Otimização de horários
Seminário

Clarisse Resende
25/01/2013

Descrição do problema

- O problema dos horários consiste numa sequência de atividades de programação, satisfazendo um conjunto de restrições de recursos.

- Pretende-se realizar **M** encontros, **T** número de vezes, recorrendo-se a **R** recursos e respeitando **C** restrições.

- Esta apresentação foca-se essencialmente na realização de horários escolares.

Descrição do problema

- A criação de horários escolares pode dar a sensação de ser um problema simples. Na verdade, é um problema bastante complexo e que tem vindo a ser um tema de estudo tanto matemáticamente como computacionalmente.

- Possíveis restrições:
  - São evitados choques de horários;
  - São respeitadas as habilitações específicas dos professores para lecionarem as disciplinas, dando a devida importância à experiência e preferência;
  - São consideradas as indisponibilidades de cada professor;
  - São respeitados os limites de carga horária de cada professor.

Descrição do problema

- Este problema é então classificado como um problema de otimização combinatorial: considerando uma contagem de n professores e m disciplinas, teríamos n^m soluções.

- Existem inúmeras restrições possíveis para um problema desta categoria. Como tal, elas podem ser classificadas como:

  - Restrições rígidas: Evitam que um estudante tenha dois exames ao mesmo tempo e que a capacidade de uma sala nunca seja ultrapassada, por exemplo;
  - Restrições flexíveis: Tentam que um estudante não tenha exames em intervalos seguidos ou que se minimize o número total de períodos de tempo necessários para realizar todos os exames, entre muitas outras variantes.

Descrição do problema

- Considere-se agora um problema em que se tem como objetivo:
  - Encontrar a solução com o menor número de versões total de exame usadas dentro de um número máximo de secções dado

- Onde:
  - Secção: Duração de um exame;
  - A fim de tornar a programação mais razoável, assume-se que cada exame pode ter um máximo de *n* versões.

Descrição do problema

- Tabela 1:

Objectivo e restrições do problema
Objectivo
0. Encontrar a solução com o menor número de versões total de exame usadas dentro de um número máximo de secções dado
Restrições
1. Cada estudante deve fazer todos os exames que era suposto fazer inicialmente; 2. Nenhum estudante deve fazer dois ou mais exames na mesma secção; 3. Cada exame pode ser realizado n vezes; 4. Lotação da sala; 5. Deve haver salas suficientes para se realizarem todos os exames.

Descrição do problema

- Seja:
  - $I = \{1, \dots, |I|\}$: o conjunto de número de identificação dos estudantes;
  - $J = \{1, \dots, |J|\}$: o conjunto de número de identificação dos exames;
  - $S = \{1, \dots, |S|\}$: o conjunto de secções disponíveis;
  - $R = \{1, \dots, |R|\}$: o conjunto de salas disponíveis.

- Parâmetros:
  - $t_{ij}, i \in I, j \in J$: valores binários, 1 se o estudante i faz o exame j, 0 caso contrário.
  - $m_r, r \in R$: número máximo de lugares em cada sala;
  - n: número de versões de cada exame;

Descrição do problema

- Variáveis:
  - $x_{ijsr}, i \in I, j \in J, s \in S, r \in R$: variáveis de decisão binárias, 1 se o estudante I faz o exame j, na secção s, na sala r, e 0 caso contrário.
  - $y_{jsr}, j \in J, s \in S, r \in R$: variável de decisão binária, 1 se o exame j é realizado na secção s e na sala r, 0 caso contrário.
  - $z_{js}, j \in J, s \in S$: variável de decisão binária, 1 se o exame j é realizado na secção s, 0 caso contrário.

Descrição do problema

Objetivo:

$$\min \sum_{j \in J, s \in S} z_{js} \qquad (1)$$

Restrições:

$$\sum_{s \in S, r \in R} x_{ijsr} = t_{ij} \quad \forall i \in I, j \in J \qquad (2)$$

$$\sum_{j \in J, r \in R} t_{ij} \cdot x_{ijsr} \leq 1 \quad \forall i \in I, s \in S \qquad (3)$$

$$\sum_{i \in I} x_{ijsr} \leq m_r \cdot y_{jsr} \quad \forall j \in J, s \in S, r \in R \qquad (4)$$

- (1) O objetivo é minimizar o número total de versões usadas no exame.
- (2) Significa que cada aluno deve fazer todos os exames que era suposto fazer.
- (3) Garante que cada aluno não pode ter mais do que um exame ao mesmo tempo.
- (4) Se j exame é realizado na seção s e r sala de aula, o número total de estudantes devem satisfazer a lotação da sala de aula.

Descrição do problema

- Restrições:

$$\sum_{j \in J} y_{jsr} \leq 1 \quad \forall s \in S, r \in R \qquad (5)$$

$$\sum_{s \in S} y_{jsr} \leq |R| \cdot z_{js} \quad \forall j \in J, s \in S \qquad (6)$$

$$\sum_{s \in S} z_{js} \leq n \quad \forall j \in J \qquad (7)$$

- (5) Cada sala não pode ter mais do que um exame ao mesmo tempo.
- (6) É apenas necessário uma versão de exame se forem usadas múltiplas salas para o mesmo exame.
- (7) Dita que não pode haver mais do que n versões de exame para cada um destes.

Descrição do problema

- Em problemas de ordem prática, verifica-se que os tamanhos do problema são demasiado grandes para resolver.

- Normalmente, um problema real consiste em cerca de 15.000-20.000 estudantes, 70-100 exames, 8-10 secções e 100-200 salas de aula.

- Existem $|I| \cdot |J| \cdot |S| \cdot |R| + |J| \cdot |S| \cdot |R| + |J| \cdot |S| \approx |I| \cdot |J| \cdot |S| \cdot |R|$ possibilidades de solução final, que corresponde a um número demasiado elevado para ser desenvolvido através de programação matemática. Sendo assim, tenta-se uma reformulação do problema.

Descrição do problema

- A reformulação inicia-se pelas variáveis binárias com menor tamanho. Neste caso é a variável correspondente às salas. Assume-se então que as restrições 3 e 4 da Tabela 1 não precisam de ser respeitadas.
- Assim, a nossa formulação fica reduzida a:
- Seja:
  - $I = \{1, \dots, |I|\}$: o conjunto de número de identificação dos estudantes;
  - $J = \{1, \dots, |J|\}$: o conjunto de número de identificação dos exames;
  - $S = \{1, \dots, |S|\}$: o conjunto de secções disponíveis;

Descrição do problema

- Parâmetros:
  - $t_{ij}, i \in J$: valores binários, 1 se o estudante i faz o exame j, 0 caso contrário.
  - n: número de versões de cada exame;

- Variáveis:
  - $x_{ijs}, i \in I, j \in J, s \in S$: variáveis de decisão binárias, 1 se o estudante *i* faz o exame j, na secção s, e 0 caso contrário.
  - $z_{js}, j \in J, s \in S$: variável de decisão binária, 1 se o exame j é realizado na secção s, 0 caso contrário.

Descrição do problema

Objetivo:

$$\min \sum_{j\in J, s\in S} z_{js} \quad (8)$$

Restrições:

$$\sum_{s\in S} x_{ijs} = t_{ij} \quad \forall i \in I, j \in J \quad (9)$$

$$\sum_{j\in J} t_{ij} \cdot x_{ijs} \leq 1 \quad \forall i \in I, s \in S \quad (10)$$

$$\sum_{i\in I} x_{ijs} \leq |I| \cdot y_{js} \quad \forall j \in J, s \in S \quad (11)$$

$$\sum_{s\in S} y_{js} \leq n \quad \forall j \in J \quad (12)$$

- (11) Um exame tem de ser feito por mais do que um aluno.
- Nota: Apresenta-se referenciado no artigo a conclusão de que o que foi assumido para a reformulação do problema é razoável em termos práticos.

Descrição do problema

- É comum, no mundo real, que vários alunos partilhem o mesmo horário de exames. Se agrupar-mos estes alunos a solução ótima não será afetada e reduz-se o número de variáveis x_{ijs}.
- Este procedimento corta, implicitamente, muitos ramos de uma árvore de busca, poupando no tempo de execução computacional.
- Chega-se assim a uma divisão do problema original em vários sub-problemas, em que cada um deles é um problema de horários independente.
- Assumindo que existe sempre salas para realizar todos os exames, a solução ótima global, será a soma das soluções dos subproblemas.

Descrição do problema

- Exemplo 1:
  - Os estudantes 1 e 2 fazem o exame A e B, o estudante 3 faz o exame B e o estudante 4 faz o exame C. Assim, considera-se 3 grupos.

- Exemplo 2:
  - O estudante 1 faz o exame A e B, o estudante 2 faz o exame A e C, o estudante 3 faz o exame B e C, o estudante 4 faz o exame D e E, o estudante 5 faz o exame E. Podemos dividir os estudantes em duas divisões: $\{1,2,3\}$ e $\{4,5\}$. Assim como os exames: $\{A, B, C\}$ e $\{D, E\}$. Os nossos subproblemas serão portanto: $I_1 = \{1,2,3\}$; $J_1 = \{A, B, C\}$ e $I_2 = \{1,2,3\}$; $I_1 = \{D, E\}$.

Descrição do problema

- O procedimento de divisão pode ser descrito da seguinte maneira:
  - Passo 0: Considerar I o número de alunos disponíveis. Os subconjuntos $\bar{I}$ e $\bar{J}$ serão os subconjuntos de estudantes e exames, respectivamente, incialmente vazios.
  - Passo 1: Escolher um estudante de I. Se este estudante fizer pelo menos um exame em $\bar{J}$, adiciona-se todos os exames que este aluno pretende realizar ao subconjunto, adiciona-se o estudante a $\bar{I}$ e elimina-se de I. Passa-se para o estudante seguinte até que todos os estudantes de I sejam verificados.
  - Passo 2: $\bar{I}$ e $\bar{J}$ passam a ser um conjunto para um novo subproblema. Caso I não esteja vazio, retorna-se ao passo 1.

Descrição do problema

- Exemplo 3:
  - $I = \{1,2,3,4,5\}; J = \{A, B, C, D, E, F\}$ e a matriz t_{ij}:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

  - Primeiro agrupa-se a 1ª e 5ª linhas:

- G = $\begin{array}{c} \\ 1' \\ 2' \\ 3' \\ 4' \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$

  - Em que $1' = \{1,5\}, 2' = \{2\}, 3' = \{3\}, 4' = \{4\}.$

Descrição do problema

- Em seguida, o processo de divisão serve para transformar a matriz G numa matriz diagonal, multiplicando por umas determinadas matrizes de permutação P_1 e P_2: $G' = P_1 \mathrm{G} P_2$.

$$
\begin{array}{c} \\ 1' \\ 4' \\ 2' \\ 3' \end{array}
\begin{array}{c}
\begin{array}{cccccc} A & C & E & B & D & F \end{array} \\
\left[\begin{array}{cccc|cc}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1
\end{array}\right]
\end{array}
$$

- Resultam assim os subproblemas: $I_1 = \{1', 4'\} = \{1,4,5\}, J_1 = \{A, C, E\}$ e $I_2 = \{2', 3'\}, = \{2,3\}, J_2 = \{B, D, F\}$.

Descrição do problema

- A divisão em subproblemas é uma técnica que, segundo os autores, vale a pena tentar, dado que não danifica significativamente a solução final do problema, para além de, claramente, simplificar a resolução deste.

Método Tabu Search

- O método *Tabu search* foi criado em 1986 e formalizado em 1989;
- É um método de procura de uma nova solução (não necessariamente melhor do que a anterior) na vizinhança da solução atual – *método iterativo* – com vista a obter uma solução ótima global do problema e não apenas local;
- É bastante comum aplicá-lo em problemas de otimização matemática;

Método Tabu Search

- O objetivo é pesquisar localmente e chegar assim à solução ótima global.

- *Tabu search (TS)* usa um procedimento local ou de vizinhança para iterativamente partir de uma solução potencial para uma solução melhorada localizada na vizinhança da primeira, até que haja um critério que não tenha sido satisfeito por uma delas;

- A pesquisa local pode ficar presa em soluções pouco viáveis. Para evitar este problema e garantir que todo o espaço de pesquisa seja verificado, o *TS* explora, usando estruturas de memória, cuidadosamente toda a vizinhança de cada solução à medida que a pesquisa avança.

Método Tabu Search

- Se o espaço de pesquisa é muito grande ou de alta dimensionalidade, é fácil manter-se dentro de uma pequena área. Para contornar esse problema, algumas implementações do *TS* criam uma tabu list que consiste nos atributos de uma solução, em vez de considerar soluções candidatas.

- **Nota:**
  - Tabu list: regista as soluções que foram visitadas num passado recente.

- Listas tabu, contendo atributos (em vez de soluções integrais), podem ser mais eficazes para alguns domínios, embora eles levantam um novo problema.

- Quando um único atributo é marcado como tabu, resulta em mais do que uma solução ser tabu. Algumas dessas soluções, poderiam ser de excelente qualidade e não serão visitadas.

- No próximo exemplo, a lista tabu é simplesmente uma estrutura de memória de curto prazo (existe também a de longe e médio prazo), que irá conter um registo dos elementos das soluções visitadas recentemente.

Algoritmo

```
Procedimento Busca_Tabu()
     //Inicialização
Início-Busca_Tabu
1    Construa uma solução inicial s0
2    s := s0    //s é a solução corrente
3    f*:= f(s0) //Valor de s*
4    s*:= s0    //s* é a melhor solução
5    T := {}    //Inicialize Lista Tabu como vazia
     //Busca
6    Enquanto o critério de parada não é satisfeito faça
7    Início-Enquanto
8         Selecione s em argmin[f(s')]; onde s' pertence a Ñ(s)
          //N(s) é a vizinhança de s
          //Ñ(s) é o subconjunto de N(s) admissível
          //(não é tabu ou é permitido pelo critério de
          //aspiração)
9         Se f(s) < f* então
10        Início-Se
11             f*:= f(s)
12             s*:= s
13        Fim-Se
14        Grave como tabu o movimento atual em T
15   Fim-Enquanto
Fim-Busca_Tabu
```

Linha 1-5: serve para inicializar o algoritmo, escolhendo uma solução s0 aleatória. Essa solução é considerada inicialmente como a melhor encontrada até à data e a tabu list encontra-se vazia. Neste exemplo, a tabu list é simplesmente uma estrutura de memória de curto prazo, que irá conter um registo dos elementos dos estados visitados;

Algoritmo

```
Procedimento Busca_Tabu()
     //Inicialização
Início-Busca_Tabu
1    Construa uma solução inicial s0
2    s := s0     //s é a solução corrente
3    f*:= f(s0) //Valor de s*
4    s*:= s0     //s* é a melhor solução
5    T := {}     //Inicialize Lista Tabu como vazia
     //Busca
6    Enquanto o critério de parada não é satisfeito faça
7    Início-Enquanto
8         Selecione s em argmin[f(s')]; onde s' pertence a Ñ(s)
          //N(s) é a vizinhança de s
          //Ñ(s) é o subconjunto de N(s) admissível
          //(não é tabu ou é permitido pelo critério de
          //aspiração)
9         Se f(s) < f* então
10        Início-Se
11             f*:= f(s)
12             s*:= s
13        Fim-Se
14        Grave como tabu o movimento atual em T
15   Fim-Enquanto
Fim-Busca_Tabu
```

- **Linha 7-15:** este ciclo vai continuar a procurar uma solução ideal até que uma condição especificada for violada. Inicializa-se uma lista de candidatos vazia $\tilde{N}(s)$. São verificadas as soluções vizinhas para os elementos marcados como tabu. Se a solução não pertencer à lista tabu, é adicionada na lista candidata;

Terminar o algoritmo

- Em teoria, a pesquisa poderia continuar para sempre, a menos que o valor ideal do problema em questão seja conhecido. Na prática, evidentemente, a procura tem de ser interrompida em algum ponto.

- Os pontos de paragem mais comuns usados em TS são:
  - depois de um número fixo de iterações;
  - após um certo número de iterações sem uma melhoria do valor da função objectivo;
  - quando o objetivo atinge um valor limite pré-especificado.

- Em esquemas complexos de TS, a pesquisa é normalmente encerrada após uma sequência de fases, sendo a duração de cada fase determinada por um dos critérios acima referidos.

Referências

- http://dbpedia.org/page/Tabu_search

- *Weiwei Chen and Leyuan Shi, "A Variant of Examination Timetabling Problem", 2008*

- Moacir Kripka, Rosana Maria Luvezute Kripka and Neuza Terezinha Oro. "*Aplicação de técnicas de otimização na determinação da distribuição de cargas horárias na Universidade de Passo Fundo*"