# Constraint Programming
# (Programação com Restrições)

# O que é Constraint Programming

Imagine que você é um organizador de eventos tentando montar a agenda de um festival. Você precisa decidir quais shows vão em quais palcos e horários, mas há regras: alguns shows não podem acontecer ao mesmo tempo, cada palco só pode ter um show por vez, e certos artistas precisam de horários específicos. Constraint Programming (CP), ou Programação com Restrições, é uma técnica para resolver esse tipo de problema, onde você define regras (restrições) e o computador encontra uma solução que as satisfaça.

# O que é Constraint Programming

CP é como montar um quebra-cabeça: cada peça (uma decisão, como escolher um horário) tem que se encaixar com as outras, respeitando as regras do jogo. É usado em áreas como planejamento, logística, análise de dados, e até em jogos como Sudoku! Diferente dos algoritmos genéticos ou PSO, que exploram soluções de forma aproximada, CP busca soluções exatas que atendam a todas as restrições, ou prova que nenhuma solução existe.

# Como funciona o Constraint Programming

CP organiza o problema em três partes principais:

**Variáveis**: São as decisões que precisamos tomar. Por exemplo, em nosso festival, cada show é uma variável, e a decisão é “em qual horário ele acontece”. Em análise de dados, variáveis podem ser coisas como “qual máquina processa um dado” ou “qual recurso é alocado para uma tarefa”.

# Como funciona o Constraint Programming

CP organiza o problema em três partes principais:

**Domínios**: Cada variável tem um conjunto de valores possíveis, chamado domínio. Para o show, o domínio pode ser os horários disponíveis (ex.: 18h, 19h, 20h). Em um problema de dados, o domínio pode ser um intervalo de números ou uma lista de opções.

# Como funciona o Constraint Programming

CP organiza o problema em três partes principais:

**Restrições:** São as regras que limitam as combinações de valores. Por exemplo, “dois shows não podem acontecer no mesmo palco ao mesmo tempo” ou “um show específico só pode ser às 20h”. Em dados, restrições podem ser “uma máquina só processa uma tarefa por vez” ou “o custo total não pode exceder um orçamento”.

# Por que usar Constraint Programming

CP é poderoso porque:

**Garante soluções exatas**: Se existe uma solução que respeita todas as regras, o CP a encontra. Se não existe, ele avisa.

**É flexível**: Funciona para problemas variados, como agendamento, alocação de recursos, ou até planejamento de experimentos em ciência de dados.

**Lida com regras complexas**: Você pode expressar restrições como “se A acontece, então B não pode” ou “o total deve ser menor que X”.

# Por que usar Constraint Programming

Comparado com **PSO** (**Particle Swarm Optimization**) ou algoritmos genéticos, CP é menos sobre “adivinhar” e mais sobre “raciocinar” com base nas regras. Isso o torna ideal para problemas onde as restrições são claras, como organizar tarefas ou recursos em análise de dados.

# O que veremos hoje?

Vamos aplicar CP a dois problemas reais de análise de dados, usando datasets públicos:

**Exemplo 1**: Usaremos o dataset Iris para agendar tarefas de análise de dados. Cada tarefa processa um subconjunto de flores, e queremos decidir quando cada tarefa acontece, respeitando restrições como “uma máquina só faz uma tarefa por vez” e “algumas tarefas precisam de horários específicos”.

# O que veremos hoje?

Vamos aplicar CP a dois problemas reais de análise de dados, usando datasets públicos:

**Exemplo 2**: Usaremos o dataset Wine Quality para alocar recursos (máquinas de teste) para avaliar amostras de vinho. Queremos minimizar o custo total, respeitando restrições como capacidade das máquinas e prazos.

# O que veremos hoje?

Nos códigos, usaremos a biblioteca OR-Tools no Google Colab, que nos ajuda a definir variáveis, restrições e encontrar soluções. Vamos também criar gráficos para entender os resultados, como cronogramas ou alocações, conectando CP à prática da análise de dados!

# A CP encontra ótimo local ou global?

Na CP, quando resolvemos um **problema de satisfação de restrições** (como agendar tarefas no Iris), o objetivo é encontrar qualquer solução que respeite todas as regras — como um cronograma válido para o festival. Nesse caso, não há “ótimo” a menos que definamos um. Já em um **problema de otimização** (como minimizar custos no Wine Quality), a CP é projetada para encontrar o **ótimo global** — a melhor solução possível que obedeça às restrições.

# A CP encontra ótimo local ou global?

Pense no ótimo global como a montanha mais alta em uma paisagem. Diferente do PSO, que explora colinas e pode parar em uma menor (um ótimo local), o solucionador da CP examina combinações de forma sistemática, eliminando opções inválidas, para garantir que encontra a montanha mais alta. No exemplo do Wine Quality, a CP testa alocações de máquinas para encontrar a mais barata, garantindo o menor custo possível.

# A CP encontra ótimo local ou global?

Mas isso depende de:

**Restrições corretas**: Se esquecermos uma regra (ex.: a capacidade de uma máquina), a solução pode não ser a melhor.

**Tamanho do problema**: Em problemas muito grandes, o solucionador pode precisar de limites de tempo, mas nos nossos exemplos (agendamento e alocação pequenos), a CP acha o ótimo global sem problemas.

# E se existirem vários ótimos locais? A CP fica confusa?

Em problemas de otimização, um **ótimo local** é uma solução melhor que as vizinhas, mas não a melhor de todas — como uma colina mais alta que as próximas, mas menor que a montanha principal. Na CP, ótimos locais não são um problema como no PSO, porque a CP não “escala colinas” ou explora aos poucos. Ela usa uma busca sistemática, avaliando todas as soluções possíveis que respeitam as restrições, garantindo o ótimo global.

# E se existirem vários ótimos locais? A CP fica confusa?

Se houver **múltiplos ótimos globais** — por exemplo, dois cronogramas diferentes para as tarefas do Iris que são igualmente válidos, ou duas alocações no Wine Quality com o mesmo custo mínimo — a CP encontra um deles. Qual ela escolhe depende da lógica interna do solucionador, mas todos são igualmente bons. No Wine Quality, se alocar três amostras na Máquina 0 e duas na Máquina 1 custar o mesmo que outra alocação válida, a CP pode retornar qualquer uma, e ambas serão globalmente ótimas.