Lógica Computacional 2025-2026 - Trabalhos Práticos

Avaliação contínua

1. Os alunos participam na avaliação contínua integrados em grupos. A constituição dos vários grupos de trabalho é feita individualmente por cada aluno nesta folha de cálculo.
2. Cada grupo contém 2 alunos ou excepcionalmente 3. Nos grupos com 3 alunos, a nota do trabalho é penalizada em 5%. Devido ao elevado número de inscrições não é possível existir participação individual não integrado num grupo.
3. A ferramenta fundamental para as aulas e os trabalhos práticos é a linguagem Python e o ecosistema de "packages" a ela associadas. Recomenda-se que a instalação desses elementos seja feita via Anaconda, com as "packages" OR-Tools,. Z3-solver (ou PySMT).
4. Complementar e opcionalmente pode-se usar Julia e as packages JuMP , Satisfiability e JuliaGraphs via juliaup.
5. Em alternativa às soluções locais pode-se usar uma solução na "cloud" como o Google Colab que suporta ambos os ecosistemas Python e Julia . Esta opção exige porém que as "packages" usadas tenham de ser carregadas em cada execução de cada programa.
6. Os trabalhos têm a forma de um "notebook" (Jupyter, VSCode ou Colab) distinto para cada um dos problemas indicados. Cada notebook deve
   a. descrever o problema e a abordagem usada para o resolver,
   b. apresentar o código Python/Julia que resolve o problema,
   c. apresentar exemplos e testes de aplicação realistas que testem a correção do código, a sua eficiência computacional e a capacidade de escalar para grandes problemas.
7. A entrega do trabalho tem a forma de uma discussão oral de 30 minutos com todos os elementos do grupo, e inclui a demonstração da boa execução do código.
8. A avaliação do trabalho incide sobre os três items referidos no nº 6 e ainda a discussão oral referido no nº 7.
9. Os "notebooks" Jupyter executáveis (ou o link no caso do Colab) e uma cópia PDF de cada um, devem ser previamente enviados via e-mail ao responsável da disciplina (@José Manuel V) até às 23:59 da véspera da 1ª data de entrega

desse trabalho.
10. A classificação dos trabalhos é específica de cada elemento do grupo segundo a perceção que o avaliador tem da contribuição de cada um para o trabalho apresentado. Os resultados constam da seguinte folha de cálculo.
11. A entrega dos trabalhos realiza-se nas datas abaixo indicadas. A inscrição no horário de entrega é feita pelos grupos na folha de cálculo referida em 1.
12. Informação complementar sobre a disciplina pode ser vista nesta diretoria.

		Observações
TP1	7 e 9 de Outubro 2025	+Capítulo 2: Programação com Restrições
TP2	11 e 13 de Novembro 2025	
TP3	9 e 11 de Dezembro 2025	
TP4	06 de Janeiro 2026	
Exame de Recurso		

Exame de recurso

Por escolha própria um aluno pode realizar um exame individual de acordo com as regras seguintes.

1. O resultado do exame substitui a nota de **um** dos TP's.
2. O exame de recurso realiza-se on-line no dia definido no calendário de exames de acordo com os seguintes procedimentos.
   a. Os alunos que pretendam realizar o exame devem comunicar essa intenção ao coordenador da disciplina por e-mail não depois do 3º dia útil anterior à data do exame.
   b. No dia do exame será criada uma sessão de "online" que decorre das 09:00 até às 13:00. abertas aos alunos examinados.
   c. No dia do exame às 09:00 será enviada por e-mail a cada aluno examinada um problema que segue o formato usado nos trabalhos práticos.
   d. A resposta ao problema deve consistir em um "notebook" Jupyter contendo uma solução executável e uma descrição sucinta desse código. Valoriza-se um código bem comentado.
   e. A prova escrita é entregue por e-mail, para o endereço do coordenador da disciplina, até às 16:00 do dia de exame.

f. A prova oral é a discussão do trabalho, no formato usado na apresentação dos trabalhos práticos, em “slots” de 30 a 45 minutos definidos pelo responsável da disciplina. As provas realizam-se via uma sessão Zoom a partir das 16:30 do dia do exame.

Trabalho Prático 1

Exercício 1

Este problema usa optimização MIP (“Mixed Integer Programming” (OrTools) e representação por Grafos (NetworkX).

1. Para um distribuidor de encomendas o seu território está organizados em pontos (“nodes”) de fornecimento (“sources”), pontos de passagem e pontos de entrega (“sinks”) ligados por vias de comunicação (“edges”) bidirecionais cada uma das quais associada uma capacidade em termos do número de veículos de transporte que suporta.
2. Os items distribuidos estão organizados em “pacotes” de três tipos “standard” : uma unidade, duas unidades e cinco unidades. Os pacotes são transportados em veículos todos com a capacidade de 10 unidades. Cada ponto de fornecimento tem um limite no número total de unidades que tem em “stock” e um limite no número de veículos que dispõe.
3. Cada encomenda é definida por o identificador do ponto de entrega e pelo número de pacotes, de cada um dos tipos, que devem ser entregues nesse ponto.
4. O objetivo do problema é decidir, a partir de uma encomenda e com um mínimo no número de veículos,
    - *em cada ponto de fornecimento, se estará envolvido no fornecimento de unidades que essa encomenda requer sem violar os limites do seu “stock”.*
    - *em cada ponto de fornecimento,   como empacotar as unidades disponíveis, de acordo com a encomenda”, e como as distribuir por veículos,*
    - *em cada veículo, qual o percurso a seguir até ao ponto de entrega; para cada via ao longo de cada percurso, o total de veículos não pode exceder a capacidade dessa via.*

Efectue um (ou mais!) dos seguintes exercícios

Exercício 2

Este problema deve usar a optimização CP ("Constraint Programming") no OrTools e procura implementar soluções de uma generalização do problema Sudoku.

A definição usual do problema Sudoku (extraido da Wikipedia) contém a seguinte definição

> *In classic Sudoku, the objective is to fill a 9 × 9 grid with digits so that each column, each row, and each of the nine 3 × 3 subgrids that compose the grid (also called "boxes", "blocks", or "regions") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.*

Neste trabalho pretende-se generalizar o problema em várias direções:

- Em primeiro lugar a grelha tem como parâmetro fundamental um inteiro que toma vários valores $n \in \{3, 4, ...\}$. Fundamentalmente a grelha passa de um quadrado com $n^2 \times n^2$ células para um cubo tridimensional de dimensões $n^2 \times n^2 \times n^2$. Cada posição na grelha é representada por um triplo de inteiros $(i, j, k) \in \{1..n^2\}^3$.
- Em segundo lugar as "regiões" que a definição menciona deixam de ser linhas, colunas e "sub-grids" para passar a ser qualquer "box" genérica com um número de células $\leq n^3$. Cada "box" é representado por um dicionário D que associa, no estado inicial, cada posição $(i, j, k) \in D$ na "box" a um valor inteiro no intervalo $\{0..n^3\}$.
- Na inicialização da solução as células associadas ao valor 0 estão livres para ser instanciadas com qualquer valor não nulo. Se nessa fase, uma célula está associada a um valor não-nulo, então esse valor está fixo e qualquer solução do problema não o modifica.
- A solução final do problema, tal como no problema original, verifica uma restrição do tipo **all-different** que, neste caso, tem a forma

  > *dentro uma mesma "box", todas as células têm valores distintos no intervalo $\{1..n^3\}$.*

- Considera-se neste problema suas formas básicas de "boxes":
  - "cubos" de n^3 células determinados pelo seu vértice superior, anterior, esquerdo
  - "paths" determinados pelo seu vértice de início, o vértice final e pela

ordem entre os índices dos vértices sucessivos.

O "input" do problema é um conjunto de "boxes" e um conjunto de alocações de valores a células.

Exercício 3

Este exercício usa exclusivamente "programação inteira" (IP)

Na criptografia pós-quântica os *reticulados inteiros* ("*hard lattices*") e os problemas a eles associados são uma componente essencial. Um reticulado inteiro pode ser definido por uma matriz $\mathsf{H} \in \mathbb{Z}^{m\times n}$ (com $m \geq 2\,n$) e por um inteiro <u>primo</u> $q \gg m$.

Para além dos parâmetros n, m, q , o "input" do problema é definido pelos inteiros $\mathsf{H}_{j,i}$ gerados de forma aleatória, independente e uniforme num intervalo $\{0..q-1\}$.

> Se se pretender uma solução relativamente rápida os valores típicos dos parâmetros serão $n \simeq 25$, $m \simeq 50$, $q \simeq 1000$. Em soluções criptográficas, quando se pretende que o problema seja realmente difícil, os valores dos parâmetros estão pelo menos uma ordem de grandeza acima.

O chamado *problema do vetor mais curto* (SVP) consiste no cálculo de um vetor de inteiros $e \in \{0,1\}^m$ <u>não nulo</u> que verifique as seguintes relações

$$\exists\, j < m \,\centerdot\, e_j \neq 0 \qquad ,$$
$$\forall\, i < n \,\centerdot\, \sum_{j<m} e_j \times \mathsf{H}_{j,i} \;\equiv\; 0 \mod q$$

Pretende-se determinar, em primeiro lugar, se existe uma solução e . Se e existir pretende-se determinar e que minimiza o número de componentes não nulas.

> *Um inteiro x verifica $x \equiv 0 \mod q$ sse x é um múltiplo de q. Portanto*
> $$x \equiv 0 \mod q \qquad \text{sse} \qquad \exists\, k \in \mathbb{Z} \,\centerdot\, x \;=\; q \times k.$$
> *Escrita de forma matricial, a restrição que condiciona o vetor $e \neq 0$ será*
> $$\exists\, e \in \{0,1\}^m \,\centerdot\, \exists\, k \in \{0..m\}^n \,\centerdot\, \forall\, i < n \,\centerdot\, \sum_{j<m} e_j\,\mathsf{H}_{j,i} \;=\; q\,k_i$$

Exercício 4

Este exercício é um problema de optimização não linear.

De novo em criptografia pós-quântica é importante uma segunda classe de problemas designada por “Closest Vector Problem” (CVP). Este tipo de problema define-se sobre um outro tipo de reticulados, designados por *reticulados euclidianos,* determinados por um conjunto de vetores de componentes reais designados por geradores

$$\mathsf{G} \,\equiv\, \{\, g_1, g_2, \cdots, g_k \mid g_i \in \mathbb{R}^m \,\}$$

linearmente independentes. O reticulado definido por G é o conjunto de todas as combinações lineares inteiras dos vetores g_i

$$\mathcal{L}_{\mathsf{G}} \;\equiv\; \{\, x \in \mathbb{R}^m \mid x = z_1\, g_1 + z_2\, g_2 + \cdots + z_\kappa\, g_\kappa \,\wedge\, z_i \in \mathbb{Z} \,\}$$

O problema do CVP define-se dando um vetor aleatório $t \in \mathbb{R}^m$ e tentando calcular o vetor do reticulado $x \in \mathcal{L}_{\mathsf{G}} \;=\; \sum_{i=1}^{\kappa}\, z_i\, g_i$ que está mais próximo do “target” t. Aqui proximidade define-se pela distância Euclidiana no espaço $\mathbb{R}^m$; isto é

$$\mathsf{dist}(x,t) \;\equiv\; \sum_{j=1}^{m}\, (x_i - t_i)^2$$

Neste exercício vamos considerar uma versão particular deste problema, caracterizada pelas seguintes restrições

- Os elementos $\{g_{i,j}\}$ do vector g_i assim como os elementos t_j do “target” t representam probabilidades; por isso são sempre valores reais no intervalo $0 \leq g_{i,j}\,,\, t_j \leq 1$ que, aqui, são gerados aleatoriamente.
- Existe um limite ℓ positivo não-nulo nos coeficientes inteiros z_i tal que, para todo $i = 1..\kappa$ se verifica $-\ell \leq z_i \leq \ell$

Valores típicos dos parâmetros m, κ (com $\kappa = m - n$) são análogos aos do problema anterior. Um caso particular com interesse é $\ell = 1$.