm{T}} = (x_1, ..., x_{n_c})$ ***e*** $\mathbf{y}^{\mathrm{T}} = (y_1, ..., y_{n_d})$

    n_c variáveis contínuas, n_d variáveis discretas;

  - O problema

$$\min \left\{ f(\mathbf{x},\mathbf{y}) \left| \begin{array}{l} \mathbf{h}(\mathbf{x},\mathbf{y}) = 0 \;\; \mathbf{h} : X \times U \to \mathbb{R}^{n_e} \;\; \mathbf{x} \in X \subseteq \mathbb{R}^{n_c} \\ \mathbf{g}(\mathbf{x},\mathbf{y}) \geq 0, \mathbf{g} : X \times U \to \mathbb{R}^{n_i}, \mathbf{y} \in U \subseteq \mathbb{Z}^{n_d} \end{array} \right. \right\}$$

Introdução

- <u>Formulação equivalente</u> e classes de problemas
  (*Fonte: J. Kallrath e T. Maindl, 2006*)
  – O problema é não-linear se ***f***, ***g*** ou ***h*** for não-linear;
  – Qualquer vector $\mathbf{x}_{\oplus}^{\mathrm{T}}$ que satisfaz as restrições do problema é uma **solução válida**;
  – Qualquer solução válida cujo valor associado da função objectivo é igual ou inferior ao valor de todas as outras soluções válidas é **solução óptima** do problema.
  » Por definição, um problema de optimização pode ter mais do que uma solução óptima.
  » Exemplo: *problema de Programação Inteira Não-Linear puro*

$$\min_{y_1, y_2} \left\{ 3y_1 + 2y_2^2 \;\middle|\; \begin{array}{l} y_1^4 - y_2 - 15 = 0 \\ y_1 + y_2 - 3 \geq 0, \end{array} \quad y_1, y_2 \in U = \mathbb{N}_0 = \{0, 1, 2, ...\} \right\}$$

> Soluções válidas »»» $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_1 = \lambda \\ y_2 = \lambda^4 - 15 \end{pmatrix} \quad , \quad \lambda \in \{2, 3, 4, ...\}$

> Solução óptima (única) »»» $\mathbf{y}^* = (y_1, y_2)^* = (2, 1)$

Introdução

- Formulação equivalente e classes de problemas

  (*Fonte: J. Kallrath e T. Maindl, 2006*)

Classe de problemas	$f(\mathbf{x},\mathbf{y})$	$\mathbf{h}(\mathbf{x},\mathbf{y})$	$\mathbf{g}(\mathbf{x},\mathbf{y})$	n_d
Programação Linear	$\mathbf{c}^{\mathrm{T}}\mathbf{x}$	$\mathsf{A}\mathbf{x}-\mathbf{b}$	$\mathbf{x}$	0
Programação Linear Inteira Mista	$\mathbf{c}^{\mathrm{T}}\mathbf{x}_{\oplus}$	$\mathsf{A}\mathbf{x}_{\oplus}-\mathbf{b}$	$\mathbf{x}_{\oplus}$	≥ 1
Programação Não-Linear Inteira Mista				≥ 1
Programação Quadrática Inteira Mista	$\mathbf{x}_{\oplus}^{\mathrm{T}}\mathsf{Q}\mathbf{x}_{\oplus}+\mathbf{c}^{\mathrm{T}}\mathbf{x}_{\oplus}$	$\mathsf{A}\mathbf{x}_{\oplus}-\mathbf{b}$	$\mathbf{x}_{\oplus}$	≥ 1
Programação Não-Linear				0
Optimização Global				≥ 0

olunas;

Componentes
Índices

(*Fonte: J. Kallrath e T. Maindl, 2006; H. Williams, 2005*)

- Importância dos índices
  - Aspectos de implementação (escalabilidade do modelo, ligação conceptual com o modelo de dados);
  - Aspectos de desenho e concepção do modelo (índices representam objectos básicos do modelo);
  - Legibilidade.
- Exemplo
  - » Modelizar quantidades de um produto produzido em diferentes períodos de tempos (12, p.ex.) em diferentes locais (5, p.ex.)
    - » Má abordagem: considerar cada quantidade separadamente;
    - » Boa abordagem: usar uma matriz de variáveis *x* indexadas por período de tempo e local.

    - » Usar t=1, 2, 3, ... 12 para indexar períodos de tempo, e l= 1, 2, ..., 5 para indexar os locais;
    - » O domínio dos índices ({1, 2, 3, ..., 12} para t, e {1, 2, ...,5} para l) é chamado o conjunto de índices;

    - ! O conjunto de índices não é restrito a valores inteiros;
    - > Podemos ter: $l \in \{Braga,\ Porto,\ Vila\ Nova\ de\ Mil\ Fontes\}$

Componentes
Índices

- Expressões
  - Para $T \in \{1, 2,3, \ldots, 12\}$ e $\mathcal{L} \in \{1, 2,3, \ldots, 12\}$, temos as seguintes expressões equivalentes

$$
\begin{aligned}
x_{tl} &\leq 5 \quad ; t \in \mathcal{T} \quad , \quad l \in \mathcal{L} \\
x_{tl} &\leq 5 \quad ; t = 1, 2, ..., 12 \quad ; \quad l = 1, 2, ..., 5 \\
x_{tl} &\leq 5 \quad ; \forall t \quad , \quad \forall l \quad . \\
x_{tl} &\leq 5 \quad ; \forall \{tl\}
\end{aligned}
$$

  - Operadores:
    - $\in$: "é elemento de";
    - $\forall$: "para todos"
    - Exemplo

$$
\begin{aligned}
x_{tl} &\leq 5 \quad ; t = 2, 4, ..., 12 \quad ; \quad l = 1, 2, ..., 5 \\
x_{tl} &\leq 5 \quad ; \forall (t, l) \in \{(2, 1), (2, 5), (12, 5)\}
\end{aligned}
$$

Componentes
Dados

- Modelos de optimização dependem de dados prontamente disponíveis

PROBLEMA **>** Nem sempre (raramente?) estes dados estão disponíveis

EXEMPLO **>** Um processo de produção caracterizado por 17 estágios intermédios
  - o Qual é o valor dos custos que devo considerar?
  - o Discordâncias dos “clientes” nos custos a usar e na quantificação do seu valor.

SOLUÇÃO **>** Compromissos

! Compromissos *vs* confiança do utilizador no modelo

- Abordagem
  - Definir os melhores procedimentos de recolha de dados possíveis;
  - Definir procedimentos aceites por todos, em particular se o modelo tem de ser resolvido usando dados que mudam com o tempo;
  - Dados novos devem ser levantados usando os mesmos procedimentos que os dados antigos;
  - Deixar claro que os resultados assentam em assunções;
  - Deixar claro também que os resultados podem ser relevantes também desde que analisados de forma crítica.
- Abordagens para mitigar deficiências nos dados
  - Sensibilidade e inspecção de cenários alternativos;
  - Análise pós-optimalidade;
  - Análise de intervalos (de variação de dados/parâmetros).

Componentes
Variáveis

- Programação Matemática | Tipos de variáveis
  - **Variáveis contínuas**: $x \geq 0$
    - Variáveis não-negativas com valor entre 0 e X, $\mathrm{X} \geq x \geq 0$, sendo que X pode tender para infinito;
    - Exemplo: quantidade produzido de um determinado artigo.
  - **Variáveis inteiras**: $x \in \{0,1,2,3,4,\ldots\}$
    - Variáveis não-negativas que só podem tomar valores inteiros entre 0 e um dado limite superior;
    - Exemplo: número de pessoas a contratar.
  - **Variáveis binárias**: $x \in \{0,1\}$
    - Variáveis inteiras especiais que só podem tomar o valor 0 ou 1;
    - Formulação equivalente: $1 \geq x \geq 0, e\ x \in \mathbb{N}^0$;
    - Relaxação (ou relaxação do domínio): $1 \geq x \geq 0, e\ x \in \mathbb{R}$;
    - Variáveis binárias modelam decisões com duas alternativas (a atribuição do valor 0 ou 1 a uma alternativa depende de quem concebe o modelo);
    - Variáveis binárias são frequentemente associadas a variáveis contínuas (var. binária: produzo ou não; var. contínua: quanto produzo);

    » Variáveis binárias usadas para formular dicotomias;

    » Variáveis contínuas para formular as consequências dessas decisões.

Componentes
Variáveis

- Programação Matemática | Tipos de variáveis

  menos usuais

  – **Variáveis semi-contínuas**
    - Variáveis que podem tomar o valor 0, ou qualquer valor positivo entre X^- e X^+, com $X^- > 0$ e X^+ podendo tender para infinito;
    - Variáveis usadas para modelar processos industriais em que
      – OU não se produz nada;
      – OU se produz acima de um limite mínimo.

  – **Variáveis semi-inteiras**
    - Generalização das variáveis semi-contínuas;
    - Variáveis devem tomar valores inteiros apenas se o seu valor for inferior a um determinado limite;
    - Acima desse limite, a parte fraccionária da variável é considerada insignificante comparada com a parte inteira, e é desprezada.
    - Variável é considerada contínua se o seu valor for suficientemente elevado.

  – **Variáveis livres**
    - Variáveis em modelos de Programação Matemática são normalmente consideradas como não-negativas;
    - Algumas quantidades (lucros vs. Perdas, p.ex.) podem assumir valores positivos ou negativos que podem ser modeladas usando variáveis sem restrição de sinal;
    - Variável livre *x* pode ser expressa como a diferença entre duas variáveis não-negativas *y* e *z*: *x*=*y*-*z.*

Componentes
Restrições

- Restrições: limitam os valores que as variáveis podem assumir
- Forma geral

(LADO ESQUERDO DA RESTRIÇÃO) **RELAÇÃO** (LADO DIREITO DA RESTRIÇÃO)

RELAÇÃO »»» $\leq, \geq,$ ou $=$

- Modelos de Programação Linear e Programação Linear Inteira Mista
  - » Expressões lineares em ordem às variáveis
  - » Exemplos

$$x + y + w = 12$$
$$2x - 3y + 4.7w \geq 5.8$$
$$x + 3y - z \leq 5 \quad \Leftrightarrow \quad x + 3y \leq 5 + z$$

  - » Variáveis binárias permitem formular relações lógicas e funções mais complexas (valor absoluto, p.ex.)

Componentes
Restrições

- Alguns tipos de restrições
  - **Restrições de capacidade**
    - Recursos consumidos numa determinada actividade são limitados;
    - Esses limites (escassez de recursos) restringem efectivamente o espectro de alternativas;
    - Exemplos: capacidade de produção, número de colaboradores numa equipa.
  - **Disponibilidade de matérias-primas**
  - **Restrições comerciais e limites**
    - Limite no número de produtos que podem ser vendidos;
    - Procura de um cliente que obriga à produção de um valor mínimo de produtos;
    - Restrições desta natureza dão origem a limites (inferior ou superior) simples nas variáveis.
  - **Restrições de balanceamento (continuidade)**
    - Situações em que as quantidades que são *input* de um processo devem ser iguais ao total do que sai do processo:

$$\sum_j x_j - \sum_k y_k = 0$$

    - Os pesos numa restrição desse género podem representar perdas ou ganhos no processo.

Componentes
Restrições

- Alguns tipos de restrições
  - **Restrições/requisitos de qualidade**
    - Exemplo: nutrientes resultantes da combinação de alimentos, valores de octanas em petróleos,...
  - **Restrições fortes v*s.* restrições fracas**
    - Restrições fortes: especificações técnicas, p.ex., não violáveis/não relaxáveis (capacidade ditada pelo diâmetro de um tubo, p.ex.);

$$\sum_j a_j x_j \leq b$$

Restrições fracas: restrições que podem ser violadas por um determinado custo (disponibilidade de matéria-prima p.ex.)

$$\sum_j a_j x_j - u \leq b$$

Com restrições de igualdade, e para modelar a possibilidade de exceder ou ficar aquém do valor exigido:

$$\sum_j a_j x_j + u - v = b$$

Componentes
Restrições

- Alguns tipos de restrições
  - **Restrições conflituosas**
    - Existência de restrições que não podem ser todas satisfeitas simultaneamente;
    - Objectivo: satisfazer "o mais possível" todas as restrições;
    - Abordagem:
      i. Restrições originais $$\sum_j a_{ij}x_j = b_i,\ \forall i$$
      ii. Transformar as restrições originais em restrições fracas $$\sum_j a_{ij}x_j + u_i - v_i = b_i$$

Alternativas

1. Minimizar o total dos desvios nas restrições relativamente aos valores dos b_i;
2. Minimizar o desvio máximo.

Abordagens

1. FO: Minimizar a soma total das variáveis *u* e *v*;
2. Adicionar $$\begin{array}{l} z - u_i \geq 0, \forall i \\ z - v_i \geq 0, \forall i \end{array}$$, e minimizar *z*.

Componentes
Restrições

- Alguns tipos de restrições
  - **Restrições redundantes**
    - Restrição *binding, non-binding*: solução óptima com folga nessa restrição;
    - Pode não ser (e na prática não é muitas vezes) aparente a priori quais serão as restrições *binding* e *non-binding*;
    - Restrição *non-binding* pode tornar-se *binding* com outros dados.
  - **Limites simples e generalizados**
    - Limites simples: só envolvem uma variável ;
    - Limites generalizados: envolvem um conjunto de variáveis:

$$\sum_j x_j \leq b$$

  - **Restrições não usuais**
    - Exemplo: restrições lógicas;
    - Necessitam tipicamente recorrer a variáveis inteiras/binárias.

Componentes

Restrições

- Restrições com variáveis binárias
  - » Relações lógicas

Proposições lógicas R_i; Variáveis binárias y_j

$\neg R_1$ corresponde a $1 - y_1$;

$R_1 \wedge R_2$ corresponde a $y_1 + y_2 = 2$;

$R_1 \wedge R_2 \wedge R_3 \wedge \ldots \wedge R_n$ corresponde a $y_1 + y_2 + y_3 + \ldots + y_n = n$;

$R_1 \vee R_2$ corresponde a $y_1 + y_2 \geq 1$;

$R_1 \vee R_2 \vee R_3 \vee \ldots \vee R_n$ corresponde a $y_1 + y_2 + y_3 + \ldots + y_n \geq 1$;

$R_1 \vee (R_2 \wedge R_3)$ corresponde a $y_1 + y_2 \geq 1$ e $y_1 + y_3 \geq 1$;

$R_1 \wedge (R_2 \vee R_3)$ corresponde a $y_1 = 1$ e $y_2 + y_3 \geq 1$;

$R_1 \Rightarrow R_2$ corresponde a $y_1 \leq y_2$;

$R_1 \Rightarrow (R_2 \wedge R_3)$ corresponde a $y_1 \leq y_2$ e $y_1 \leq y_3$;

$R_1 \Rightarrow (R_2 \vee R_3)$ corresponde a $y_1 \leq y_2 + y_3$;

Componentes
Função objectivo

- Função objectivo (FO)
  - Baseada no critério de valorização das soluções alternativas de um modelo;
  - Estipula qual é a melhor solução em função do critério escolhido.
- FO | Exemplos
  - Maximizar lucro;
  - Minimizar perdas;
  - Maximizar a utilidade;
  - Maximizar o volume de negócios;
  - Maximizar ROI;
  - Maximizar valor líquido presente;
  - Maximizar o número de empregados;
  - Minimizar o número de empregados;
  - Minimizar redundâncias;
  - Maximizar a satisfação dos clientes;
  - Maximizar a probabilidade de sobrevivência;
  - Maximizar a robustez de um plano de operações;
  - ...

Componentes
Função objectivo

- Tipos de objectivos
  - Único;
  - Múltiplos e conflituosos;
  - Não optimizáveis

- Tipos de objectivos | Objectivo único
  - PL/PLIM: contribuições variáveis e proporcionais ao valor das variáveis;
  - PL/PLIM: custos fixos são formulados usando variáveis binárias
  - FO clássicas: *min / max;*
  - Outras FO
    - Minimax $\min \quad \max_i \sum_j a_{ij} x_j$

$$\text{s. a} \quad \ldots$$

pode ser formulado » » »

$$\boxed{\begin{aligned} &\min \quad z \\ &\text{s. a} \quad \sum_j a_{ij} x_j - z \leq 0, \ \forall i \end{aligned}}$$

Componentes
Função objectivo

- Tipos de objectivos | Objectivo único
  - Outras FO
    - Rácios (não linear)

$$\min \quad \frac{\sum_j a_j x_j}{\sum_j b_j x_j}$$

$$\text{s. a} \quad \sum_j d_j x_j \geq e$$

pode ser linearizado desde que $\sum_j b_j x_j$ tenha sempre o mesmo sinal:

1) Substituir $\dfrac{1}{\sum_j b_j x_j}$ por uma variável *t*;

2) Substituir os produtos $x_j t$ por variáveis w_j

A FO passa a ser $\min \text{ ou } \max \quad \sum_j a_j w_j$

3) Introduzir a restrição $\sum_j b_j w_j = 1$

e converter as restrições do modelo original em $\sum_j d_j w_j - et \geq 0$

APLICAÇÕES: balanceamento de inputs/outputs em função da escolha de pesos e factores (Análise qual. de dados)

Componentes
Função objectivo

- Tipos de objectivos | Objectivos múltiplos e conflituosos
  - Existência de vários objectivos
    - Exemplo: maximização das vendas + minimização dos *stocks* (conflito!)
  - » Abordagens
    - > Escolher um objectivo principal, e analisar o impacto das soluções nos outros objectivos;
    - > Resolver o modelo com cada um dos objectivos, e comparar as soluções;
    - > Trocar restrições e objectivos
      - Os modelos de Programação Matemática permitem fazer isso de forma muito simples;
      - Exemplo: objectivos de carácter social *vs.* minimização de custos;
    - > Exprimir os vários objectivos através de uma combinação linear de cada objectivo
      - Problema: determinar os pesos de cada objectivo;
      - Abordagem: testar com pesos diferentes e avaliar soluções obtidas com o cliente para definir políticas;
    - > Optar pela formulação de um modelo multi-objectivo.

Componentes
Função objectivo

- Tipos de objectivos | Objectivos não optimizáveis (ou não existentes)
  - Situações em que não existe um objectivo para optimizar (sobrevivência de uma empresa, p.ex.)
  - Modelo de Programação Matemática pode mesmo assim ter muito valor
    - > Determinar uma solução válida para um problema difícil (PLIM, p.ex.);
    - > Avaliar soluções extremas à luz de um determinado objectivo
      - > Se as soluções obtidas não são aceitáveis, a validade do modelo pode estar em causa;
      - > Instrumento para avaliar a validade de um modelo de Programação Matemática.

Exemplo

(*Fonte: H. Williams, 2005*)

- Área: GESTÃO DE PESSOAL

❖ O CONTEXTO

Companhia passa por processo de transformação que vai afectar as suas necessidades de pessoal nos próximos anos. Máquinas novas vão ser instaladas. As necessidades em pessoal não qualificado vão diminuir, enquanto as necessidades em pessoal qualificado ou semi-qualificado irão aumentar. Além disso, a crise irá provocar uma redução global das necessidades em pessoal de todas as categorias.

Necessidade em pessoal para os próximos 3 anos

	Não qualificado	**Semi-qualificado**	**Qualificado**
Trabalhadores actuais	2000	1500	1000
Ano 1	1000	1400	1000
Ano 2	500	2000	1500
Ano3	0	2500	2000

Exemplo

- Área: GESTÃO DE PESSOAL

  - **O CONTEXTO**

    A companhia quer decidir a política a seguir durante os próximos 3 anos no que toca a:

    i. Recrutamento
    ii. Formação
    iii. Despedimento
    iv. Redução do tempo de trabalho

    Um número não desprezível de trabalhadores abandonam o seu posto durante o 1º ano de trabalho. As percentagens de saídas em função do tempo de serviço e qualificações são as seguinte:

	Não qualificado	Semi-qualificado	Qualificado
Menos de 1 ano de serviço	25%	20%	10%
Mais de 1 ano de serviço	10%	5%	5%

Não houve recrutamentos feitos recentemente, e todos os colaboradores actuais estão ao serviço há mais de um ano.

Exemplo

- Área: GESTÃO DE PESSOAL

  - **O CONTEXTO**

    Recrutamento

    O número de trabalhadores que podem ser contratados no exterior da companhia é limitado. Em qualquer ano, esses limites são dados na tabela seguinte:

Não qualificado	Semi-qualificado	Qualificado
500	800	500

Formação

- Até 200 trabalhadores não qualificados podem ser formados, tornando-se semi-qualificados. O custo por trabalhador dessa formação é de 400€;
- Trabalhadores semi-qualificados podem tornar qualificados através de uma formação cujo custo é de 500€ por trabalhador. O número máximo de trabalhadores abrangidos não pode exceder 25% do número total de trabalhadores já qualificados na companhia (a formação envolve treino na própria companhia);
- É possível fazer um *downgrade* de qualificações (a custo 0), mas em média 50% dos trabalhadores abrangidos sairão da companhia.

Exemplo

- Área: GESTÃO DE PESSOAL
  - **O CONTEXTO**

Despedimento

Indemnizações:

- Trabalhador não qualificado: 200€;
- Trabalhador semi-qualificado ou qualificado: 500€.

Contratação adicional

Podem ser contratados até 150 trabalhadores adicionais, com os custos adicionais seguintes:

Não qualificado	Semi-qualificado	Qualificado
1500€	2000€	3000€

Redução do tempo de trabalho

Até 50 trabalhadores em cada categoria podem ser sujeitos a uma redução do seu tempo de trabalho. O custo associado (por trabalhador e por ano) é o seguinte:

Não qualificado	Semi-qualificado	Qualificado
500€	400€	400€

Um trabalhador nesta situação realiza metade do trabalho de um trabalhadora tempo inteiro.

Exemplo

- Área: GESTÃO DE PESSOAL
    - **O CONTEXTO**
        - **OBJECTIVO DA COMPANHIA**

Minimizar o número de despedimentos.

Se a política da companhia consitisse em minimizar os custos totais, qual seria a poupança realizada comparado com o cenário anterior?

Exemplo

- Área: GESTÃO DE PESSOAL
  - ❖ **O MODELO**
    - » **ASSUNÇÕES**
      - – Tudo acontece no primeiro dia de cada ano:
      - > Os trabalhadores são recrutados no primeiro dia do ano;
      - > Uma proporção desses trabalhadores sairá imediatamente (menos de um ano de serviço);
      - > Uma proporção dos trabalhadores já na companhia sairá imediatamente (mais de um ano de serviço);
      - > Um certo número de trabalhadores serão formados (simultaneamente);
      - > Um certo número de trabalhadores serão despedidos;
      - > Um certo número de trabalhadores irão ver o seu tempo de trabalho reduzido.

Exemplo

- Área: GESTÃO DE PESSOAL
  - **O MODELO**
    - **VARIÁVEIS**
      - ***Número de trabalhadores***

t_{SKi}: número de trabalhadores qualificados empregues no ano i;

t_{SSi}: número de trabalhadores semi-qualificados empregues no ano i;

t_{USi}: número de trabalhadores não qualificados empregues no ano i.

- ***Recrutamento***

u_{SKi}: número de trabalhadores qualificados recrutados no ano i;

u_{SSi}: número de trabalhadores semi-qualificados recrutados no ano i;

u_{USi}: número de trabalhadores não qualificados recrutados no ano i.

Exemplo

- Área: GESTÃO DE PESSOAL
  - **O MODELO**
    - » **VARIÁVEIS**
      - – ***Formação***

v_{USSSi}: n.º trabalhadores não qualificados em formação que se tornam semi-qualificados no ano i;
v_{SSSKi}: n.º trabalhadores semi-qualificados em formação que se tornam qualificados no ano i.

– ***Downgrading***

v_{SKSSi}: n.º trabalhadores qualificados que passam a semi-qualificados no ano i;
v_{SKUSi}: n.º trabalhadores qualificados que passam a não qualificados no ano i;
v_{SSUSi}: n.º trabalhadores semi-qualificados que passam a não qualificados no ano i.

– ***Despedimentos***

w_{SKi}: número de trabalhadores qualificados despedidos no ano i;
w_{SSi}: número de trabalhadores semi-qualificados despedidos no ano i;
w_{USi}: número de trabalhadores não qualificados despedidos no ano i.

Exemplo

- Área: GESTÃO DE PESSOAL
  - **O MODELO**
    - » **VARIÁVEIS**
      - *Redução do tempo de trabalho*

x_{SKi}: número de trabalhadores qualificados com redução do tempo de trabalho no ano i;

x_{SSi}: número de trabalhadores semi-qualificados com redução do tempo de trabalho no ano i;

x_{USi}: número de trabalhadores não qualificados com redução do tempo de trabalho no ano i.

- *Contratação adicional*

y_{SKi}: número de trabalhadores qualificados contratados adicionalmente no ano i;

y_{SSi}: número de trabalhadores semi-qualificados contratados adicionalmente no ano i;

y_{USi}: número de trabalhadores não qualificados contratados adicionalmente no ano i.

Exemplo

- Área: GESTÃO DE PESSOAL
  - **O MODELO**
    - » **RESTRIÇÕES**
      - – ***Continuidade***

$$t_{SKi} = 0.95t_{SKi-1} + 0.9u_{SKi} + 0.95v_{SSSKi} - v_{SKSSi} - v_{SKUSi} - w_{SKi},$$
$$t_{SSi} = 0.95t_{SSi-1} + 0.8u_{SSi} + 0.95v_{USSSi} - v_{SSSKi} + 0.5v_{SKSSi} - v_{SSUSi} - w_{SSi},$$
$$t_{USi} = 0.9t_{USi-1} + 0.75u_{USi} - v_{USSSi} + 0.5v_{SKUSi} + 0.5v_{SSUSi} - w_{USi}.$$

- – ***Formação dos trabalhadores semi-qualificados***

$$v_{SSSKi} - 0.25t_{SKi} \leq 0$$

- – ***Contratação adicional***

$$y_{SKi} + y_{SSi} + y_{USi} \leq 150$$

Exemplo

- Área: GESTÃO DE PESSOAL

  - **O MODELO**

    » **RESTRIÇÕES**

    – ***Requisitos***

$$t_{SKi} - y_{SKi} - 0.5x_{SKi} = 1000,\ 1500,\ 2000 \quad (i = 1, 2, 3)$$
$$t_{SSi} - y_{SSi} - 0.5x_{SSi} = 1400,\ 2000,\ 2500 \quad (i = 1, 2, 3)$$
$$t_{USi} - y_{USi} - 0.5x_{USi} = 1000,\ 500,\ 0 \quad (i = 1, 2, 3)$$

    – ***Condições iniciais***

$$t_{SK0} = 1000, t_{SS0} = 1500, t_{US0} = 2000$$

$(i = 1, 2, 3)$

Recrutamento: $u_{SKi} \leq 500,\ u_{SSi} \leq 800, u_{USi} \leq 500$

Redução do tempo de trabalho: $x_{SKi} \leq 50,\ x_{SSi} \leq 50, x_{USi} \leq 50$

Formação: $v_{USSSi} \leq 200$

Exemplo

- Área: GESTÃO DE PESSOAL
  - **O MODELO**
    - » **FUNÇÃO OBJECTIVO**
      - ***Minimizar o número de despedimentos***

$$\min \sum_i (w_{SKi} + w_{SSi} + w_{USi})$$

      - ***Minimizar custos totais***

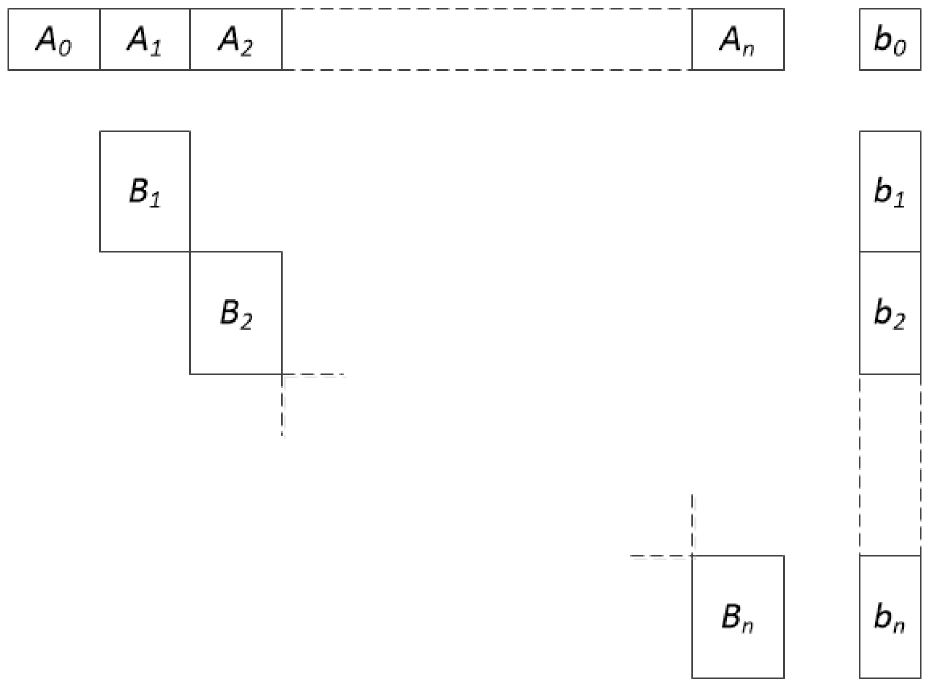
$$\min \sum_i (400v_{USSSi} + 500v_{SSSKi} + 500w_{SKi} + 500w_{SSi} + 200w_{USi} + 500x_{USi} + 400x_{SSi} + 400x_{SKi} + 1500y_{USi} + 2000y_{SSi} + 3000y_{SKi})$$

    - » **DIMENSÃO DO MODELO**
      - 60 variaveis;
      - 24 restrições;
      - 21 limites nas variáveis.

Estrutura

(*Fonte: H. Williams, 2005*)

- Estrutura angular em blocos
  - Estrutura geral » » »

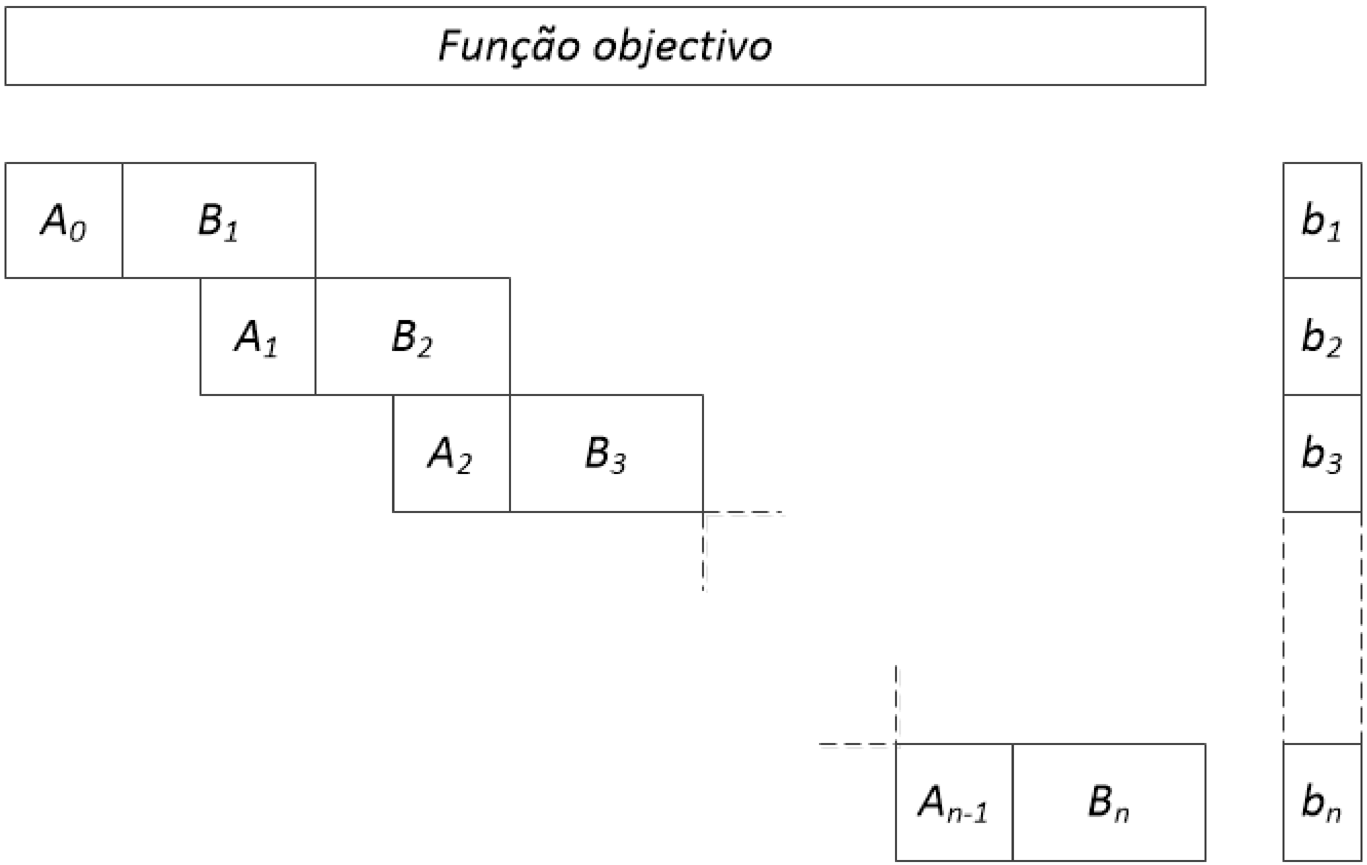


  - Aplicações (exemplos)
    - Modelos multi-fábrica (*multi-plant*);
    - Modelos multi-produto;
    - Modelos multi-período.

Estrutura

(*Fonte: H. Williams, 2005*)

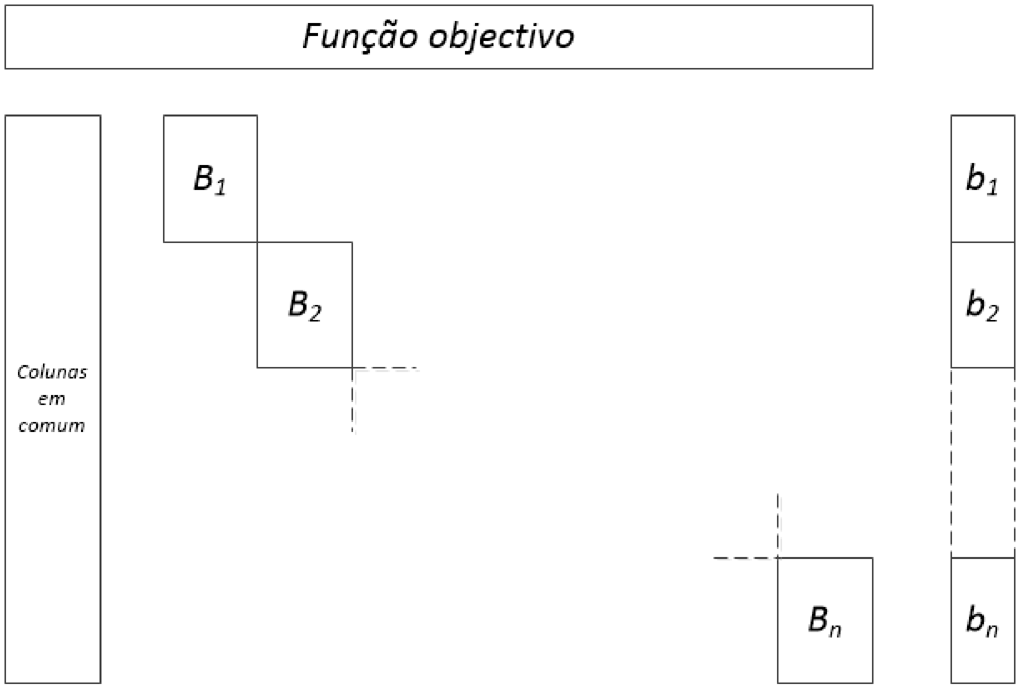
- Estrutura em “escada” (staircase)
  - Estrutura geral



Estrutura

(*Fonte: H. Williams, 2005*)

- Estrutura angular em blocos (dual)
  - Estrutura geral » » »



  - Aplicações (exemplos)
    - Modelos estocásticos
      - Situações em que as decisões são tomadas antes de outra informação necessária estar disponível;
      - Resultados das decisões num estágio *n* condicionam as restrições do estágio *n+1* ;
      - Por sua vez, a qualidade da decisão no estágio *n* depende (em parte) das decisões no estágio *n+1*.

Tipos especiais

(*Fonte: H. Williams, 2005*)

- Modelos de Programação Matemática | Tipos Especiais
  - Modelos económicos
    - Modelos estáticos
    - Modelos dinâmicos
  - Modelos de fluxos em rede
    - Problema de transportes
    - Problema de afectação
    - Problema de caminho mais curto
    - Problema de transbordo
    - Problema de fluxo de custo mínimo (máximo)
    - Caminho crítico

Métodos de Optimização
Introdução aos métodos heurísticos

- Heurísticas | Motivação
  - As heurísticas são usadas para determinar rapidamente soluções de boa qualidade para problemas cuja complexidade inviabiliza o recurso a um método exacto;
- Diferença com métodos exactos de optimização
  - » NÃO garantem que a solução óptima do problema seja encontrada;
- Vários tipos de métodos heurísticos
  - Específicos a um determinado problema;
  - Baseados numa estrutura geral, adaptáveis a cada caso particular
    - » *meta-heurísticas, p.ex.*

Métodos de Optimização
Introdução aos métodos heurísticos

- Heurísticas mais simples » » » Métodos de construção
- Pesquisa local
  - Procuram encontrar soluções melhores analisando a vizinhança da solução corrente;
  - Inconveniente: convergem muitas vezes para mínimos locais;
  - Abordagem: *meta-heurísticas* (procuram escapar a mínimos locais permitindo transições para soluções de pior qualidade).
- Outros tipos de métodos heurísticos
  - *Algoritmos de aproximação:* métodos com os quais é possível obter soluções cuja qualidade é conhecida.

Métodos de Optimização
Métodos de construção

- Métodos de construção | Definição
  - » Heurísticas usadas para construir de raiz uma solução válida para um problema;
  - » Método específico de um determinado problema
    - = *não pode ser usada para resolver um problema diferente*
  - » Na maior parte dos casos, são procedimentos simples, e fáceis de implementar
- Métodos de construção | Abordagens
  - » Estratégias miópicas (ou gulosas)
    - = *Em cada passo, a decisão é tomada usando um critério de "satisfação imediata"*
  - » Reduzir a instância do problema a um caso especial que seja de fácil resolução

Métodos de Optimização

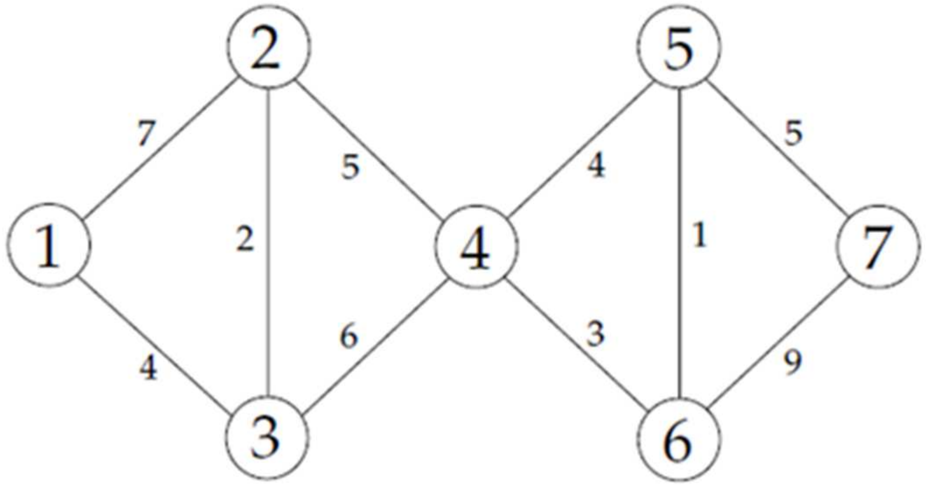
Métodos de construção

- EXEMPLO | Problema da Árvore de Suporte de Custo Mínimo
  - O problema consiste em determinar o subgrafo de um grafo G=(V,A) tal que exista um caminho entre todos os vértices, e o custo total dos arcos escolhidos seja o menor possível;
  - Esse subgrafo é uma árvore: não tem ciclos, e é constituído por |V|-1 arcos;
  - Método de construção: adicionar os arcos por ordem crescente do seu custo, excepto se a adição do arco gerar um ciclo;
  - Uma solução construída dessa forma é sempre uma solução óptima do problema da árvore de suporte de custo mínimo.

Métodos de Optimização

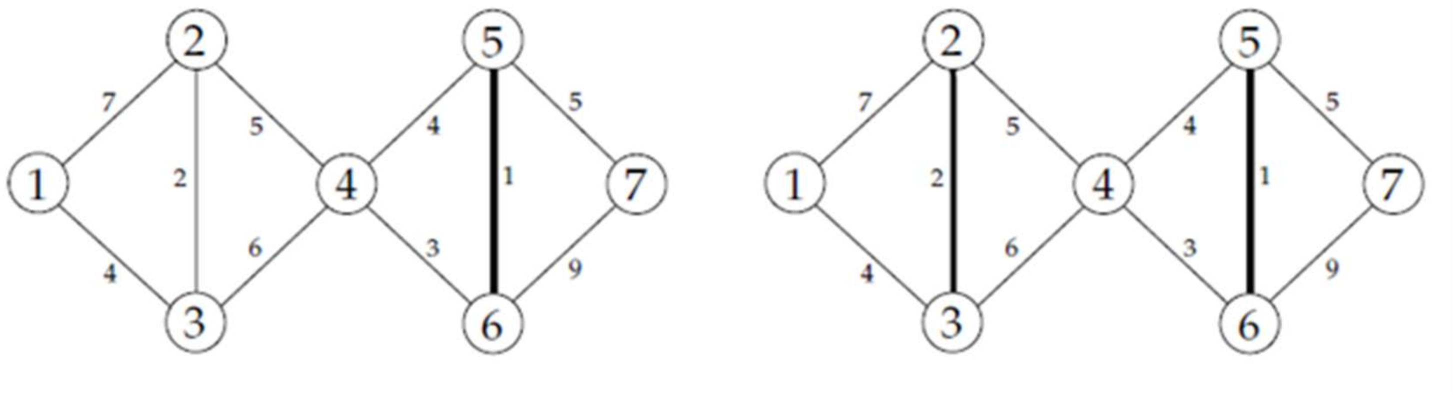
Métodos de construção

- EXEMPLO | Problema da Árvore de Suporte de Custo Mínimo
  - Uma instância do problema da árvore de suporte de custo mínimo



Métodos de Optimização
Métodos de construção

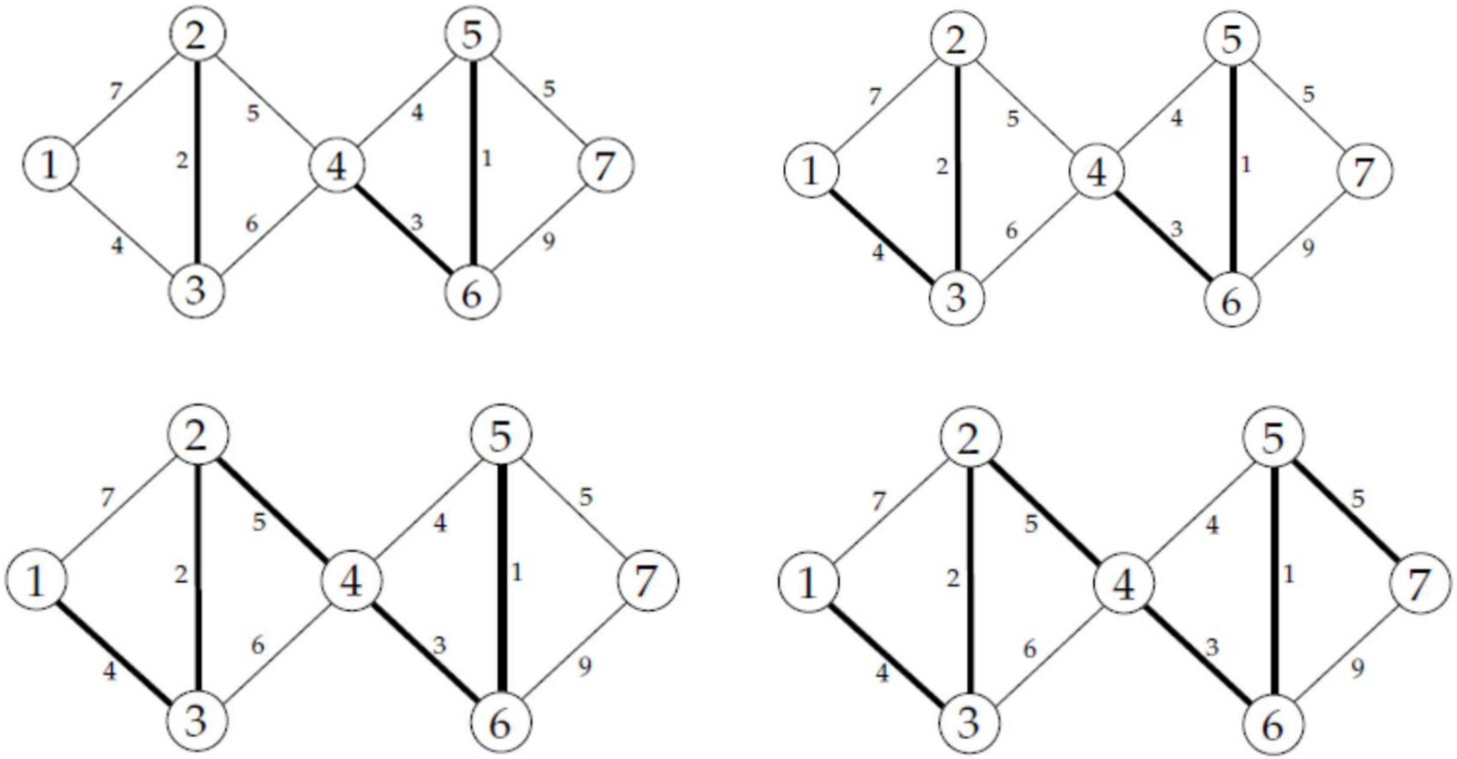
- EXEMPLO | Problema da Árvore de Suporte de Custo Mínimo
  - RESOLUÇÃO
    - Solução (árvore) é constituída por 6 arcos (7-1)
    - Primeiro arco a ser escolhido: arco (5,6)
    - Arcos seguintes: (2,5), (4,6), (1,3)
    - Arco (4,5) deveria ser escolhido a seguir, mas não é considerado porque gera um ciclo



Métodos de Optimização

Métodos de construção

- EXEMPLO | Problema da Árvore de Suporte de Custo Mínimo
  - RESOLUÇÃO



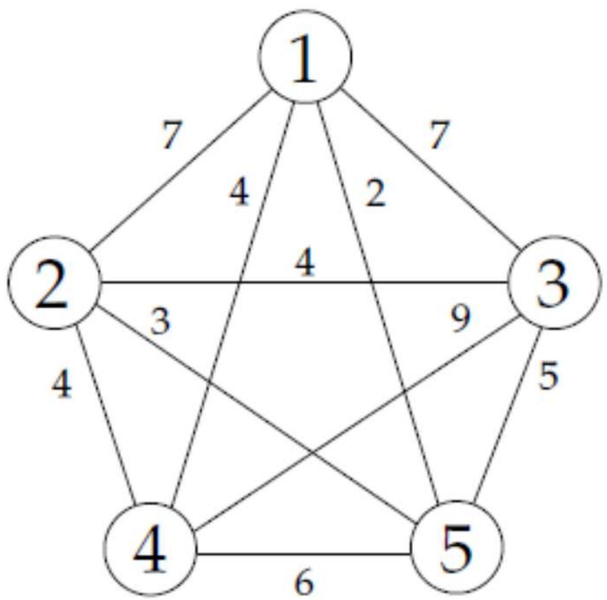
Métodos de Optimização

Métodos de construção

- EXEMPLO | Problema do Caixeiro Viajante
  - Geralmente, os métodos de construção permitem apenas determinar soluções sub-óptimas
  - Exemplo: o problema do caixeiro viajante (problema NP-difícil);
    - O problema consiste em determinar o circuito de menor custo que passa por todos os vértices de um grafo uma única vez
  - Heurísticas (exemplos):
    - Vizinho mais próximo: adicionar arcos por ordem crescente do seu custo
    - Heurísticas de inserção: vértices são adicionados passo-a-passo a um circuito (podem ser usados diferentes critérios para determinar o próximo vértice a inserir, e a sua posição no circuito).

Métodos de Optimização
Métodos de construção

- EXEMPLO | Problema do Caixeiro Viajante
  - Uma instância do problema



Métodos de Optimização
Métodos de construção

- EXEMPLO | Problema do Caixeiro Viajante
  - RESOLUÇÃO
    - Solução gerada pela heurística do vizinho mais próximo partindo do vértice 1
      - Circuito: 1-5-2-4-3-1 (custo [comprimento] = 25)
    - ➢ Uma heurística de inserção
      - Escolher o vértice mais próximo de um dos vértices do circuito corrente;
      - Inserir esse vértice numa posição tal que o incremento do custo seja o menor possível;
      - ❖ Partindo do vértice 1
        - Escolher o vértice 5 (o mais próximo do vértice 1): circuito resultante “1-5-1”
        - Próximo vértice: 2 → “1-2-5-1”
        - Próximo vértice: 4 (o mais próximo de um dos vértices do circuito)
          - o Determinar a sua posição no circuito (critério: menor incremento do custo

$$\min\{c_{14} + c_{42} - c_{12}, c_{24} + c_{45} - c_{25}, c_{54} + c_{41} - c_{51}\} = \min\{1, 7, 8\} = 1.$$

c_{ij}: custo do arco (i,j);

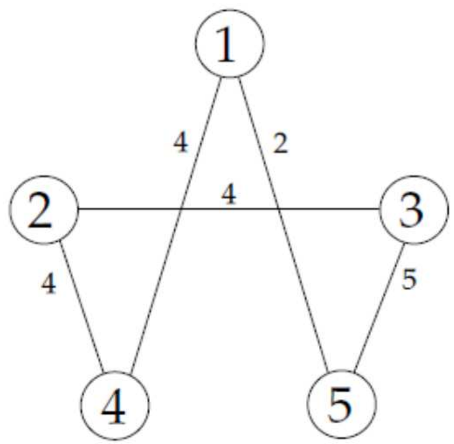
- o O vértice é inserido entre o 1 e o 2 → “1-4-2-5-1”

Métodos de Optimização
Métodos de construção

- EXEMPLO | Problema do Caixeiro Viajante
  - RESOLUÇÃO
    - Uma heurística de inserção
      - Partindo do vértice 1
        - Próximo vértice: 3
          - Determinar a sua posição no circuito (critério: menor incremento do custo)

$$\min\{c_{13} + c_{34} - c_{14}, c_{43} + c_{32} - c_{42}, c_{23} + c_{35} - c_{25}, c_{53} + c_{31} - c_{51}\} = 7.$$

          - Circuito final: “1-4-2-3-5-1”, custo = 20



Métodos de Optimização

Pesquisa local

- Métodos de pesquisa local | Objectivos e princípios
  - » Tentar determinar soluções de melhor qualidade partindo de uma solução inicial
  - » A pesquisa dessas soluções é feita de forma iterativa, explorando a vizinhança da solução corrente
  - » Terminam quando não é possível encontrar uma solução de melhor qualidade na vizinhança
  - » Métodos de pesquisa local são métodos de melhoramento
  - » A solução que é encontrada no final é melhor que as outras soluções da sua vizinhança (assumindo que é explorada toda a vizinhança) > **óptimo local**
  - » Muitas vezes, a solução encontrada é apenas óptimo local (não é óptima para o problema global)
  - » A optimalidade da solução para o problema global depende da função de vizinhança
  - » Elemento fundamental: funções de vizinhança (determinam o conjunto de soluções que podem ser exploradas numa iteração)

Métodos de Optimização
Pesquisa local

- Funções de vizinhança
  - Uma função de vizinhança é exacta, se a solução gerada pelo método de pesquisa local for a solução óptima do problema (método Simplex, p. ex.)
- Funções de vizinhança | Exemplo
  - Problema do Caixeiro Viajante:
    - Circuitos representados como sequências de vértices (a-b-c-d-e-a, p.ex.);
    - Função de vizinhança definida a partir da troca de dois elementos da sequência:
    - Vizinhança de a-b-c-d-e-a:

$$a - c - b - d - e - a$$
$$a - b - d - c - e - a$$
$$a - b - c - e - d - a$$

Métodos de Optimização

Pesquisa local

- Métodos de pesquisa local | Elementos
  - » Método de geração de soluções iniciais
  - » Função de vizinhança
  - » Função de avaliação

    Usada para avaliar as soluções da vizinhança, e escolher a solução seguinte

  - » Regra de selecção da próxima solução

    Exemplos: escolher a melhor, escolher a primeira que melhore o valor da solução corrente

  - » Critério de paragem

    Terminar a execução quando não é possível encontrar uma solução melhor na vizinhança

    Outros critérios (exemplos): tempo de execução, número de iterações

Métodos de Optimização
Pesquisa local

- Métodos de pesquisa local | Elementos
  - Funções de vizinhança
    - Problema de optimização *P*
    - Instância *I* de *P*: conjunto de soluções válidas *S*, e uma função objectivo *f* → *I* = (*S*,*f*)
    - Função de vizinhança *V*: *S* → *T* determina o conjunto de soluções que são consideradas vizinhas das soluções de *S*
    - Uma solução j é vizinha de i se e só se $j \in V(i)$
    - Geralmente, essas funções são simétricas: se $j \in V(i)$, então $i \in V(j)$

Métodos de Optimização
Pesquisa local

- Métodos de pesquisa local | Elementos
  - Funções de vizinhança | Exemplos

    Problema de Programação Inteira

    - Função de vizinhança: trocar o valor de k variáveis
    - Exemplo: vizinhança de $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1)$ para *k*=2

$$
\begin{aligned}
&(0, 1, 0, 1),\\
&(0, 0, 1, 1),\\
&(0, 0, 0, 0),\\
&(1, 1, 1, 1),\\
&(1, 1, 0, 0),\\
&(1, 0, 1, 0).
\end{aligned}
$$

Métodos de Optimização

Pesquisa local

- Métodos de pesquisa local | Elementos
  - Funções de vizinhança | Exemplos

    Problema do Caixeiro Viajante

    - Soluções (circuitos) representadas através de sequências de vértices
    - Função de vizinhança: variar a posição de um único vértice no circuito
      - Exemplo: vizinhança de a-b-c-d-e-a

$$b - a - c - d - e - b$$

$$b - c - a - d - e - b$$

$$b - c - d - a - e - b$$

Métodos de Optimização

Pesquisa local

- Métodos de pesquisa local | Elementos
  - Funções de vizinhança | Exemplos

    Problema do Caixeiro Viajante

    - Soluções (circuitos) representadas através de sequências de vértices
    - Outra função: troca de k vértices
      - Exemplo: para k=3, a vizinhança de a-b-c-d-e-a é a seguinte

$$c - b - a - d - e - c$$

$$a - d - c - b - e - a$$

$$a - b - e - d - c - a$$

$$d - b - c - a - e - d$$

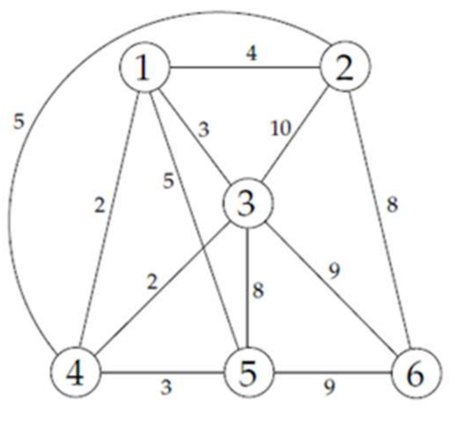
Métodos de Optimização

Pesquisa local

- Métodos de pesquisa local | Exemplo

Problema do Caixeiro Viajante

- Instância



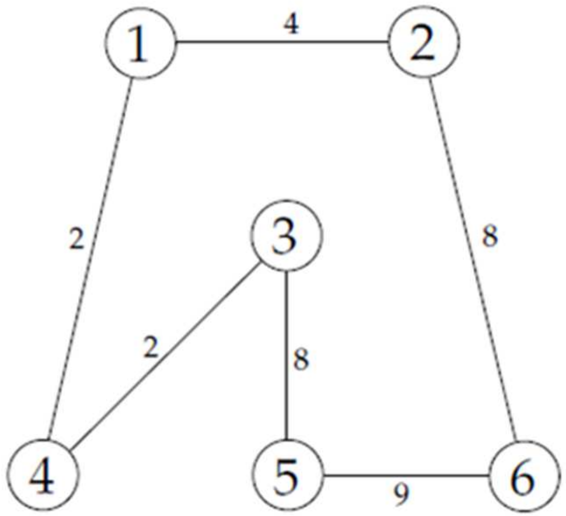
- Método de pesquisa local:
  - Solução inicial obtida usando a heurística do vizinho mais próximo
  - Função de avaliação baseada no comprimento dos circuitos (função objectivo do problema)

Métodos de Optimização
Pesquisa local

- Métodos de pesquisa local | Exemplo

Problema do Caixeiro Viajante

- Solução inicial
  - Circuito: “1-4-3-5-6-2-1”
  - Comprimento: 33



Métodos de Optimização

Pesquisa local

- Métodos de pesquisa local | Exemplo

Problema do Caixeiro Viajante

- Função de vizinhança: trocar a posição de um vértice no circuito
  - Vizinhança de “1-4-3-5-6-2-1”, trocando a posição do vértice 1

$$4-1-3-5-6-2-4 \quad (\text{Comprimento} = 34)$$
$$4-3-1-5-6-2-4 \quad (\text{Comprimento} = 32)$$

  - Vizinhança de “1-4-3-5-6-2-1”, trocando a posição do vértice 3

$$1-4-5-3-6-2-1 \quad (\text{Comprimento} = 34)$$
$$1-4-5-6-3-2-1 \quad (\text{Comprimento} = 37)$$
$$1-4-5-6-2-3-1 \quad (\text{Comprimento} = 35)$$
$$3-4-5-6-2-1-3 \quad (\text{Comprimento} = 29)$$

Métodos de Optimização
Pesquisa local

- Métodos de pesquisa local | Exemplo

  Problema do Caixeiro Viajante

  - Função de vizinhança: trocar a posição de um vértice no circuito
    - Vizinhança de "1-4-3-5-6-2-1", trocando a posição do vértice 4

$$1-3-4-5-6-2-1 \quad (\text{Comprimento} = 29)$$
$$1-3-5-6-2-4-1 \quad (\text{Comprimento} = 35)$$

    - Vizinhança de "1-4-3-5-6-2-1", trocando a posição do vértice 5

$$5-4-3-6-2-1-5 \quad (\text{Comprimento} = 31)$$
$$1-4-5-3-6-2-1 \quad (\text{Comprimento} = 35)$$

Métodos de Optimização
Pesquisa local

- Métodos de pesquisa local | Exemplo

  Problema do Caixeiro Viajante

  - Função de vizinhança: trocar a posição de um vértice no circuito
    - Próxima solução: a de menor comprimento na vizinhança da solução corrente

      (2 circuitos de comprimento 29; escolhemos arbitrariamente um deles)

$$1 - 3 - 4 - 5 - 6 - 2 - 1$$

    - Não existem soluções de comprimento inferior na vizinhança desse circuito
      - ➢ Essa solução é um óptimo local

Métodos de Optimização
Pesquisa tabu

- Pesquisa tabu| Motivação e princípios básicos
  - » Métodos de pesquisa local convergem frequentemente para óptimos locais
  - » Métodos baseados em pesquisa tabu = meta-heurísticas que procuram escapar a óptimos locais permitindo transições para soluções de pior qualidade
  - » Pesquisa tabu integra procedimentos de pesquisa local
  - » Métodos de pesquisa tabu prosseguem mesmo depois de ter atingido um óptimo local
  - » Para evitar que processo de pesquisa entre em ciclo, esses métodos recorrem à memória *= listas tabu*

Métodos de Optimização
Pesquisa tabu

- Lista tabu
  - » Registal os últimos movimentos que foram efectuados

    Movimentos = transições entre soluções vizinhas numa iteração
  - » Esses movimentos são proibidos (tabu)

    Numa iteração, todas as soluções associadas a movimentos tabu não são consideradas
  - » Exemplo | Problema do Caixeiro Viajante

    A lista tabu pode ser usada para registar pares de arcos eliminados nas 3 últimas iterações;
  - » Os critérios de aspiração são usados para permitir movimentos que à partida são tabus

    Se a solução resultante desse movimento for melhor que a melhor solução encontrada até ao momento, p.ex.
  - » Na pesquisa tabu, a memória pode ser de curto prazo ou de longo prazo (tamanho da lista tabu).

Métodos de Optimização
Pesquisa tabu

- Lista tabu | Memória de curto prazo
  - » Métodos de pesquisa tabu baseados em memória de curto prazo são os mais simples
  - » Esses métodos não eliminam por completo o risco da pesquisa entrar em ciclo
  - » O tamanho da lista tabu condiciona a pesquisa: quanto menor for, maior será a tendência da pesquisa se concentrar na vizinhança do óptimo local
  - » Listas maiores permitem escapar mais facilmente a um óptimo local
  - » Mecanismos para acelerar a pesquisa em vizinhanças de grande dimensão: listas de candidatos guardam “bons” movimentos que serão escolhidos preferencialmente

Pesquisa Tabu

Input: Instância $I = (S, f)$ de um problema de optimização (minimização) ;

`/* Assumimos que a função objectivo do problema é usada para avaliar as soluções */`

Seja t o tamanho da lista tabu ;

Determinar uma solução inicial x_0 para I ;

$\overline{x} := x_0$, $\overline{f} := f(x_0)$, $i := 1$;

Enquanto *não é satisfeito o critério de paragem* **fazer**

> Seja V' um subconjunto das soluções na vizinhança de x_{i-1} ;
>
> $f' := +\infty$;
>
> `/* Determinar a melhor solução na vizinhança` V' `de` x_{i-1}`. */`
>
> **Para todos os** $x' \in V'$ **fazer**
>
> > **se** *((x' não está associada a um movimento tabu) ou (x' satisfaz algum critério de aspiração)) e ($f(x') < f'$)* **então**
> >
> > > $x_i := x'$, $f' := f(x')$;
> >
> > **se** $f' < \overline{f}$ **então**
> >
> > > $\overline{x} := x'$, $\overline{f} := f'$;
>
> Actualizar a lista tabu ;
>
> `/* Remover o movimento tabu mais antigo, e colocar o novo no início da lista. */`
>
> $i := i + 1$;

Output: $(\overline{x}, \overline{f})$;

Stop ;

Métodos de Optimização

Pesquisa tabu

- Lista tabu | Memória de longo prazo
  - » Os elementos guardados estão associados a movimentos relativos a iterações efectuadas desde o início da pesquisa
  - » A informação é usada para guiar a pesquisa para regiões do espaço de soluções que não estejam na vizinhança da solução corrente
  - » Esses métodos recorrem a registos de frequências (permanência de um atributo na solução, execução de um movimento) para penalizar ou favorecer movimentos, incluindo movimentos que conduzam a soluções fora da vizinhança da solução corrente
  - » Estratégias adicionais de intensificação (concentrar a pesquisa em determinadas regiões) e diversificação (conduzir a pesquisa para regiões que não ainda não exploradas)

Métodos de Optimização
Pesquisa tabu

- Lista tabu | Exemplo (memória de curto prazo)

Problema de Programação Inteira

$$
\begin{array}{lrcrcrcrcrcrcrcl}
\max\ z = & 12x_1 & + & 17x_2 & + & 7x_3 & + & 23x_4 & + & 16x_5 & + & 19x_6 & + & 13x_7 & & \\
s.\ a & & & & & & & & & & & & & & & \\
 & 7x_1 & + & 9x_2 & + & 3x_3 & + & 11x_4 & + & 13x_5 & + & 5x_6 & + & 7x_7 & \leq & 25 \\
 & 17x_1 & + & 23x_2 & + & 9x_3 & + & 27x_4 & + & 28x_5 & + & 14x_6 & + & 19x_7 & \leq & 60 \\
 & 8x_1 & + & 8x_2 & + & 5x_3 & + & 14x_4 & + & 11x_5 & + & 12x_6 & + & 7x_7 & \leq & 32 \\
 & 9x_1 & + & 25x_2 & + & 6x_3 & + & 22x_4 & + & 31x_5 & + & 23x_6 & + & 11x_7 & \leq & 49 \\
 & 6x_1 & + & 16x_2 & + & 7x_3 & + & 17x_4 & + & 17x_5 & + & 12x_6 & + & 8x_7 & \leq & 33 \\
 & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & , & x_7 & \in & \{0,1\}
\end{array}
$$

Métodos de Optimização

Pesquisa tabu

- Lista tabu | Exemplo (memória de curto prazo)

Problema de Programação Inteira

» Tamanho da lista tabu L: 3

» Função de vizinhança: troca do valor de uma única variável

» Solução inicial

Exemplo: atribuir o valor 1 às variáveis por ordem decrescente do seu coeficiente na FO

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (0, 0, 0, 1, 0, 1, 0)$$

Solução de valor z = 42

» **1ª iteração**

L={}; Vizinhança da solução corrente

$$\begin{array}{ll} (0,0,0,0,0,1,0) & z = 19 \\ (0,0,0,1,0,0,0) & z = 23 \end{array}$$

Métodos de Optimização
Pesquisa tabu

- Lista tabu | Exemplo (memória de curto prazo)

Problema de Programação Inteira

» Escolhemos a 2ª solução (trocar valor de x6) que passa a ser a solução corrente;

» A lista tabu é actualizada: $L = \{x_6\}$

» **2ª iteração**

Vizinhança

$$\begin{array}{ll} (1,0,0,1,0,0,0) & z = 35 \\ (0,1,0,1,0,0,0) & z = 40 \\ (0,0,1,1,0,0,0) & z = 30 \\ (0,0,0,0,0,0,0) & z = 0 \\ (0,0,0,1,0,1,0) & \textbf{tabu} \\ (0,0,0,1,0,0,1) & z = 36 \end{array}$$

Escolhemos a 2ª solução (trocar valor de x2)

Métodos de Optimização

Pesquisa tabu

- Lista tabu | Exemplo (memória de curto prazo)

Problema de Programação Inteira

» **3ª iteração**

Lista tabu: $L = \{x_2, x_6\}$

Vizinhança $(0,0,0,1,0,0,0)$ **tabu**

$(0,1,0,0,0,0,0)$ $z = 17$

» O método encontra a solução óptima na 8ª iteração

$$(1,0,1,0,0,1,1) \qquad z = 51$$

... o método prosseguiria até que fosse verificado o critério de paragem estipulado inicialmente.

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Princípios básicos
  - » Os métodos de pesquisa por arrefecimento simulado (simulated annealing) são meta-heurísticas que tentam reproduzir processos termodinâmicos usados p. ex. na indústria metalúrgica e vidreira
  - » Esses métodos determinam a próxima solução a partir da vizinhança da solução corrente (pesquisa local)
  - » Em vez de explorar toda a vizinhança da solução corrente, os métodos de pesquisa por arrefecimento simulado vão gerar essas soluções de forma aleatória

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Princípios básicos

  » Se a solução que tiver sido gerada for melhor do que a solução corrente, ela é escolhida automaticamente

  » Caso contrário, a solução é escolhida com uma probabilidade p que depende dos valores dessa solução e da solução corrente e de uma temperatura T: $p = e^{-\Delta t}$

com $$\Delta t = \frac{|z_{i-1} - z_i'|}{T}$$

z_{i-1}: valor da solução corrente;
z'_i: valor da solução gerada na iteração i.

Métodos de Optimização

Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Princípios básicos
  - » A probabilidade *p* é tanto maior quanto maior for a temperatura *T*, e quanto menor for a diferença entre o valor das soluções
  - » Inicialmente, a temperatura tem um valor elevado para favorecer a escolha de soluções piores do que a solução corrente
  - » O valor de T é diminuído pouco a pouco até que o valor da probabilidade de aceitação seja muito pequeno
  - » A diminuição é feita segundo um esquema de arrefecimento que é definido em função do problema

$$T_{k+1} = \alpha T_k$$

Com $0 < \alpha < 1$ (o parâmetro que é usado tipicamente varia entre 0.8 e 0.99).

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Princípios básicos
  - » A temperatura T_k é mantida durante L_k iterações
  - » Geralmente, temos que L_k =L, para qualquer k
  - » Não existem regras gerais para determinar o valor de L e da temperatura inicial: esses valores dependem do problema e da função de vizinhança

Pesquisa por Arrefecimento Simulado

Input: Instância $I = (S, f)$ de um problema de optimização ;

`/* Assumimos que a função objectivo do problema é usada para avaliar as soluções */`

Input: Esquema de arrefecimento, temperatura inicial T_0, L ;

`/* Assumimos que` $L_k = L, \ \forall k$ `*/`

Determinar uma solução inicial x_0 para I ;

$\overline{x} := x_0$, $\overline{f} := f(x_0)$, $i := 1$;

$k := 0$, $j := 1$;

Enquanto *não é satisfeito o critério de paragem* **fazer**

  Gerar aleatoriamente uma solução x' na vizinhança de x_{i-1} ;

  se $(f(x') \leq f(x_{i-1}))$ **então**

    $x_i := x'$;

    se $f(x') < \overline{f}$ **então**

      $\overline{x} := x'$, $\overline{f} := f'$;

  senão

    $p := e^{-\frac{|f(x_{i-1}) - f(x')|}{T_k}}$;

    Gerar um número aleatório p' entre 0 e 1 ;

    se $p' \leq p$ **então**

      $x_i := x'$;

    senão

      $x_i := x_{i-1}$;

    $i := i + 1$, $j := j + 1$;

    se $j > L$ **então**

      $j := 1$, $k := k + 1$;

      Calcular T_k segundo o esquema de arrefecimento ;

Output: $(\overline{x}, \overline{f})$;

Stop ;

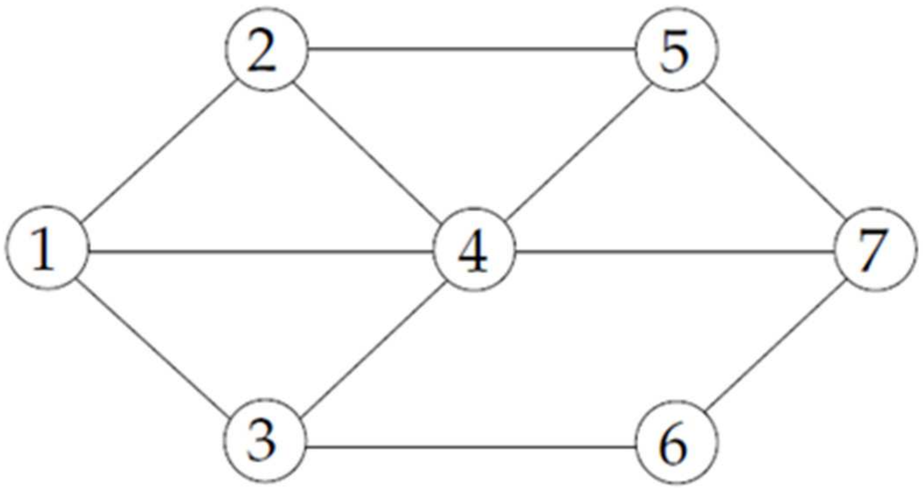
Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo

Problema de coloração de grafos

» Determinar o número mínimo de cores necessárias para pintar os vértices de um grafo de modo a que dois vértices adjacentes não tenham cores idênticas

» Uma instância do problema



Métodos de Optimização
Pesquisa por arrefecimento simulado

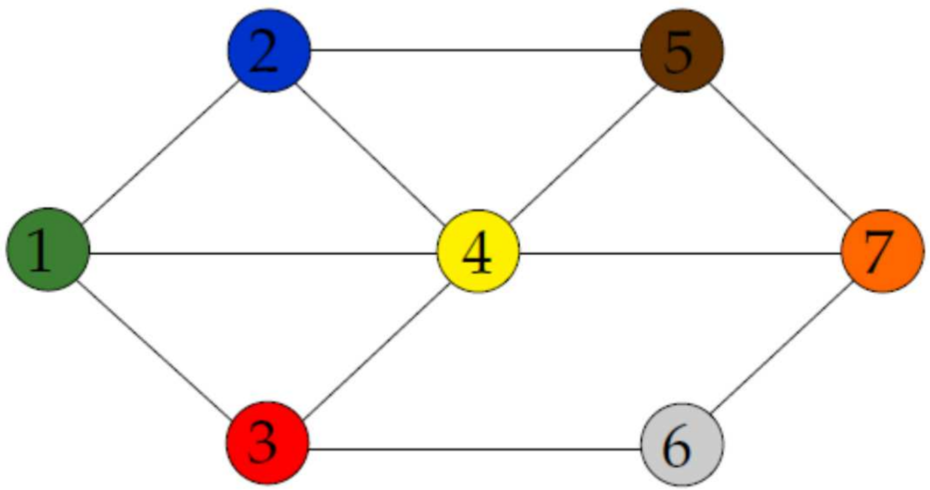
- Pesquisa por arrefecimento simulado| Exemplo

Problema de coloração de grafos

» Soluções representadas através de conjuntos de vértices aos quais foi atribuída a mesma cor

» As soluções vão ser avaliadas com base na FO do problema (número de cores)

» Solução inicial: uma cor para cada vértice



Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo

Problema de coloração de grafos

» Método de geração de soluções vizinhas

» A probabilidade de escolher um vértice é a mesma para todos os vértices: (1/7)

» Escolhemos uma cor entre todas as cores usadas pelos outros vértices e uma ``outra cor'' (com a mesma probabilidade)

» A cor escolhida é atribuída ao vértice escolhido

» Esquema de arrefecimento: actualização da temperatura segundo a expressão geométrica apresentada atrás com $\alpha = 0.7$ e T0 = 7 (número de vértices do grafo)

» Temperatura é mantida ao longo de 3 iterações consecutivas

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo

  Problema de coloração de grafos

  » Temperaturas e probabilidade *p* de aceitação

α	k	T_k	$\lvert z_{i-1} - z'_i \rvert$	Probabilidade de aceitação
0.7	0	7.00000	1	0.86688
	1	4.90000		0.81540
	2	3.43000		0.74711
	3	2.40100		0.65936
	4	1.68070		0.55157
	5	1.17649		0.42742
	6	0.82354		0.29693
	7	0.57648		0.17646
	8	0.40354		0.08390
	9	0.28248		0.02901
	10	0.19773		0.00636

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo
  - **1ª iteração**

Custo da solução corrente: 7

Solução candidata:

a) Escolher vértice:

- Número aleatório: 0.17724
- Vértice escolhido: 2

b) Escolher cor:

- Número aleatório: 0.20193
- Cor escolhida: Vermelho

Solução candidata é válida? *sim*

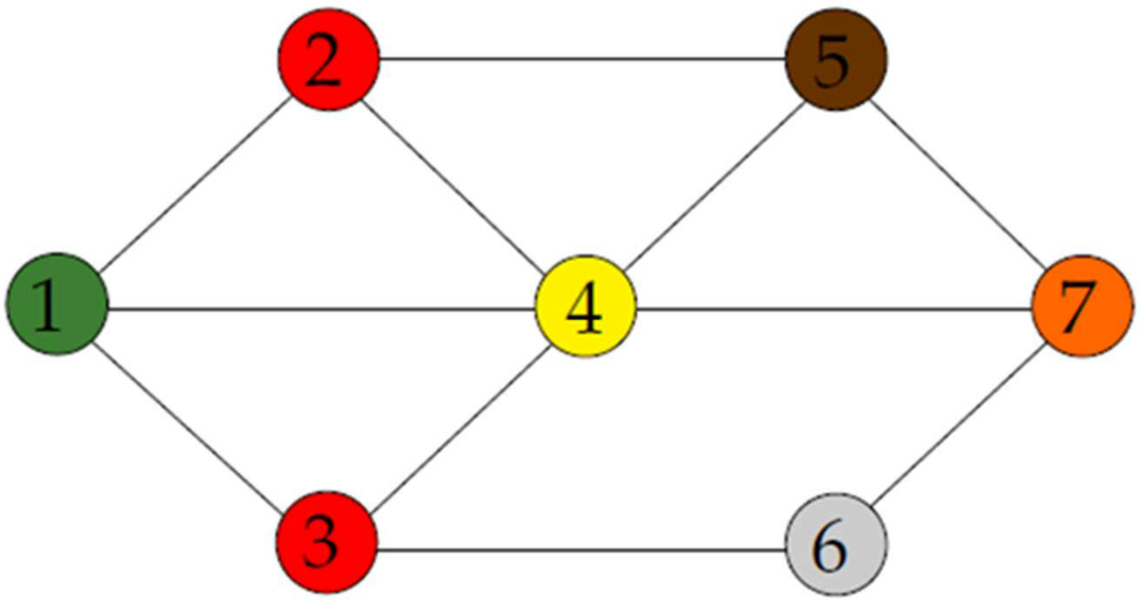
Custo da solução candidata: 6. A solução é automaticamente escolhida.

Número de iterações com T_0: 1

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo
  - » **2ª iteração**

    Solução corrente (custo: 6)



Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo
  - » **2ª iteração**

    Solução candidata

    a) Escolher vértice:

    – Número aleatório: 0.59710

    – Vértice escolhido: 5

    b) Escolher cor:

    – Número aleatório: 0.51524

    – Cor escolhida: Cinzento

    Solução candidata é válida? *sim*

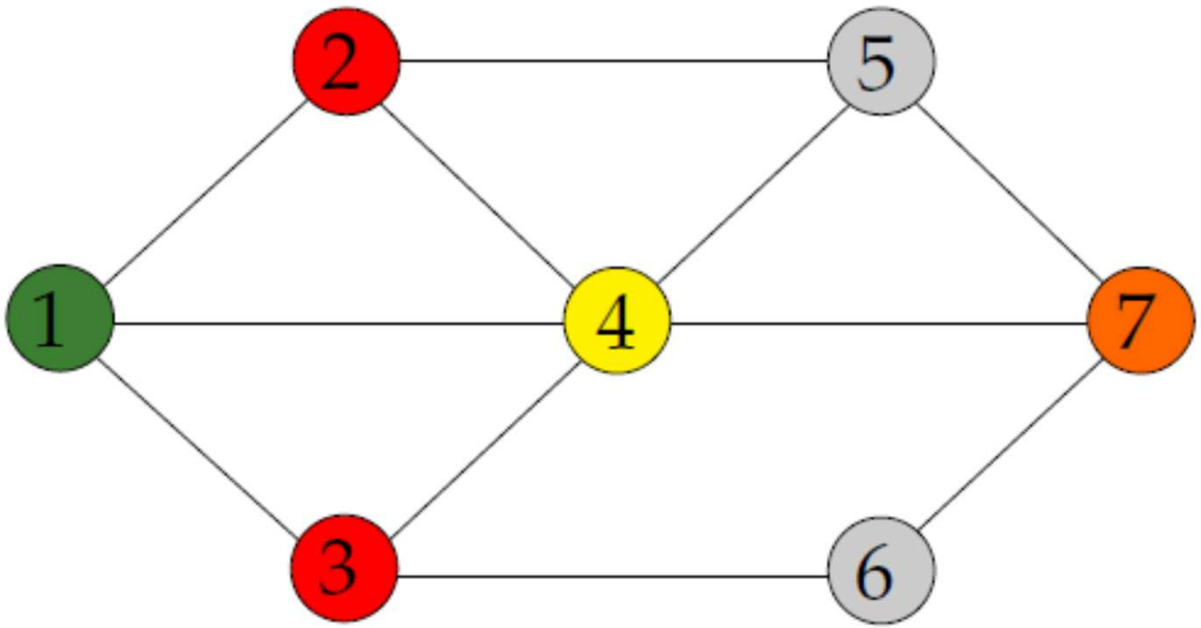
    Custo da solução candidata: 5. A solução é automaticamente escolhida.

    Número de iterações com T_0: 2

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo
  - » **3ª iteração**

    Solução corrente (custo: 5)



Métodos de Optimização

Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo
  - » **3ª iteração**

    Solução candidata

    a) Escolher vértice:

    – Número aleatório: 0.45562

    – Vértice escolhido: 4

    b) Escolher cor:

    – Número aleatório: 0.87103

    – Cor escolhida: "Outra cor"

    Solução candidata é válida? *sim*

    Custo da solução: 6

Métodos de Optimização
Pesquisa por arrefecimento simulado

- Pesquisa por arrefecimento simulado| Exemplo
  - » **3ª iteração**

A solução deve ser escolhida?

- Probabilidade de aceitação da solução candidata: $e^{-\frac{1}{7}} = 0.86688$
- Número aleatório: 0.23017; como $0.23017 \leq 0.86688$, a solução é aceite.

Número de iterações com T_0: 3
A temperatura é actualizada. Passamos a usar a temperatura T_1 (4.9)

O processo é repetido até que sejam satisfeitos os critérios de paragem estipulados.

Bibliografia

- H. Williams, “Model building in Mathematical Programming”, Wiley, 2002.
- J. Kallrath, T. Maindl, “Real Optimization with SAP APO”, Springer, 2006.
- Z. Michalewicz, D. B. Fogel, “How to Solve it: Modern Heuristics”, Springer, 2ª ed., 2004.
- C. Blum e A. Roli, “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”, ACM Computing Surveys, 35(3):268–308, 2003.
- F. Glover e M. Laguna, “Tabu Search”, Kluwer Academic Publishers, 1997.