OLPR0001

(27 de junho de 2016) – 7a.lista – *cumulative* e *disjuntive*

Fundamentos de Programação por Restrições

Joinville, 27 de junho de 2016

Sumário

1 Problemas do K-Cavalos

Este problema apresenta muitas variações e tão interessante no estudo da PR como o problema das N-rainhas. Neste problema, há um aspecto a mais que no problema da N-rainhas clássico não apresentava: a otimização. Exceto se for as N-rainhas com peso, ver no github do professor.

Para o nosso problema é dado por: *quantos cavalos K (nossos cavalos tem os mesmos movimentos permitidos ao jogo de xadrez) são necessários para cobrir um tabuleiro $N \times N$?* Diferentemente ao problema das rainhas, aqui os cavalos podem se atacar mutuamente, veja uma solução exemplo na figura 1.

Assim, sua tarefa é encontrar o menor número de cavalos necessários para atacar **todas** as células deste tabuleiro.

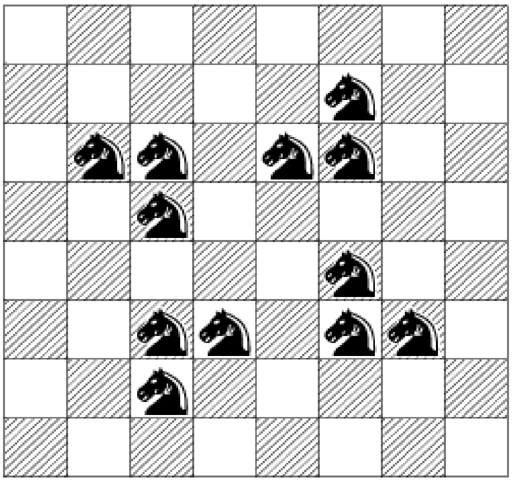

Figura 1: Cavalos atacando todas as células

Implemente este problema, contudo, rode para algumas instâncias ($N = 5, 10, 20, 40, 80$ – como não conheço o número de instâncias plausíveis, creio que até $N = 20$ teremos respostas em tempos aceitáveis) e procure preencher a tabela com os diversos tipos de escolha de variáveis e de domínio. Siga como exemplo a tabela abaixo (referência: tabelas 1 e 2) como guia, e escolha uns 5 testes do quais voce entende que para este problema vai alcançar um bom e ruim desempenho. Veja no exemplo das n-rainhas, slides do professor, alguns parâmetros e variações.

Descreva a característica da máquina que fizeste os testes, bem como uma execução por vez é o suficiente, já que a idéia é a minimização.

1.1 Explorar o *search* do Minizinc

Estrutura geral do *search*:

```
solve :: int_search( [UM ARRAY AQUI] , SELECIONA_VARIAVEL, ESCOLHE_DOMINIO, complete)
      minimize ou maximize F_OBJETIVO;
```

isto vale para busca *booleana* e *float* (funciona tambmém!)

Quanto a Seleção de Variável:

```
anti_first_fail, dom_w_deg, first_fail, impact, input_order, largest, max_regret,
most_constrained, occurrence, smallest
```

Quanto a Escolha de Domínios:

`indomain`, `indomain_interval`, `indomain_max`, `indomain_median`, `indomain_middle`, `indomain_min`, `indomain_random`, `indomain_reverse_split`, `indomain_split`, `indomain_split_random`, `outdomain_max`, `outdomain_median`, `outdomain_min`, `outdomain_random`

Preencha a tabela abaixo com o maior N que encontrares.

Tabela 1: Tempos de execução com manipulação dos parâmetros do *search*. $N = ...$

Sel. Variável / Atr. Dominio	`first_fail`	`anti_first_fail`	`occurrence`
`indomain_min`	....	....	....
`indomain_max`	....	....	....
`indomain_median`	....	....	....
`indomain_split`	....	....	....
`indomain_random`	....	....	....
`outdomain_max`	....	....	....

Tabela 2: Continuação da tabela anterior. $N = ...$

Sel. Variável / Atr. Dominio	`most_constrained`	`max_regret`	`dom_w_deg`
`indomain_min`	....	....	....
`indomain_max`	....	....	....
`indomain_median`	....	....	....
`indomain_split`	....	....	....
`indomain_random`	....	....	....
`outdomain_max`	....	....	....

Conclua o experimento, respondendo:

1. Porquê ocorreram os piores e melhores resultados para este problema em específico?
2. O que poderia ser feito para ser melhorado?

Se quiser estender o experimento acima e fazer uma *verdadeiro experimento científico*[1], fique confortável e será contabilizado na nota.

[1] Aumentar `N`, variar heurísticas, etc.

2 Planejamento *Clássico*

O exercício abaixo é uma aplicação direta do *cumulative* (e/ou *disjuntive*) onde é para se fazer um escalonamento de um sistema de renovação de projeto (nada redundante, escalonar as atividades de um projeto, apenas isto!).

A duração de cada atividade, de `a` a `j`, com as restrições de precedência estão na tabela 2. Estas precedências estão ilustradas na figura 3.

Example 4.16: A simple project management problem
Consider a small renovation project with the following activities:

Activity	Description	Duration	Preceding Activities
a	Paper work and drafting	3	None
b	Manpower planning	4	a
b	Material planning	4	a
d	Transporting materials	2	c
e	Site preparation	4	a
f	Work Planning	6	c
g	Hiring equipment	3	d, b
h	Plan evaluation	1	e
i	Renovation work	12	f, g
j	Inspection and certification	2	i, h

Figura 2: 1a. Parte - 1/3

2.1 Uma Modelagem

Uma modelagem clássica segue das figuras 3 e 4.

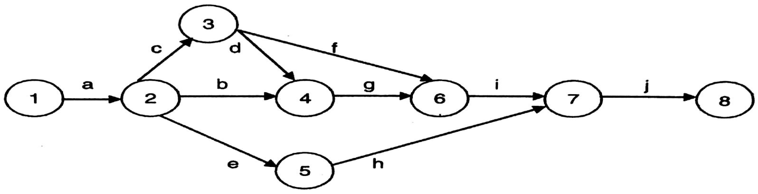

FIGURE 4.8
A project network.

The network of the above project can be drawn as Figure 4.8.

The activities other than on the longest path have flexibility in starting times as they can be started either at their earliest possible or latest possible time. By definition, the earliest possible start time for an activity to occur is immediately after all the preceding activities have been completed. The latest start time is the time that allows an activity to occur without causing a delay in the project-completion time. The LP model, for determining project duration, is usually developed considering one of the above two start times.

Defining variables:
Considering the earliest start times
T_k = earliest start time of activity k (k = a, b, . . . , j, where a, b, . . . , j represents all the activities).

Defining project data:
t_k = activity time of activity k (k = a, b, . . . , j).

Objective function:
The objective is to minimize the overall project completion time. It can be achieved by minimizing the sum of the earliest start times of all activities.

$$\text{Minimize } Z = T_a + T_b + T_c + \cdots + T_j \qquad (4.112)$$

Constraints:
The constraints represent only the precedence constraints.

At node 2:

$$T_b \geq T_a + t_a \qquad (4.113)$$
$$T_c \geq T_a + t_a \qquad (4.114)$$
$$T_e \geq T_a + t_a \qquad (4.115)$$

Figura 3: 2a. Parte - 2/3 – Um pouco distorcida mas creio que dá para ler

Mais alguns detalhes ....

2.2 Continuação da Modelagem

At node 3:

$$T_d \geq T_c + t_c \tag{4.116}$$
$$T_f \geq T_c + t_c \tag{4.117}$$

At node 4:

$$T_g \geq T_b + t_b \tag{4.118}$$
$$T_g \geq T_d + t_d \tag{4.119}$$

At node 5:

$$T_e \geq T_h + t_h \tag{4.120}$$

At node 6:

$$T_i \geq T_f + t_f \tag{4.121}$$
$$T_i \geq T_g + t_g \tag{4.122}$$

At node 7:

$$T_j \geq T_i + t_i \tag{4.123}$$
$$T_j \geq T_h + t_h \tag{4.124}$$
$$T_i \geq 0 \quad \text{for all activities} \tag{4.125}$$

After organizing the variables in the left-hand side of the constraints and the constants in the right-hand side, the final model becomes

$$
\begin{aligned}
&\textbf{Minimize } Z = T_a + T_b + T_c + \cdots + T_j \\
&\textbf{Subject to} \\
&\quad T_b - T_a \geq t_a \\
&\quad T_c - T_a \geq t_a \\
&\quad T_e - T_a \geq t_a \\
&\quad T_d - T_c \geq t_c \\
&\quad T_f - T_c \geq t_c \\
&\quad T_g - T_b \geq t_b \\
&\quad T_g - T_d \geq t_d \\
&\quad T_e - T_h \geq t_h \\
&\quad T_i - T_f \geq t_f \\
&\quad T_i - T_g \geq t_g \\
&\quad T_j - T_i \geq t_i \\
&\quad T_j - T_h \geq t_h \\
&\quad\quad T_i \geq 0 \quad \text{for all activities}
\end{aligned}
\qquad \textbf{Model (4.18)}
$$

After solving the model, the value of $T_j + t_j$ would provide the project duration.

Figura 4: 3a. Parte - 3/3

3 *Job Shop Scheduling* – Um pequeno Exemplo

Implementar um *Job Shop Scheduling Problem* (JSSP) clássico. O problema JSSP tem a seguinte descrição básica:

1. n *jobs* ou tarefas, m máquinas;
2. Cada um dos *jobs* ou tarefas tem um rota ou sequência de máquinas a serem passadas;
3. Nem todas as tarefas precisam passar por todas a máquinas;
4. Uma operação (i, j), leia-se processar um *job* j na máquina i;
5. Assuma que os *jobs* não circulam novamente. Se apresentam apenas um única vez;
6. Tempo de processamento é indicado por p_{ij} (i: da máquina e j: da tarefa, seria o *job* em inglês);
7. Objetivo é minizimizar o tempo de execução de todos os *jobs* respeitando as sequências individuais de cada *job*.
8. Preste a atenção no par (i, j), i: da máquina e j: da tarefa.

3.1 Problema 01

Tarefas(j)	Sequência em M_i	Tempo de Processamento
1	1,2,3	$p_{11} = 10$, $p_{21} = 8$, $p_{31} = 4$
2	2,1,4,3	$p_{22} = 8$, $p_{12} = 3$, $p_{42} = 5$, $p_{32} = 6$
3	1,2,4	$p_{13} = 4$, $p_{23} = 7$, $p_{43} = 3$

Para este exemplo tem-se:

- $1 \leq i \leq 3$ (máquinas)
- $1 \leq j \leq 4$ (tarefas)

O grafo de precedência é dado pela figura 5.

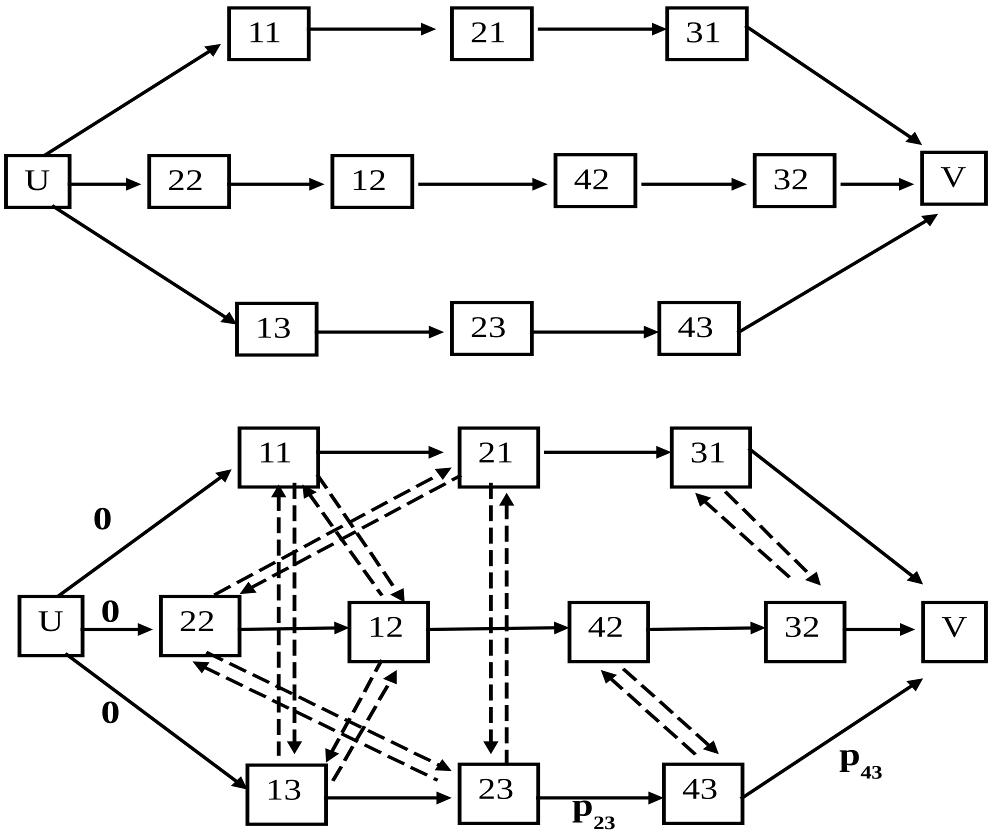

Figura 5: Grafo de Precedência e ***algumas*** restrições disjuntivas (flechas tracejadas – arcos disjuntivos)

Observe na figura 5 que as tarefas 1 e 3 não passam por todas as máquinas.

Este problema tem a descrição matemática em um arquivo doc aqui no GitHub (peguei este exemplo dele), tem a solução de um gráfico de Gantt etc. Não peguem soluções prontas do Minizinc ou do Hakank, faça a sua e depois.

Para usar o *cumulative* e *disjunctive* no Minizinc

```
include "cumulative.mzn";
ou
include "globals.mzn";
```

3.2 Problema 02

Idêntico ao problema anterior, mas agora só estamos passando o arquivo de entrada em Minizinc.

```
n = 3; %% Numero de JOBS
m = 4; %% Numero de MAQUINAS

%% Matriz de duração de cada tarefa em cada máquina
    %  J1 J2 J3 J4
d = [| 3, 3, 4, 4   % M1
     | 4, 3, 2, 2   % M2
     | 3, 3, 3, 4   % M3
     |];

%% Matriz de sequencia de maquina por tarefa

seq=[| 1, 2, 3, 4  % J1 faz a sequencia: M1->M2->M3->M4
     | 1, 3, 2, 4  % J2 faz a sequencia: M1->M3->M4->M4
     | 4, 2, 1, 3  % J3 faz a sequencia: M4->M2->M1->M3
     |];
```

Diferente do problema anterior, aqui todas as tarefas passam por todas as máquinas.

4 Considerações Finais:

- ➭ Faça vários testes. Em geral ninguém faz, mas, é para fazer vários testes de I/Os
- ➭ Assuma e justifique os dados que faltarem.