

Programação por Restrições

Um breve sobrevôo

Claudio Cesar de Sá

Independent Researcher and WhatsTV Inc.

Programação por Restrições

- ▶ O que é a PR?
- ▶ Seus princípios
- ▶ Onde usar?
R: No mundo dos NP-completos (complexidade exponencial), preferencialmente, domínios discretos \Rightarrow Otimização Combinatória
- ▶ Contexto da IA e a PO (Pesquisa Operacional)
- ▶ Contexto do Brasil \times cenário mundial
onde está esta *turma*? onde se usa *isto*?
- ▶ As ferramentas e linguagens
- ▶ *Mãos a obra*: códigos
- ▶ Conclusões

Rápida Apresentação Minha

- ▶ Prof universitário aposentado – UDESC
- ▶ Trabalhei em várias IES: entre elas a UFAL
- ▶ Atualmente: avô, estagiário em uma *start-up*, jardinagem e maratonista de águas-abertas
- ▶ Interesses: CP, Picat, Linux, V, ARMs, etc

Rápida Apresentação Minha

- ▶ Prof universitário aposentado – UDESC
- ▶ Trabalhei em várias IES: entre elas a UFAL
- ▶ Atualmente: avô, estagiário em uma *start-up*, jardinagem e maratonista de águas-abertas
- ▶ Interesses: CP, Picat, Linux, V, ARMs, etc
- ▶ Ou seja, estou no grupo dos inquietos!

Programação por Restrições (PR)

- ▶ A **Programação por Restrições** (PR) é conhecida por *Constraint Programming* ou simplesmente **CP**
- ▶ *Restrição* em inglês é *restriction*, só que tem a conotação é área, espaço, delimitação, etc, de algo que não é permitido ou seja restrito. O que não se relaciona com *constraint* do inglês.
- ▶ Mas *constraint* não tem um boa tradução. Algo como: *filtragem, encolhimento* para **este** contexto é quase lá.

Programação por Restrições (PR)

- ▶ A **Programação por Restrições** (PR) é conhecida por *Constraint Programming* ou simplesmente **CP**
- ▶ *Restrição* em inglês é *restriction*, só que tem a conotação é área, espaço, delimitação, etc, de algo que não é permitido ou seja restrito. O que não se relaciona com *constraint* do inglês.
- ▶ Mas *constraint* não tem um boa tradução. Algo como: *filtragem, encolhimento* para **este** contexto é quase lá.
- ▶ Uma poderosa teoria (e técnica) que contorna a complexidade de certos problemas exponenciais

Em termos práticos (e rápido):

- ▶ Sim, há muitos casos de uso no Brasil: **ALL** (usa o CPLEX para escalonamento e horário dos trens no sul do Brasil), **TOTVS** (comprou a DATASUL, módulo de planejamento de produção: SICSTUS - PROLOG), mais rescentemente, a **Mercado Livre**, usam a OR-TOOLS para roteamento, logística de depósitos e empacotamento (*bin packing* nas vans. Tem uma equipe considerável só neste segmento!
- ▶ IBM oferece consultoria e soluções com uma equipe enorme nesta área. Há um custo clássico da IBM.
- ▶ O pacote *free* de roteamento/escalonamento de veículos da Google, tem feito muitas empresas investirem, nesta direção para agregar valor aos seus produtos.

Um Algoritmo da Programação por Restrições (PR)

- ▶ *Aproximadamente* o algoritmo da **PR** é dado:
 1. Avaliar algebricamente os domínios das variáveis com suas restrições
 2. Intercala iterativamente a **propagação de restrições** com um **algoritmo de busca**
 3. A cada variável instanciada, o processo é repetido sobre as demais variáveis, reduzindo progressivamente o espaço de busca
 4. Volte ao passo inicial até que os domínios permaneçam estáticos e que as variáveis apresentem instâncias consistentes

Um Algoritmo da Programação por Restrições (PR)

- ▶ *Aproximadamente* o algoritmo da **PR** é dado:
 1. Avaliar algebricamente os domínios das variáveis com suas restrições
 2. Intercala iterativamente a **propagação de restrições** com um **algoritmo de busca**
 3. A cada variável instanciada, o processo é repetido sobre as demais variáveis, reduzindo progressivamente o espaço de busca
 4. Volte ao passo inicial até que os domínios permaneçam estáticos e que as variáveis apresentem instâncias consistentes
- ▶ Uma das virtudes da **PR**: a legibilidade e clareza em construir **modelos**
- ▶ Exemplo: $10x = y$ tal que $D_x = \{10..20\}$ e $D_y = \{150..170\}$, logo, a solução aqui é um cartesiano:
 $\{(15, 150), \dots, (17, 170)\}$

Fluxo de Cálculo da PR

figures/dinamica_pr.pdf

Onde o objetivo da PR é:

figures/reducao_PR_01.pdf

Figura: Realizar buscas com regiões reduzidas – promissoras (regiões factíveis de soluções)

Redução Iterativa em Sub-problemas

figures/reducao_PR_02.pdf

Figura: Redução de um CP em outros sub-problemas CPs equivalentes – uma bisecção

Conceitos

A PR tem os seguintes elementos:

- ▶ Um conjunto de **variáveis**: $X_1, X_2, X_3, \dots, X_n$
- ▶ Um conjunto de **domínios** dessas variáveis: $D_{X_1}, D_{X_2}, D_{X_3}, \dots, D_{X_n}$
- ▶ Finalmente, as **restrições**, que são relações n-árias entre estas variáveis
- ▶ Exemplo completo: $D_{X_1} = D_{X_2} = \{3, 4\}$ e $X_1 \neq X_2$

Aliado a tudo isto, a CP tem um jargão todo particular: I

- ▶ Problemas \Rightarrow Modelos matemáticos (quase sempre, prontos para codar)
- ▶ Ferramentas de CP: linguagens \times *solvers* \times bibliotecas e extensões de linguagens
- ▶ Sintaxe: algumas simples outras mas todas entendem $7\# = x$ é reflexivo
- ▶ Tipagem?
- ▶ Domínios: inteiros? reais?
- ▶ Tipos de restrições: globais \times a objetos
- ▶ Nível das restrições: simples ($a\# > b$) as complexas (*circuit(all_vertex)*), dfa – exp regulares, soma....
- ▶ Otimização: *branch-bound*
- ▶ Bisseção: ramos/trilhas por onde segue a busca
- ▶ Suporte a clones? (*threads*)

Aliado a tudo isto, a CP tem um jargão todo particular: II

- ▶ Reificação: condição lógica que valida uma disjunção de restrições

- ▶ *Channeling* : $y = f(x) \leftrightarrow x = g(y)$

Exemplificando estes 2 tópicos em OR-TOOLS (Python):

```
# Criar duas restrições reifadas tal que : c1 <--> c2
# c1: b implica (y == 10 - x).
model.Add(y == 10 - x) . OnlyEnforceIf (b)
# c2: not(b) implica y == 0.
model.Add(y == 0) . OnlyEnforceIf (b.Not())
```

- ▶ Na CP 'raiz', não há "if-then-else", então a *reifagem* (*reyfing*) contorna esta dificuldade
- ▶ Finalmente:

Aliado a tudo isto, a CP tem um jargão todo particular: III

- ▶ **Escolha da sequência** da variáveis do modelo a serem exploradas
- ▶ **Escolha** por onde se inicia a exploração dos **domínios** destas variáveis
- ▶ Provedores SAT, provedores lógicos baseado em LPO etc
- ▶ No meio de tudo isto, linguagens declarativas *caíram feito luva* \Rightarrow Prolog e seus dialetos
- ▶ Resumo: CP é **postar** restrições (**declarar**) afim de *encolher* o domínio problema
- ▶ A CP quanto há um paradigma programação (semântica):
$$C_1 \wedge C_2 \wedge \dots \wedge C_m \vdash \{\blacksquare_1, \dots, \blacksquare_n\} \vee \emptyset$$

Problemas para CP na indústria

- ▶ Escalonamento e alocação de recursos
- ▶ Distribuição de turnos de trabalhos
- ▶ Problemas de empacotamento (a van do Mercado Livre)
- ▶ Roteamento (veículos, cabos, painéis, placas)
- ▶ Sequenciamento de DNA (várias regras sobre sequências válidas)
- ▶ Planejamento financeiro
- ▶ Enfim: problemas que envolvam uma busca combinatória e otimização

Ou seja, ainda bem que existem os NPs !!!!

A CP no Brasil

Rapidamente:

- ▶ Década de 70: primeiros doutores em CC regressam ao país
- ▶ Década de 80: primeiros programas de pós-graduação em CC
- ▶ Década de 90: primeiros simpósios, congressos e áreas se estabelecendo
- ▶ Ano 2000: CAPES qualifica a pesquisa, programas, etc, indicadores de produtividade

A CP no Brasil

Rapidamente:

- ▶ Década de 70: primeiros doutores em CC regressam ao país
- ▶ Década de 80: primeiros programas de pós-graduação em CC
- ▶ Década de 90: primeiros simpósios, congressos e áreas se estabelecendo
- ▶ Ano 2000: CAPES qualifica a pesquisa, programas, etc, indicadores de produtividade
- ▶ **Aí correria começa ⇒ publicar !!!!**
- ▶ **Apenas uma parte do modelo americano foi instanciado !**

A CP no Brasil

figures/ia_brasil_90.jpg

Figura: Cenário *aproximado* do borburinho em torno da **IA**

A CP no Brasil

figures/ia_brasil_2020.jpg

Figura: A chegada dos métodos evolutivos ou bio-inspirados

A CP no Brasil

figures/ia_brasil_2020.jpg

Figura: A chegada dos métodos evolutivos ou bio-inspirados

Sim, um TSP alcançou dimensões jamais vistas com um ACO (colônia de formigas)

Contexto Mundial I

- ▶ No cenário mundial, a CP já tinha seus nichos de sucesso na indústria. Muitas aplicações de escalonamento foram possíveis com as linguagens declarativas!
- ▶ Sim, o IJCAI tem a sua força de impacto!

Contexto Mundial II

- ▶ **França**: pessoas espalhadas (isoladas), mas presente em quase todas universidades. Embora tenha sido o berço do Prolog, hoje os admiradores estão em Marselha apenas. Contudo, o ILOG veio de lá e foi comprado pela IBM, virou o CPLEX (muito investimento por parte da IBM), com vários cases de sucesso pelo mundo. Há outros produtos por lá: Choco (Java com CP) e Scala/JaCoP
- ▶ **Alemanha**: vários centros de pesquisa e aplicações consolidadas. Contudo, o ambiente com **Answer Set Programming**, da Universidade de Potsdam tem atraído muito a atenção. Grupo POTASCO. Usa várias lógicas para fazer inferência em seus modelos *ground*. Essencialmente, visa descobrir o Universo de Herbrand, por isto é conjunto de respostas!

Contexto Mundial III

- ▶ **Itália**: tem investido em ferramentas para o ASP. Há alguns expoentes por lá.
- ▶ **UK**: vários centros. Destaque: Imperial College com o **ECLiPSe-CLP**, contudo, Cork na Irlanda tem um grupo forte de CP. Até Oxford tem investido em ASP+Prolog+Python
- ▶ **Suécia**: uma empresa produz o SICtus Prolog, um Prolog compilado com mais de 20 anos de estrada e aplicações. Ainda ativa! *Gecode*: uma biblioteca em C++, muito veloz, junto com o pessoal da Alemanha mantém este resolvidor ativo, e **muito bem documentado**. Na Suécia, temos ainda: Hakan Kjellerstrand – <http://hakank.org>

Contexto Mundial IV

- ▶ **Noruegua**: há uma empresa chamada PDC (Prolog compilado e tipado), com uma história de uso e sucesso. Apenas sob Windows. A PDC existe com muitos cases pela Europa.
- ▶ **Holanda**: **Swi-Prolog**, do departamento de psicologia da universidade Amsterdam. Ainda ativo e mantido pelo Jan e outros. Interpretado e academicamente é o mais usado no mundo. A propósito, o Prolog é bem usado nas boas e grandes universidades pelo mundo.
- ▶ **Polônia**: há um livro escrito free sobre CLP – A Gentle Guide to Constraint Logic Programming via ECLiPSe por Antoni Niederlinski - Varsóvia
http://www.anclp.pl/download/AN_CLP.pdf

Contexto Mundial V

- ▶ **Tchéquia**: em Praga há o Roman Barták (planejamento, Prolog, Picat, PDDL, etc), esteve no NE algumas vezes
- ▶ **USA**: tivemos nas décadas de 80/90 Arity Prolog, Amzi Prolog, Stramberry Prolog, empresas fechadas. Restou a Google, com o OR-TOOLS (núcleo em C++) e interfaces de programação em: Java, C++, Python. Devido o investimento nos algoritmos de roteamento de veículos autônomos, hoje é a ferramenta com maior número de usuários. Seu suporte é espalhado, mas, gerenciado por Paris (resquícios do pessoal do ILOG). Há ainda o Picat de Neng-Fa e outros, como um dos principais avanços do Prolog em 40 anos. O livro do Picat está disponível ...

Contexto Mundial VI

- ▶ **Austrália**: NICTA patrocina a turma CP e OR há vários anos. Ações conjuntas e um plano nacional, tornou o país de referência na área de CP. Destaque para: Peter Stuckley, Pascal Van Hentenryck, Guido, ... há um solver para OR+CP chamado FlatZinc, e sua linguagem de interface é o Minizinc. Hoje, os *aussies* (embora a maioria deles tenham vindo da Europa) ditam as várias ramificações da área de Otimização Combinatória (sim, a turma da PO está junto)

Em resumo: lá fora a área é ativa com várias inserções na indústria, aqui corremos atrás das publicações com qualis!

Eventos

- ▶ CP 2022 : Principles and Practice of Constraint Programming – Haifa – Israel (capítulos de livros)
- ▶ CPAIOR 2022 : International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research – Los Angeles – USA

Até 20xx nenhum brasileiro havia publicado nestes 2 eventos!
Sem qualis ...

Antes das *mãos-na-massa*:

Antes dos códigos, lembrando que a CP realiza buscas, encolhendo domínios, realizando saltos a frente, ou quando voltar, sabe por onde já passou.

Ou seja, **a CP é uma técnica que preza por uma busca completa!**

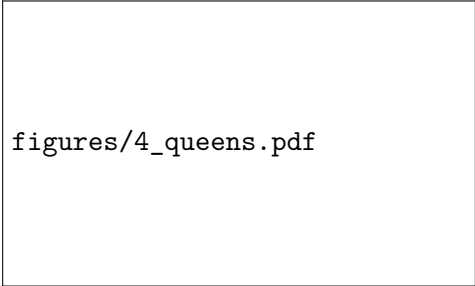
Exemplo clássico 01: Coloração de Mapas

Seja um mapa com 5 países. Cada país deve ter uma cor, mas os países vizinhos não podem ter a mesma cor.

`figures/coloracao_01.jpg`

- ▶ **Variáveis** são os países: X1 a X5 – **nós** em uma representação em grafo
- ▶ Quanto ao **domínio**? $\text{Cores} = \{1, 2, 3, 4\}$ – **rotulação** destes nós
- ▶ **Restrições**: as **arestas** de um grafo representam a relação entre países-vizinhos

Exemplo clássico 02: N-Rainhas no Tabuleiro $N \times N$ I



figures/4_queens.pdf

Figura: Distribuir 4 rainhas no tabuleiro sem se atacarem mutuamente

Exemplo clássico 02: N-Rainhas no Tabuleiro $N \times N$ II

figures/4_rainhas_01.jpg

figures/4_rainhas_02.jpg

- ▶ **Este problema tem muitas aplicações práticas! – uma metáfora**
- ▶ Muitas abordagens quanto ao número de variáveis e domínios
- ▶ Estudo sobre **simetrias** (e *espelhamentos*) de na modelagem e resultados (algo comum na PR)

Exemplos – Mão na massa

Apresentando a essência dos elementos da CP na prática:

1. Um Cripto-Aritmético: um clássico escrito em Minizinc (boa para modelar) e OR-TOOLS (Python)
2. Um de escala de consultórios médicos (uso de matriz) \Rightarrow em Picat (um Prolog *vitaminado*)

Basicamente, 2 problemas distintos!

Exemplo – 01 – Um Cripto-Aritmético

$$\begin{array}{rcccccc} V & I & O & L & I & N \\ - & C & E & L & L & O \\ \hline C & O & R & N & E & T \end{array}$$

Preencher as letras acima, com números de $\{0..9\}$, sem repetições e que satisfaçam a subtração acima.

Exemplo – 01 – Um Cripto-Aritmético

$$\begin{array}{rcccccc} V & I & O & L & I & N \\ - & C & E & L & L & O \\ \hline C & O & R & N & E & T \end{array}$$

Preencher as letras acima, com números de $\{0..9\}$, sem repetições e que satisfaçam a subtração acima.

Poderíamos transformar o problema:

$$\begin{array}{rcccccc} C & O & R & N & E & T \\ + & C & E & L & L & O \\ \hline V & I & O & L & I & N \end{array}$$

Mas, vamos manter a formulação original!

- ▶ Reversibilidade (exemplo: `element(Value, L, Index)`)
- ▶ Simetria (exemplo: `n-rainhas`)
- ▶ Dualidade (*dual view*, problema da cerca – `set covering`)

Considerações sobre este clássico:

- ▶ Embora sejam *toy problems*, tem um viés de aplicações reais muito forte
- ▶ Efetivamente ilustram a força da CP. Como aperitivo: implemente o problema acima em uma linguagem procedural (9 laços de repetição aninhados)
- ▶ As árvores de busca ilustradas nas figuras anteriores, estão aqui presentes
- ▶ Características da CP: **escolha de variável** e **varredura no domínio** tem seus impactos bem visíveis nestes exemplos (não apresentarei estes itens)

Implementação em Minizinc

Acompanhar:

https://raw.githubusercontent.com/claudiosa/CCS/master/minizinc/violin_cripto.mzn

Compiling violin_cripto.mzn

Running violin_cripto.mzn

V: 7 I: 0 O: 4 L: 8 I: 0 N: 9

C: 6 E: 2 L: 8 L: 8 O: 4

C: 6 O: 4 R: 1 N: 9 E: 2 T: 5

f MAXIMIMIZACAO: 21

C1: 0 C2: 1 C3: 1 C4: 0 C5: 1

=====

Implementação em OR-TOOLS (Python)

Acompanhar:

https://github.com/claudiosa/CCS/blob/master/python/or-tools/violin_cripto.py

```
$ python3 violin_cripto.py
===== RESULTS =====
MAX = 21
V:7 || I:0 || O:4 || L:8 || N:9 || C:6 || E:2 || R:1 || T:5
carry[0] : 0
carry[1] : 1
carry[2] : 1
carry[3] : 0
carry[4] : 1
** Final Statistics **
- conflicts : 16
- branches  : 29
- wall time : 0.023359 s
```

Exemplo – 02 – Escala de Consultórios

	2a.	3a.	4a.	5a.	6a.
1a. Sala	[1..7]	[1..7]	[1..7]	[1..7]	[1..7]
2a. Sala	[1..7]
3a. Sala	[1..7]
4a. Sala	[1..7]

- ▶ Basicamente, alocar 7 especialidades médicas nestas 4 salas, nestes 5 dias da semana – *assignment problem*
- ▶ 1:oftalmo, 2:otorrino, 3:pediatra, 4:gineco, 5:cardio, 6:dermato, 7:clin_geral

Exemplo – 02 – Escala de Consultórios

- ▶ Seja um Posto Atendimento Médico, um PA, com 4 consultórios e 7 especialidades médicas
- ▶ O problema é distribuir estes médicos nestes 4 consultórios tal que alguns requisitos sejam atendidos (restrições satisfeitas)
- ▶ A abordagem aqui é ingênua e sem muitos critérios (problema de minha autoria)

Modelagem do Problema

- ▶ Vamos usar uma matriz bi-dimensional para representar o problema. Linhas \leftrightarrow consultórios (1 a 4), e as colunas \leftrightarrow dias da semana (1 a 5)
- ▶ Esta matriz será preenchida com valores/códigos de 1 a 7, de acordo com a especialidade médica.
- ▶ Assim o domínio da matriz Quadro (4×5) será preenchida com um destes códigos.
- ▶ Vamos utilizar restrições globais: `member` e `all_different`
- ▶ As restrições globais se aplicam sobre um conjunto de variáveis.

Matriz de Atribuição

	2a.	3a.	4a.	5a.	6a.
1a. Sala	[1..7]	[1..7]	[1..7]	[1..7]	[1..7]
2a. Sala	[1..7]
3a. Sala	[1..7]
4a. Sala	[1..7]

O domínio de valores: 1..7 (7 especialidades médicas)

Modelagem – Comentários

- ▶ A fase de busca e propagação do comando `solve(Critérios, Variáveis)`, há dezenas de combinações possíveis: consultar o guia do usuário
- ▶ Tem-se os predicados extras ... são muitos, todos os da CP

Código Completo

- ▶ Acompanhar as explicações do código de:
`https://github.com/claudiosa/CCS/blob/master/picat/horario_medico_CP.pi`
- ▶ Confira a execução e testes

Código em Partes

```
modelo =>
    Dias = 5, % segunda= 1, ....., sexta-feira = 5
    Consultorio = 4,
    L_dom = [ oftalmo, otorrino, pediatria,  gineco,
%           1         2         3         4
            cardio, dermato, clin_geral ],
%           5         6         7
    Quadro = new_array(Consultorio, Dias ), %% Lin x Col
    Quadro :: 1 .. L_dom.len , %% operador len . "eh colado"
    ...
```

As variáveis do problema são em letras MAIÚSCULAS, idem a Prolog

Código em Partes

```
%% O medico 2 NUNCA trabalha no consultorio 1
foreach ( J in 1 .. Dias )
    Quadro[1,J] #!= 2
end,
```

```
%% O medico 5 NUNCA trabalha no consultorio 4
foreach ( J in 1 .. Dias )
    Quadro[4,J] #!= 5
end,
```

...

Código em Partes

```
%% O Clin Geral deve vir o maior numero de dias ...
%% Esta restricao eh matematicamente é HARD
    foreach ( I in 1 .. Consultorio )
        member(7,[Quadro[I,J] : J in 1..Dias])
    end,

%% Ninguém trabalha no mesmo consultorio em dias seguidos
foreach ( J in 1 .. Dias )
    all_different( [Quadro[I,J] : I in 1..Consultorio] )
end,

%% Ninguém trabalha no mesmo dia em mais de um consultorio
foreach ( I in 1 .. Consultorio )
    all_different( [Quadro[I,J] : J in 1..Dias] )
end,
...
```


Código em Partes

```
% A BUSCA
solve([ff], Quadro),
    % UMA SAIDA

    printf("\n Uma escolha:"),
    print_matrix( Quadro ),
    print_matrix_NAMES( Quadro , L_dom ),
    printf(".....\n") .
```

Código em Partes

```
print_matrix_NAMES( M, Lista ) =>
  L = M.length,
  C = M[1].length,
  nl,
  foreach(I in 1 .. L)
    foreach(J in 1 .. C)
      printf(":%w \t" , print_n_lista( M[I,J], Lista) )
    % printf("(%d,%d): %w " , I, J, M[I,J] ) -- FINE
  end,
  nl
end.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
print_n_lista( _, [] ) = [].
print_n_lista( 1, [A|_] ) = A.
print_n_lista( N, [_|B] ) = print_n_lista( (N-1), B ) .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Saída - I

```
Picat> cl('horario_medico_CP.pi').  
Compiling:: horario_medico_CP.pi  
horario_medico_CP.pi compiled in 10 milliseconds  
loading...
```

yes

```
Picat> main
```

Uma escolha:

7 1 3 4 5

4 7 2 3 1

1 3 7 5 2

3 2 1 7 4

Saída - II

```
:clin_geral :oftalmo :pediatria :gineco :cardio
:gineco :clin_geral :otorrino :pediatria :oftalmo
:oftalmo :pediatria :clin_geral :cardio :otorrino
:pediatria :otorrino :oftalmo :clin_geral :gineco
.....
yes
```

Saída - III

```
$ time(picat horario_medico_CP.pi )
```

```
Uma escolha:
```

```
7 1 3 4 5
```

```
4 7 2 3 1
```

```
1 3 7 5 2
```

```
3 2 1 7 4
```

```
:clin_geral :oftalmo :pediatra :gineco :cardio  
:gineco :clin_geral :otorrino :pediatra :oftalmo  
:oftalmo :pediatra :clin_geral :cardio :otorrino  
:pediatra :otorrino :oftalmo :clin_geral :gineco
```

```
.....
```

```
real 0m0,023s
```

```
user 0m0,007s
```

```
sys 0m0,013s
```

```
[ccs@gerzat picat]$
```

Resumindo a PR

- ▶ Há outros métodos para se resolver problemas.
Exemplo: Programação Linear, Buscas Heurísticas, AGs, Busca Gulosa, ACOs etc

Resumindo a PR

- ▶ Há outros métodos para se resolver problemas.
Exemplo: Programação Linear, Buscas Heurísticas, AGs, Busca Gulosa, ACOs etc
- ▶ As **restrições globais** se aplicam sobre um conjunto de variáveis e há muitas disponíveis nos sistemas de CP, que promovem um atalho nas soluções

Resumindo a PR

- ▶ Há outros métodos para se resolver problemas.
Exemplo: Programação Linear, Buscas Heurísticas, AGs, Busca Gulosa, ACOs etc
- ▶ As **restrições globais** se aplicam sobre um conjunto de variáveis e há muitas disponíveis nos sistemas de CP, que promovem um atalho nas soluções
- ▶ A área é extensa e caminha para um hibridismo: paradigmas e computação evolutiva (os reais)

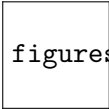
Resumindo a PR

- ▶ Há outros métodos para se resolver problemas.
Exemplo: Programação Linear, Buscas Heurísticas, AGs, Busca Gulosa, ACOs etc
- ▶ As **restrições globais** se aplicam sobre um conjunto de variáveis e há muitas disponíveis nos sistemas de CP, que promovem um atalho nas soluções
- ▶ A área é extensa e caminha para um hibridismo: paradigmas e computação evolutiva (os reais)
- ▶ Por tratar de técnica baseada em **busca completa**, pode ser uma técnica essencial em algumas áreas: finanças, regras de negócio, etc.

Resumindo a PR

- ▶ Há outros métodos para se resolver problemas.
Exemplo: Programação Linear, Buscas Heurísticas, AGs, Busca Gulosa, ACOs etc
- ▶ As **restrições globais** se aplicam sobre um conjunto de variáveis e há muitas disponíveis nos sistemas de CP, que promovem um atalho nas soluções
- ▶ A área é extensa e caminha para um hibridismo: paradigmas e computação evolutiva (os reais)
- ▶ Por tratar de técnica baseada em **busca completa**, pode ser uma técnica essencial em algumas áreas: finanças, regras de negócio, etc.
- ▶ Resumo da PR: segue por uma notação/manipulação algébrica restrita, simplificar e bissecionar as restrições, instanciar variáveis, verificar inconsistências, avançar sobre as demais variáveis, até que todas estejam instanciadas.

Perguntas e Agradecimentos



`figures/thank-you-cloud.jpg`

- ▶ `https://github.com/claudiosa`
- ▶ Email: `ccs1664@gmail.com`
- ▶ Email: `claudio@colmeia.udesc.br`