U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Programação em Lógica com Restrições

Novembro de 2019

Parcialmente baseado em slides anteriores de
Henrique Lopes Cardoso (hlc@fe.up.pt),
Luís Paulo Reis (lpreis@fe.up.pt),
Pedro Barahona, John Hooker,
Willem-Jan van Hoeve
e outros autores

Daniel Castro Silva
dcs@fe.up.pt

18/11/2019

Conteúdo

1. Introdução à Programação com Restrições
2. Exemplos de Problemas de Satisfação e Otimização
3. Exemplos de Resolução de Problemas em PLR
4. Complexidade e Métodos de Pesquisa
5. Definições Formais e Conceitos
6. Manutenção de Consistência
7. Pesquisa, Otimização e Eficiência
8. Sistemas de PR e PLR
9. Conclusões e Leituras Adicionais

Programação em Lógica com Restrições

1. INTRODUÇÃO À PROGRAMAÇÃO COM RESTRIÇÕES

Bibliografia base:

Edward Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London and San Diego, 1993

Kim Marriott e Peter J. Stuckey. *Programming with Constraints: an Introduction*, MIT Press, 1998

Programação com Restrições

"Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it."

Eugene Freuder, 1997 ('In Pursuit of the Holy Grail', Constraints: An International Journal, 2, 57-61)

Programação em Lógica com Restrições

- A Programação em Lógica com Restrições – ***PLR***, ou ***CLP*** (*Constraint Logic Programming*) – é uma classe de linguagens de programação combinando:
  - Declaratividade da programação em lógica
  - Eficiência da resolução de restrições
- Aplicações principais na resolução de problemas de **pesquisa / otimização combinatória** (tipicamente NP-completos):
  - Escalonamento (*scheduling*), geração de horários (*timetabling*), alocação de recursos, planeamento, gestão da produção, verificação de circuitos, …
- Vantagens:
  - Clareza e brevidade dos programas
  - Reduzido tempo de desenvolvimento
  - Facilidade de manutenção
  - Eficiência na Resolução

Contexto Histórico

- Anos ’60 e ’70: Alguns desenvolvimentos prévios em tópicos e aplicações relacionadas
- Anos ’70: Prolog (‘*PROgrammation en LOGique*’)
  - Roussel, Colmerauer, Kowalsky et al.
- Anos ’80: Programação (em Lógica) com Restrições
  - Prolog III, CHIP, …
  - *Constraint & Generate* vs *Generate & Test*
- Anos ’90: Programação por restrições, primeiros *solvers* industriais (ILOG, …) e aplicações
- Anos 2000: Restrições globais, linguagens de modelação, …

Aplicações Industriais / Comerciais

- Aplicações usando programação com restrições:
  - Lufthansa
    - Escalonamento de pessoal a curto prazo
  - Renault
    - Planeamento de produção a curto prazo
  - Nokia
    - Configuração de software para telemóveis
  - Airbus
    - Layout de cabine
  - Siemens
    - Verificação de circuitos
  - Ver Helmut Simonis (1999), Building Industrial Applications with Constraint Programming, in Constraints in Computational Logics (CCL 1999)

Aplicações Industriais / Comerciais

- Escalonamento desportivo
  - Vários tipos de ligas / torneios

- 1997/1998 *ACC basketball league* (9 equipas)
  - Várias restrições laterais complicadas
  - As 179 soluções existentes foram encontradas em 24h usando enumeração e programação linear inteira [Nemhauser & Trick, 1998]
  - As 179 soluções foram encontradas em menos de um minuto usando programação com restrições [Henz, 1999, 2001]

Aplicações Industriais / Comerciais

- Escalonamento de operações

- Aeroporto de Hong Kong
  - Alocação de *gates* no Aeroporto Internacional de Hong Kong
  - Sistema foi implementado em apenas quatro meses, e inclui tecnologia de programação com restrições (ILOG)
  - Faz o escalonamento de ~1100 voos por dia (mais de 70 milhões de passageiros em 2016)

Aplicações Industriais / Comerciais

- Escalonamento de operações

- Porto de Singapura
  - Um dos maiores centros de transbordo do mundo
  - Faz a ligação a uma rede de 200 empresas de expedição com ligações a 600 portos em 123 países
  - Necessidade de atribuir localizações e planos de carga / descarga sob restrições operacionais e de segurança
  - Sistema de planeamento, baseado em programação com restrições (ILOG)

Aplicações Industriais / Comerciais

- Escalonamento de operações

- Netherlands Railways
  - Uma das redes ferroviárias mais densas do mundo, com mais de 5.500 comboios por dia
  - Programação com restrições é uma das partes do software de planeamento, usado para obter um novo horário a partir do zero (2008)
  - Escalonamento mais robusto e eficiente, leva a lucro anual adicional de 75M
  - Vencedor do prémio INFORMS Edelman (2008)

Problema de Satisfação de Restrições

- Um Problema de Satisfação de Restrições – ***PSR***, ou ***CSP*** (*Constraint Satisfaction Problem*) - é modelizado através de:
  - ***Variáveis*** representando diferentes aspetos do problema, juntamente com os seus ***domínios***
  - ***Restrições*** que limitam os valores que as variáveis podem tomar dentro dos seus domínios

- Porquê representar problemas como CSPs?
  - Representação muito próxima da original
  - Clareza e brevidade dos programas
  - Rapidez no desenvolvimento de programas
  - Maior garantia de correção dos programas
  - Algoritmos para resolver CSPs são muito eficientes

Solução de um CSP

- A ***solução*** de um CSP é uma atribuição de um valor (do seu domínio) a cada variável, de forma a que todas as restrições sejam satisfeitas
- Podemos estar interessados em:
  - Descobrir ***se*** o problema tem solução
  - Encontrar unicamente ***uma*** solução
  - Encontrar ***todas*** as soluções do problema
  - Encontrar uma solução ***ótima***, de acordo com uma função objetivo, definida em termos de algumas ou de todas as variáveis
- A solução de um CSP é encontrada através de uma ***pesquisa*** sistemática, usualmente guiada por uma heurística, através de todas as atribuições possíveis de valores às variáveis

Definição Formal de um CSP

- Mais formalmente, um CSP é um tuplo $\langle V, D, C \rangle$:
  - $V = \{x_1, x_2, \ldots, x_n\}$ é o conjunto de variáveis de domínio
  - D é uma função que mapeia cada variável de V num conjunto de valores – os domínios das variáveis
    - $D: V \rightarrow conjunto\ finito\ de\ valores$
    - $D_{x_i}: domínio\ de\ x_i$
  - $C = \{C_1, C_2, \ldots, C_n\}$ é o conjunto de restrições que afetam um subconjunto arbitrário de variáveis de V
    - Para cada restrição $c_i \in C$
      - $Vars(c_i)$ é o conjunto de variáveis envolvidas em c_i
      - Simetricamente, $Cons(x_i)$ é o conjunto de restrições nas quais a variável $x_i \in V$ está envolvida

Definição Formal de um CSP

- Solução de um CSP: atribuição de um valor a cada variável de forma a que todas as restrições sejam respeitadas:
  - $Sol = \{\langle x_1, v_1\rangle, \langle x_2, v_2\rangle, \ldots, \langle x_n, v_n\rangle\}$:
    - $\forall x_i \in V \left(\langle x_i, v_i\rangle \in Sol \ \wedge \ v_i \in D_{x_i}\right)$
    - $\forall c_k \in C \ satisfaz \ (Sol, c_k)$

Problema de Otimização com Restrições

- Um Problema de Otimização com Restrições – ***POR***, ou ***COP*** (*Constraint Optimization Problem*) – é um CSP com a adição de uma função objetivo a ser otimizada (maximizada / minimizada)

- A solução de um COP é uma atribuição de um valor (do seu domínio) a cada variável, de forma a que todas as restrições sejam satisfeitas e não haja nenhuma outra atribuição que resulte num valor superior (/ inferior) da função de avaliação

Restrições em CSPs

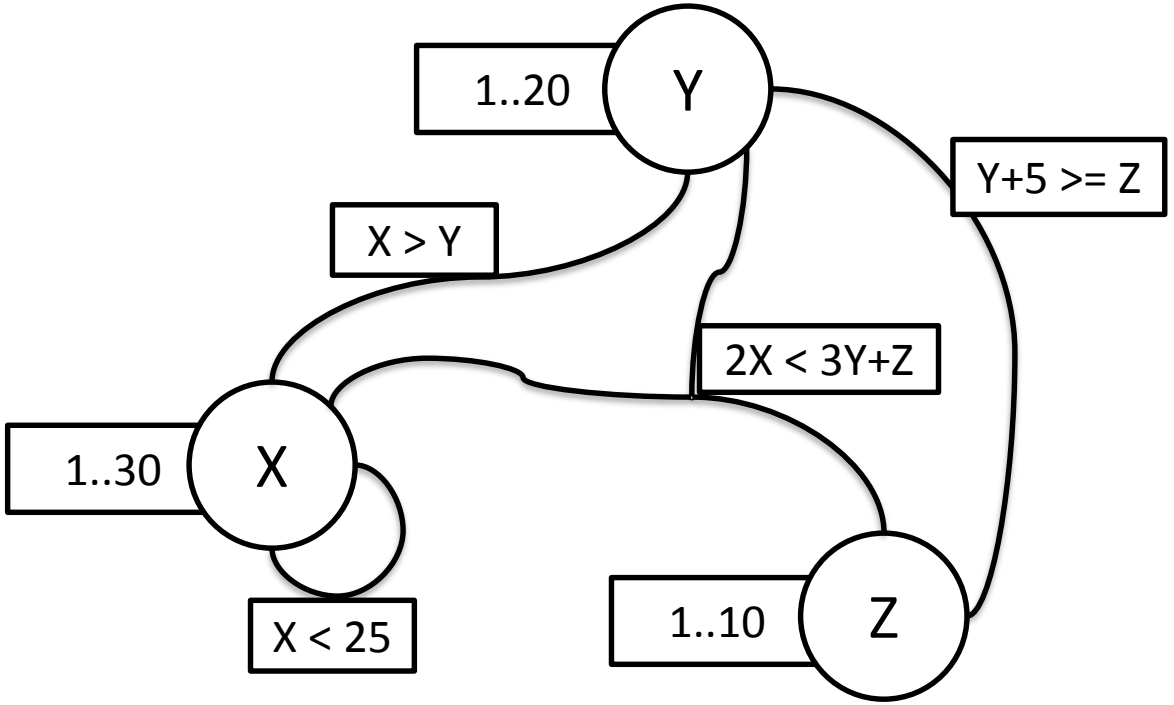
- Tipos de Restrições:
  - Unárias
    - X > 10
    - X ∈ {1,2,5}
  - Binárias
    - X < Y
    - Y+20 >= X
  - Qualquer restrição de aridade superior pode ser transformada num conjunto de restrições binárias
  - Logo, os CSPs binários podem ser considerados como representativos de todos os CSPs

- Domínios:
  - Números complexos
  - Reais
  - Racionais
  - **Inteiros**
  - Booleanos

- Tipos Simples de Restrições:
  - Lineares
    - 2*X + 4 < 4*Y + Z
  - Não Lineares
    - W*Z + 3*X > Y - Z*Z

Restrições em CSPs

- Um CSP pode ser visto como um híper-grafo, em que os nós representam variáveis (com domínios associados), e as restrições são (híper-)arcos que ligam os nós



Restrições

- As restrições de um CSP/COP podem ser:
  - **Rígidas** (*Hard Constraints*): são aquelas que têm obrigatoriamente de ser cumpridas
    - Todas as restrições num CSP são restrições rígidas
  - **Flexíveis** (*Soft Constraints*): são aquelas que podem ser quebradas
    - Deve existir um custo associado a quebrar este tipo de restrições, o qual deve ser somado na função de avaliação do COP
    - Restrições flexíveis permitem relaxamento
      - Aumenta complexidade do modelo

Resolução de um CSP

- Passos envolvidos:
  - Declarar as ***variáveis*** e os seus ***domínios*** (finitos)
  - Especificar as ***restrições*** existentes
  - ***Pesquisar*** para encontrar a solução

Resolução de um CSP

- Técnicas utilizadas na resolução:
  - Determinísticas: técnicas de ***consistência***
  - Não determinísticas: ***pesquisa*** (“search”)

- Forma de resolução:
  - Computação determinística é efetuada sempre que possível
    - Durante a propagação, sempre que é colocada uma restrição
  - Pesquisa é efetuada quando não é possível efetuar mais propagação

Mecanismo *Constrain and Generate*

- O mecanismo de ***unificação*** do Prolog é substituído por um mecanismo de ***manipulação de restrições*** num dado domínio

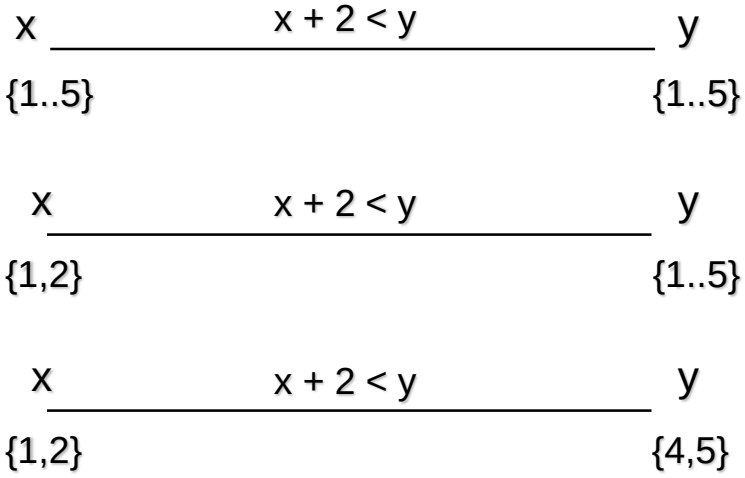
- A pesquisa do tipo "***generate and test***" do Prolog (pouco eficiente) é substituída por técnicas mais inteligentes de pesquisa (técnicas de consistência), resultando num mecanismo do tipo "***constrain and generate***"

Propagação e Consistência

- Se existir uma restrição binária C_{ij} entre as variáveis X_i e X_j, então o arco (X_i, X_j) é consistente se para cada valor a $\in D_{X_i}$, existe um valor b $\in D_{X_j}$, tal que as atribuições $X_i = a$ e $X_j = b$ satisfaçam a restrição C_{ij}.

- Qualquer valor a $\in D_{X_i}$ para o qual isto não se verifique pode ser removido de D_{X_i} (pois não pode fazer parte de nenhuma solução consistente)

- A remoção de todos estes valores torna o arco (X_i, X_j) consistente

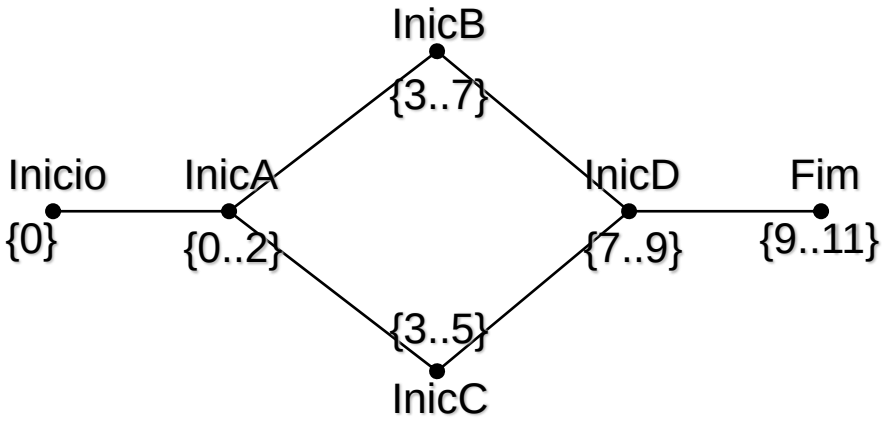


- Se todos os arcos de um CSP binário forem tornados consistentes, então todo o problema é consistente nos arcos

Propagação e Consistência

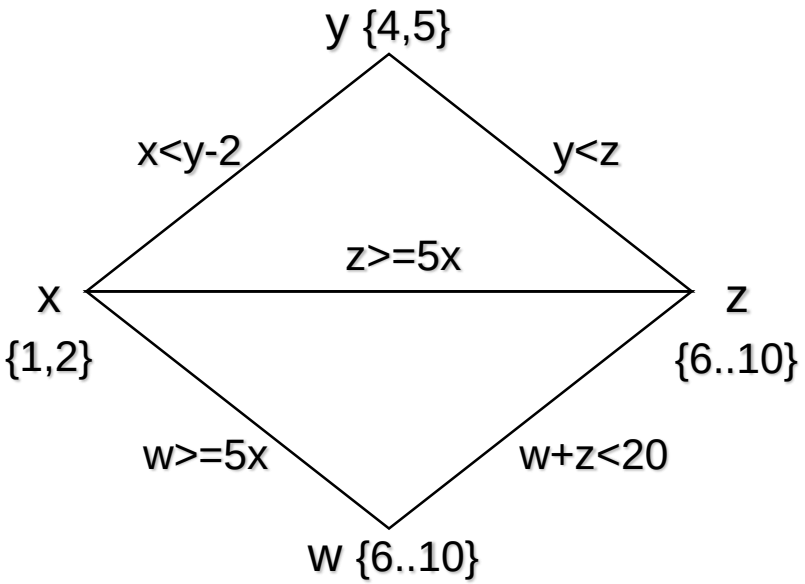
Tarefa	Duração	Precede
A	3	B,C
B	2	D
C	4	D
D	2	

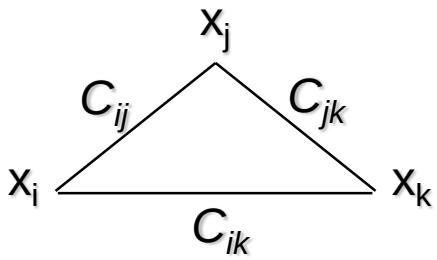
Variável	Domínio
Inicio	{0}
InicA	{0..11}
InicB	{0..11}
InicC	{0..11}
InicD	{0..11}
Fim	{0..11}



Propagação e Consistência

- Problema é consistente nos arcos, mas será que é possível x=2?
- Próximo passo: considerar triplos de variáveis em que dois pares de variáveis têm uma restrição não trivial entre eles





- Caminho (X_i, X_j, X_k) é consistente nos caminhos se para cada par de valores $v_i \in D_{X_i}$ e $v_k \in D_{X_k}$, permitidos pela restrição C_{ik}, existe um valor $v_j \in D_{X_j}$ tal que (v_i, v_j) satisfaz C_{ij} e (v_j, v_k) satisfaz C_{jk}
- Se não existir, então (v_i, v_k) deve ser removido da restrição C_{ik}

Programação em Lógica com Restrições

2. EXEMPLOS DE PROBLEMAS DE SATISFAÇÃO E OTIMIZAÇÃO

Adaptado de:
Pedro Barahona, *Página da Disciplina de Programação por Restrições, Ano Lectivo 2003/04,* disponível em:
http://ssdi.di.fct.unl.pt/~pb/cadeiras/pr/0304/index.htm
[consultado em Setembro de 2004]

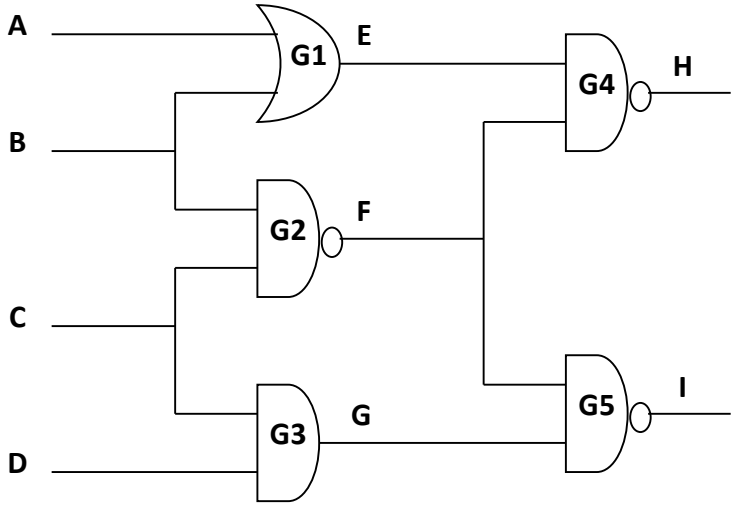
Restrições: Exemplos de Problemas

- Planeamento de Testes em Circuitos Digitais
  - Deteção de Avarias
- Gestão de Tráfego em Redes
- Gestão da Produção
- Escalonamento de Tarefas (*scheduling*)
- Geração de Horários (*timetabling*)
- Caixeiro Viajante
  - Simples ou Múltiplo
- Colocação de Objetos
  - 2D: Corte de peças (tecido, vidro, madeira, etc.)
  - 3D: Preenchimento de contentores / Empacotamento

Circuitos Digitais

- Planeamento de testes para deteção de avarias
  - Objetivo:
    - Determinar um padrão de entrada que permita detetar se uma "*gate*" está avariada
  - Variáveis:
    - Modelizam o valor do nível elétrico (*high*/*low*) nos vários fios do circuito
  - Domínios:
    - Variáveis Booleanas: 0/1.
  - Restrições:
    - Restrições de igualdade entre o sinal de saída e o esperado pelo funcionamento das "*gates*"

Circuitos Digitais



- `E = A ⊕ B ⊕ A·B` `% E = or(A,B)`
- `F = 1 ⊕ B·C` `% F = nand(B,C)`
- `G = C·D` `% G = and(C,D)`
- `H = 1 ⊕ E·F` `% H = nand(E,F)`
- `I = 1 ⊕ F·G` `% I = nand(F,G)`

(⊕ representa *xor*; · representa *and*)

Gestão de Tráfego em Redes

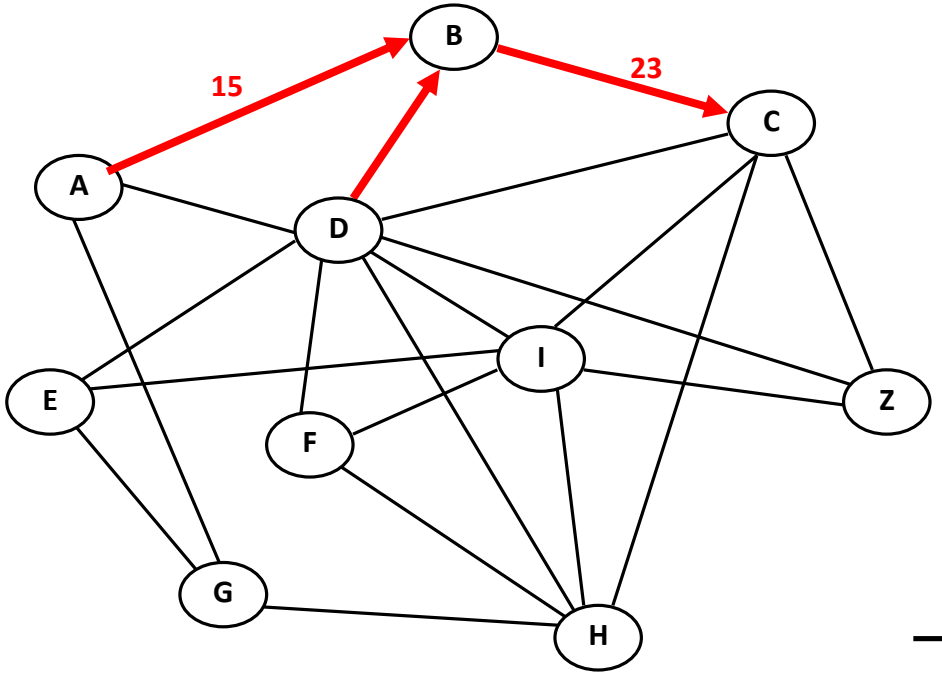
- Determinação do tráfego em redes de telecomunicações, transporte de água/eletricidade, …
  - Objetivo:
    - Determinar a quantidade de informação que flui em cada um dos troços de uma rede de comunicações
  - Variáveis:
    - Modelam o fluxo nos diversos troços
  - Domínio:
    - Reais não negativos
  - Restrições:
    - Limites de capacidade dos troços, não perda de informação nos vários nós

Gestão de Tráfego em Redes

- X_{ij} é o fluxo entre os nós *i* e *j*
  - Restrições de capacidade

    $X_{ab} \leq 15$
  - Restrições de manutenção de informação

    $X_{ab} + X_{db} = X_{bc}$



Gestão da Produção

- Determinação das quantidades ideais de itens a produzir
  - Objetivo:
    - Determinar as quantidades de itens que devem ser produzidos
  - Variáveis:
    - Modelizam o número de exemplares de cada produto
  - Domínio:
    - Inteiros / Reais não negativos
  - Restrições:
    - Limites dos recursos existentes, restrições sobre o plano produzido

Gestão da Produção

- Limites dos recursos existentes
  - R_i é a quantidade de unidades do recurso i

$$a_{i1} X_1 + a_{i2} X_2 + a_{i3} X_3 + ... \leq R_i$$

    - a_{ij} é a quantidade do recurso i necessária para produzir uma unidade do produto j
    - X_j é a quantidade do produto j produzida

- Restrições sobre o plano produzido
  - Objetivos mínimos de produção

$$X_1 + X_2 \geq 50$$

  - Equilíbrio na produção

$$| X_4 - X_5 | < 20$$

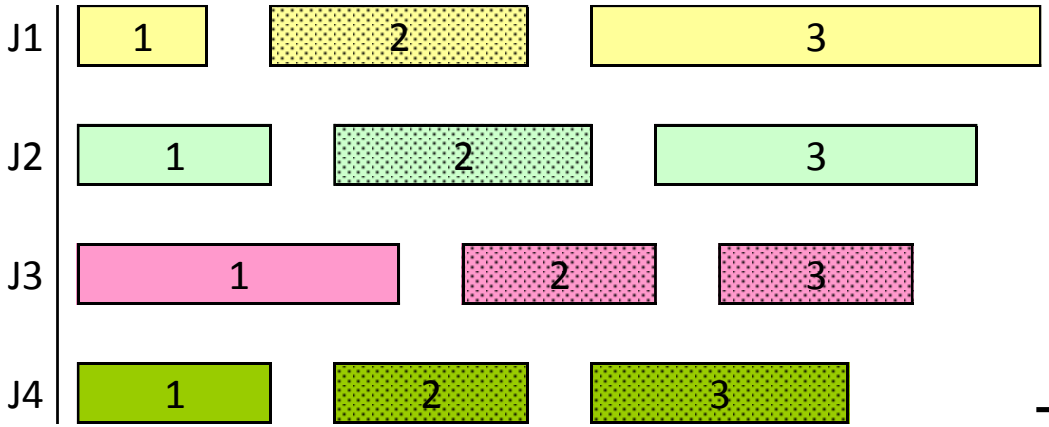
Escalonamento de Tarefas

- Determinação do escalonamento de um conjunto de tarefas no tempo (eventualmente espaço)
  - Objetivo:
    - Determinar os tempos de início de um conjunto de tarefas (eventualmente os recursos utilizados também)
  - Variáveis:
    - Modelam o início da execução das tarefas
  - Domínio:
    - Inteiros/Reais não negativos
  - Restrições:
    - Não sobreposição de tarefas que utilizam os mesmos recursos
    - Precedências de tarefas

Escalonamento de Tarefas

- Job-Shop
  - S_{ij} / D_{ij} / M_{ij}: início/duração/máquina da tarefa i para o trabalho j
  - Precedência entre tarefas: $S_{ij} + D_{ij} \leq S_{kj}$ para $i < k$
  - Não sobreposição de tarefas na mesma máquina

$$(M_{ij} \neq M_{kl}) \vee (S_{ij} + D_{ij} \leq S_{kl}) \vee (S_{kl} + D_{kl} \leq S_{ij})$$



Geração de Horários

- Especificação dos horários (e.g. escolares ou universitários) de um conjunto de eventos/tarefas
  - Objetivo:
    - Determinar o início das várias aulas/eventos num horário
  - Variáveis:
    - Modelam o tempo de início da aula, e eventualmente a sala, o professor, outros recursos, etc.
  - Domínio:
    - Finitos (horas certas) para os tempos de início
    - Finitos para as salas/recursos/professores
  - Restrições:
    - Não sobreposição de aulas na mesma sala , nem com o mesmo professor, respeito pelas impossibilidades, etc.

Caixeiro Viajante

- Determinação de percursos de caixeiros viajantes (frotas de carros, empresas de distribuição, etc.)
  - Objetivo:
    - Determinar o melhor (mais curto/mais rápido) caminho para ser seguido pelos veículos de uma empresa de distribuição
  - Variáveis:
    - Ordem em que cada localidade é atingida
  - Domínio:
    - Finitos (número de localidades existentes)
  - Restrições:
    - Não repetir localidades, garantir ciclo global, não existência de sub-ciclos

Empacotamento

- Colocação de componentes no espaço (2D/3D) sem sobreposição
  - Objetivo:
    - Determinar formas de conseguir colocar componentes num dado espaço
  - Variáveis:
    - Coordenadas (X,Y) dos elementos
    - Rotações/espelhamentos dos elementos
  - Domínio:
    - Reais / Finitos (grelha)
  - Restrições:
    - Não sobrepor componentes, não ultrapassar os limites do "contentor"

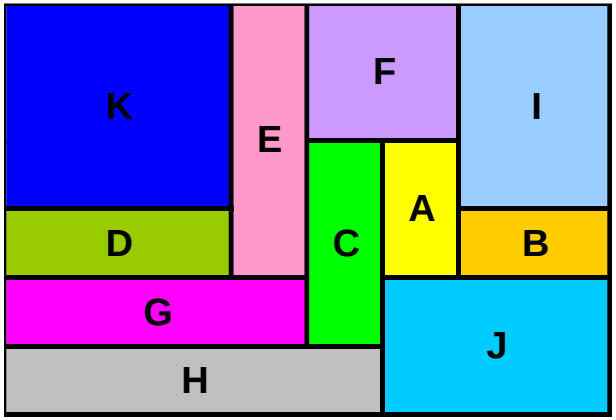
Empacotamento

- Não ultrapassar os limites do "contentor"

$$X_a + Lx_a \leq X_{max} \qquad Y_a + Ly_a \leq Y_{max}$$

- Não sobrepor componentes

$$X_a + Lx_a \leq X_b \vee X_b + Lx_b \leq X_a$$
$$Y_a + Ly_a \leq Y_b \vee Y_b + Ly_b \leq Y_a$$



Decisão vs. Otimização

- Em certos (muitos) casos o que se pretende determinar não é uma solução qualquer para o problema, mas sim a *melhor solução* (segundo um dado critério)
  - Necessário definir uma função de avaliação

Otimização: Exemplos

- Planeamento de Testes em Circuitos Digitais
  - Teste com menos entradas especificadas
- Gestão de Tráfego em Redes
  - Tráfego com menor custo / mais curto
- Gestão da Produção
  - Plano com maior lucro / menores custos
- Escalonamento de Tarefas
  - Solução com o final mais cedo
- Geração de Horários
  - Solução com menos furos
  - Solução que melhor respeita preferências de horário
- Caixeiro Viajante
  - Solução com menor distância/custo percorrida
- Empacotamento ou Corte
  - Corte / Colocação do maior número de peças
  - Menor desperdício de material