Ferramenta para Modelagem Molecular em Ambiente Distribuído

Fernando dos Santos[1]

[1]Centro de Ciências Exatas e Naturais
Departamento de Sistemas e Computação
Universidade Regional de Blumenau

Orientador: Prof. Jomi Fred Hübner

2005/1

Roteiro

Introdução

- Avanço da tecnologia de redes
- Informação distribuída
  - Problemas distribuídos
- Necessidade de métodos distribuídos
  - Paralelismo
- Exemplo: Problema de satisfação de restrições.

Problema de Satisfação de Restrições - CSP

- Um CSP é formado por variáveis, domínios e restrições entre variáveis

Definição [Russel and Norvig, 2003]

$< X, D_i, C >$, sendo:

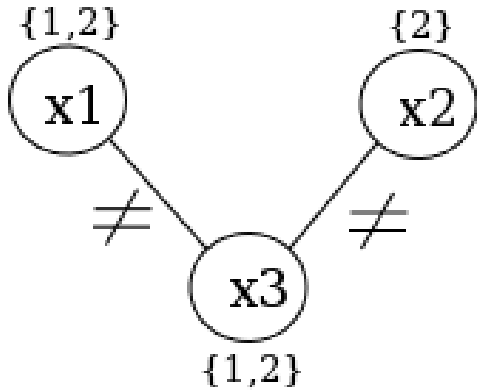
- $X = \{x_1, x_2, \ldots, x_n\}$ variáveis
- $D_i = \{d_1, d_2, \ldots, d_n\}$ domínios (d_i domínio de x_i)
- $C = \{C_1, C_2, \ldots, C_n\}$ restrições

Solução [Russel and Norvig, 2003]

Conjunto de $labels$ ($\{x_i = v_i, x_j = v_j\}$) que satisfaz todas as restrições

Problema de Satisfação de Restrições - CSP

- Um CSP pode ser representado através de um grafo de restrições



- Em ambiente distribuído: DCSP

Problema de Satisfação de Restrições Distribuído - DCSP

- Extensão de CSP $\rightarrow$ mapeamento variáveis-agentes

Definição [Yokoo, 2001]

Conjunto de m agentes

- Cada variável x_i do CSP pertence a um agente i ($pertence(x_j, i)$);
- Restrições distribuídas entre os agentes ($conhece(c_k, l)$).

Solução [Yokoo, 2001]

- $\forall i, \forall j$ onde $pertence(x_j, i)$, o valor de x_i é igual a v_j
- $\forall l, \forall c_k$ onde $conhece(c_k, l)$, c_k é satisfeita considerando a associação $x_j = v_j$.

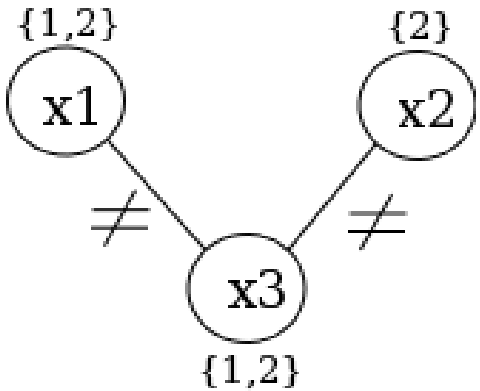
Problema de Satisfação de Restrições Distribuído - DCSP

- Algoritmos que solucionam DCSP [Yokoo, 2001]
  - *Asynchronous Backtracking*
  - *Asynchronous Weak Commitment*
  - *Distributed Breakout*
- Implementação destes algoritmos: *framework* dynDCSP [GRUPO DE INTELIGÊNCIA ARTIFICIAL DA FURB, 2005]
  - Utilizando infraestrutura de comunicação SACI

Utilização de DCSP

- Modelagem Molecular
- Semelhança com CSP/DCSP

Grafo:



Molécula de água:

- Expectativa de melhor desempenho na resolução comparado com métodos centralizados

Modelagem Molecular - Conceitos Simplificados

- Determinação de modelos moleculares
- Modelo molecular: disposição tri-dimensional dos átomos da molécula
- Interações entre átomos produzem energia
- Modelo molecular é determinado quando a energia produzida é **MINIMIZADA**

Problema

Abordagem com CSP/DCSP não satisfatória. Necessidade de **OTIMIZAR** restrições

Problema de Otimização de Restrições - COP

- Extensão de CSP
- Deve-se considerar a **melhor** solução e não **qualquer** solução
- Função de custo, que deve ser minimizada ou maximizada

Definição [Tsang, 1993]

$< X, D_i, C, f >$, onde:

- X, D_i, C é um CSP
- f é a função de otimização

Solução [Tsang, 1993]

Conjunto de $labels$ cujo valor de f é ótimo.

- Em ambiente distribuído: DCOP

Problema de Otimização de Restrições Distribuído - DCOP

- Extensão de COP $\rightarrow$ mapeamento variáveis-agentes

Definição [Modi, 2003]

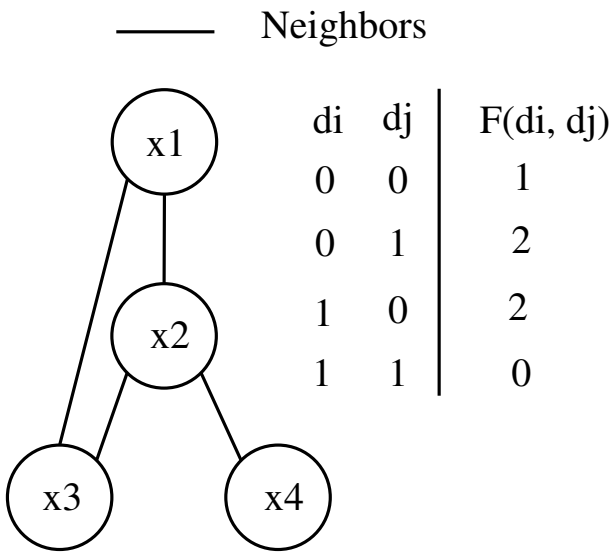
- Conjunto n variáveis $\{x_1, x_2, \ldots, x_n\}$ cada uma associada a um agente e a um domínio discreto e finito $D_1, D_2, \ldots, D_n$
- Função de custo f_{ij} para cada par x_i, x_j

Solução [Modi, 2003]

Conjunto de $labels$ cujo valor de f é ótimo.

Problema de Otimização de Restrições Distribuído - DCOP

- Exemplo de DCOP [Modi, 2003]



- Algoritmo disponível: Adopt

Objetivo do Trabalho

- Implementar e validar o algoritmo Adopt para resolução de DCOPs, para então verificar a viabilidade da especificação do problema de modelagem molecular como COP/DCOP e sua resolução através do algoritmo Adopt

extensões do *framework* dynDCSP

estudo de casos

Algoritmo Adopt - Visão Geral

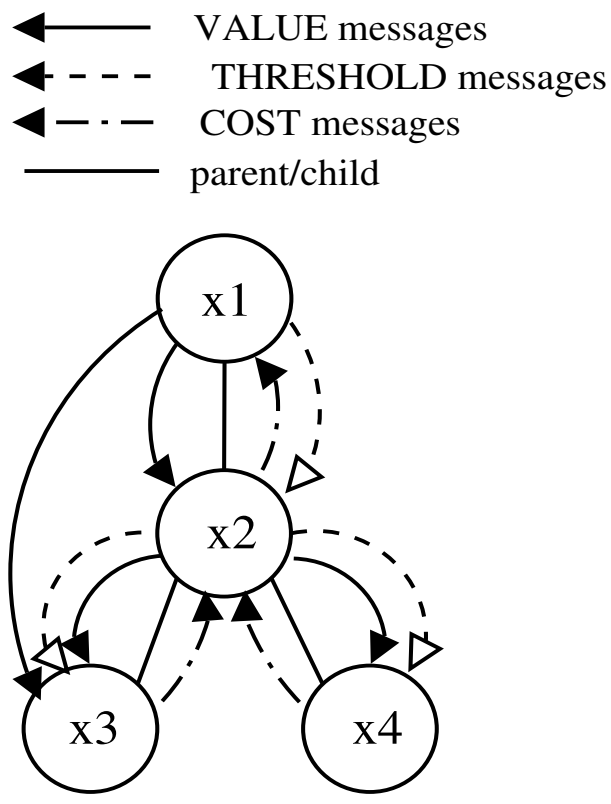
- Ph. D. Thesis - Pragnesh Jay Modi - 2003
  - Assíncrono (primeiro)
  - Comunicação localizada (entre agentes *neighbors*)
    - *neighbor*: agente que possui restrição/relação no grafo
  - Necessidade:
    - Ordenação dos agentes em árvore de busca em profundidade (DFST)

Algoritmo Adopt - Características

- Busca baseada em *lower bound* e *upper bound*
  - estabelecem estimativas de custo
  - computados com informação local, para cada valor do domínio
  - direcionam o processo de busca (convergência)
- Reconstrução de soluções
  - $threshold \rightarrow$ grau de aceitação de solução
  - $threshold$ de x_q armazenado em seu *parent* x_p
  - Ao revisitar uma solução:
    - x_p envia $threshold$ para x_q
- Detecção de término *built-in*
  - Baseada no intervalo *lower/upper bound*
  - Computação finaliza quando intervalo é zero

Algoritmo Adopt - Mensagens

- VALUE $\rightarrow$ informa valor da variável aos *neighbors*
- COST $\rightarrow$ informa *bounds* ao *parent*
- THRESHOLD $\rightarrow$ informa $threshold$ a um *child*
- TERMINATE $\rightarrow$ informa fim da computação a um *child*

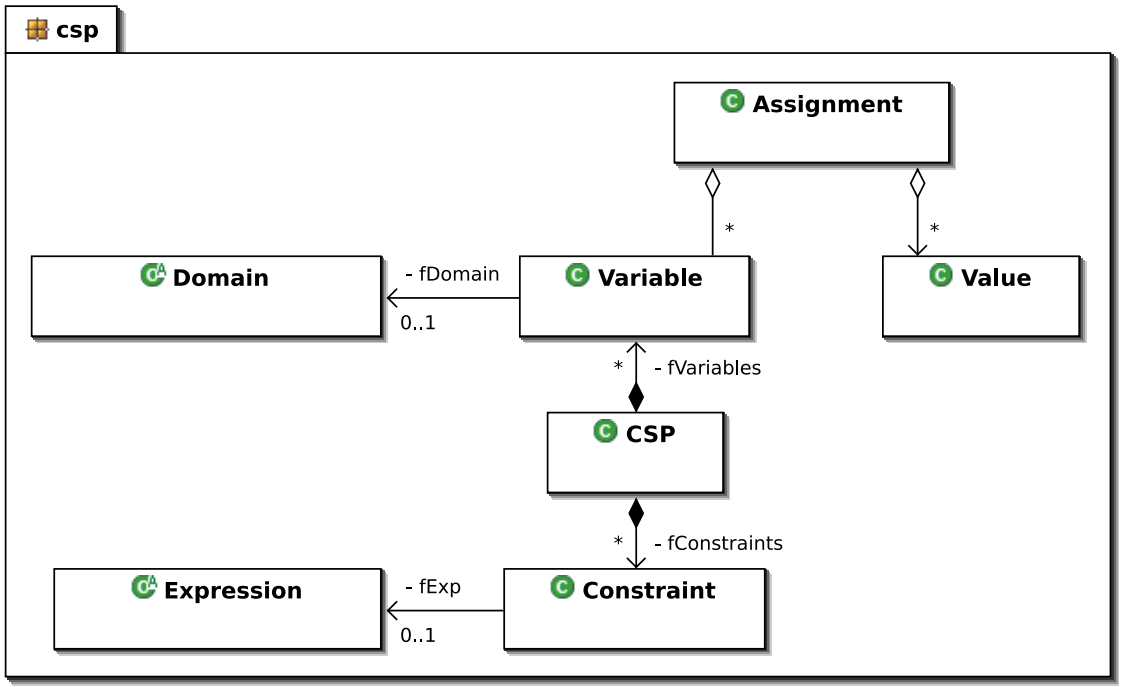


[Modi, 2003]

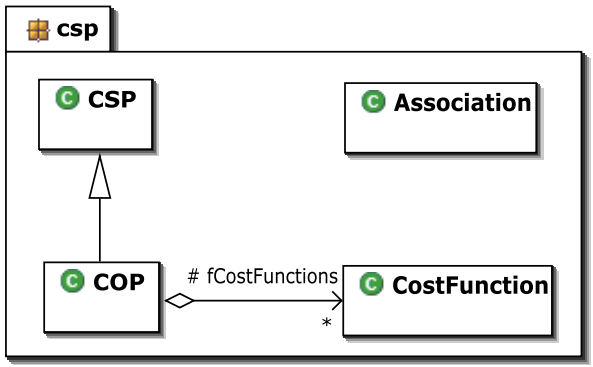
Algoritmo Adopt - Especificação

- Extensões do *framework* dynDCSP
- Pacote `csp`

Original:



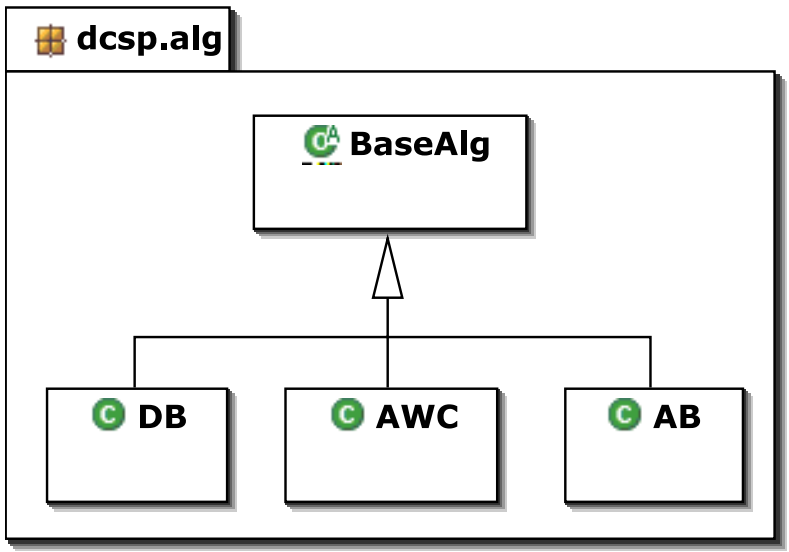
Adições:



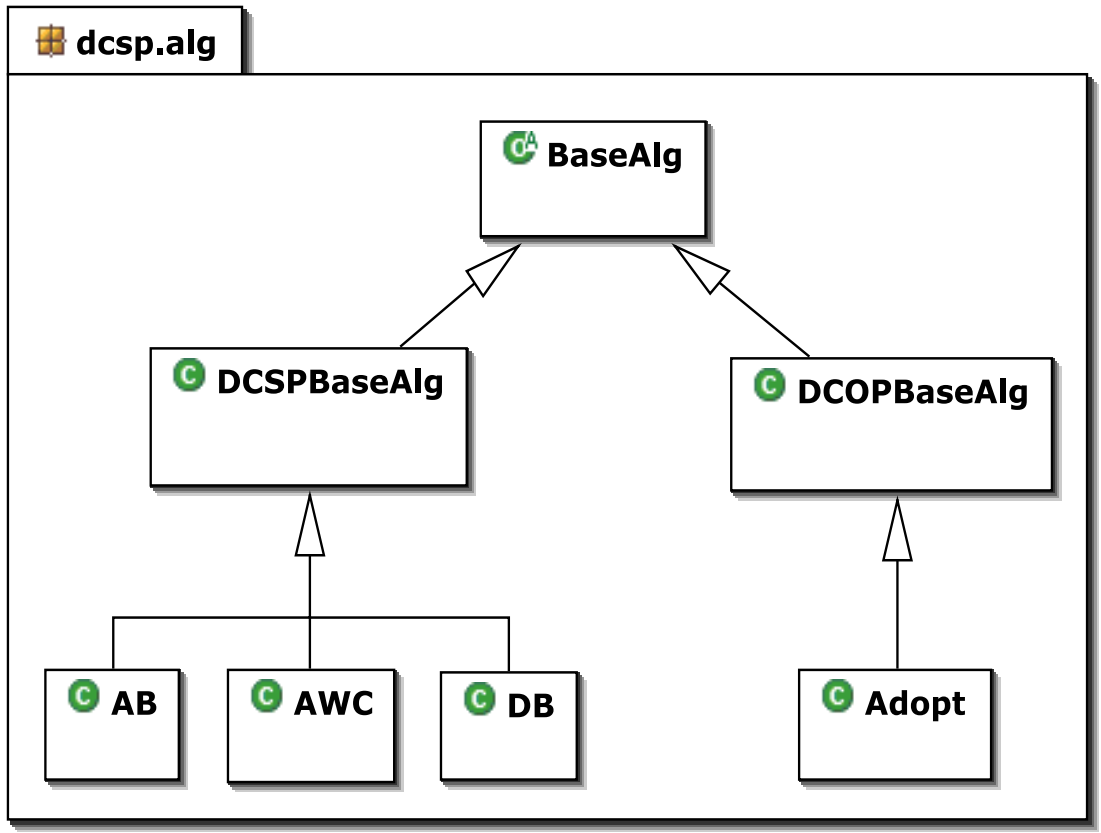
Algoritmo Adopt - Especificação

- Pacote `dcsp.alg`

Original:

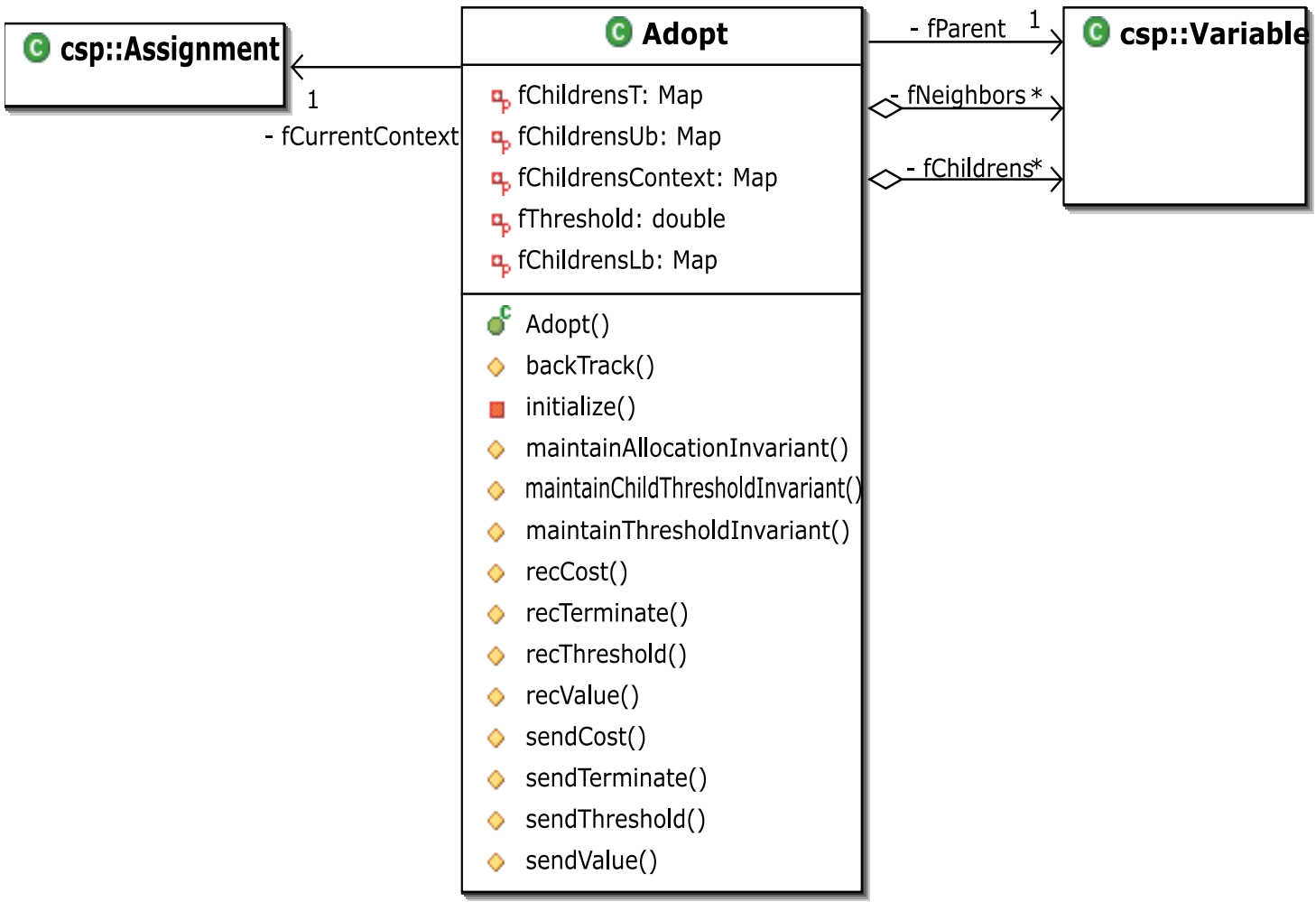


Atual:



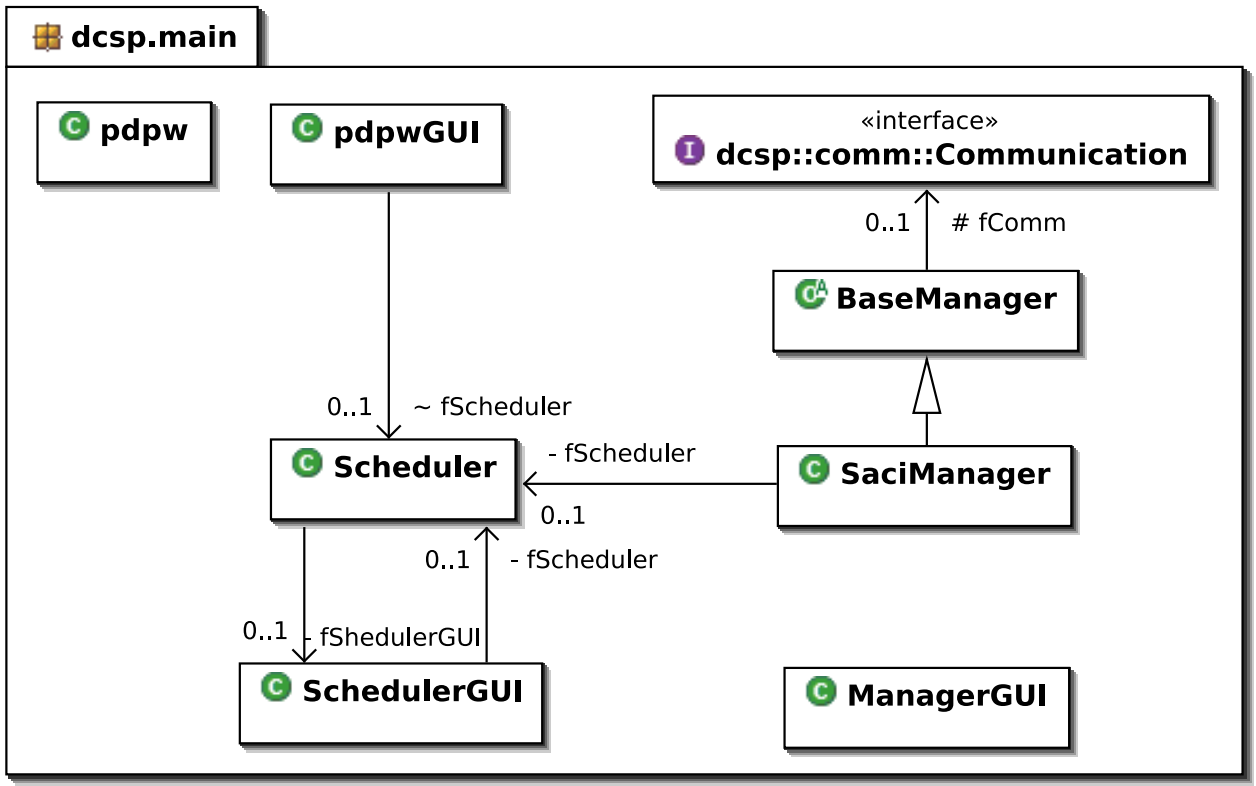
Algoritmo Adopt - Especificação

- Classe `Adopt`



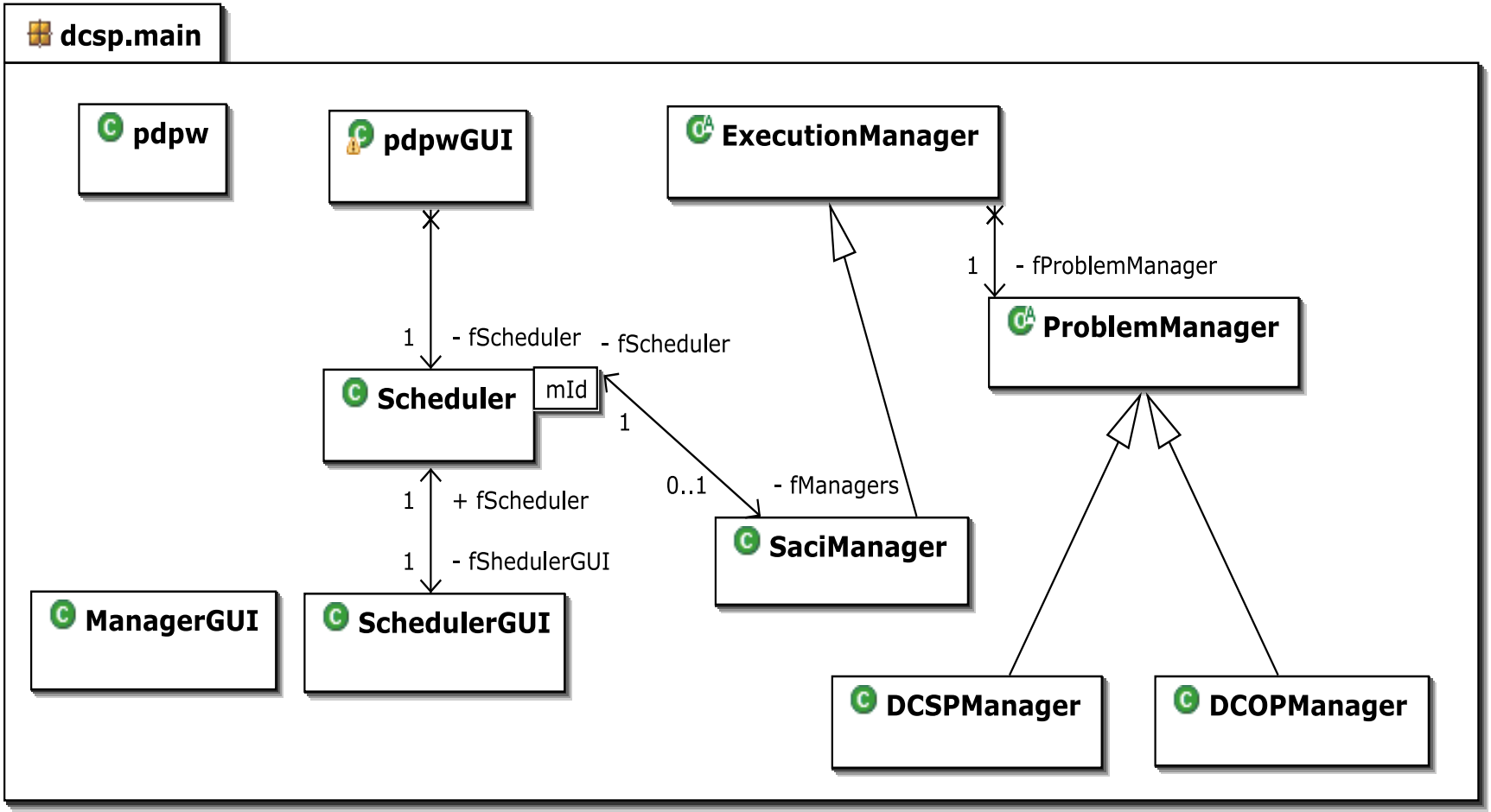
Algoritmo Adopt - Especificação

- Pacote `dcsp.main` (original)



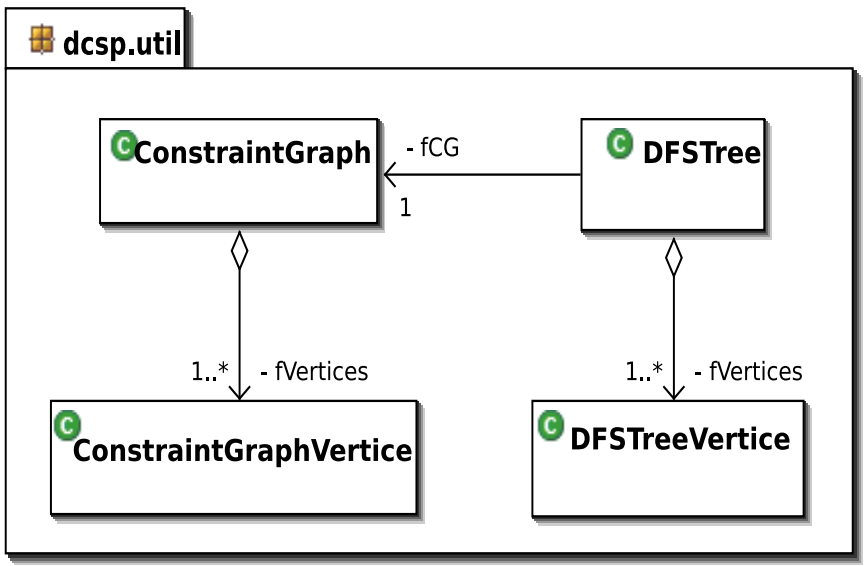
Algoritmo Adopt - Especificação

- Pacote `dcsp.main` (atual)



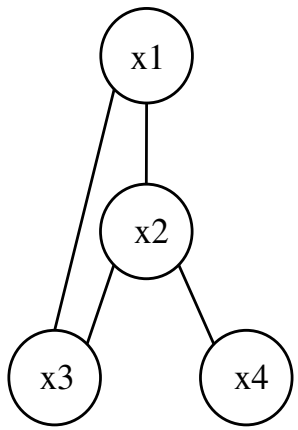
Algoritmo Adopt - Especificação

- Pacote `dcsp.util` (adicionado)

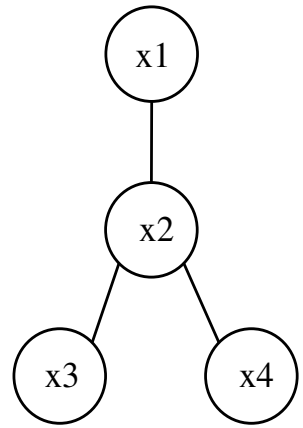


- Geração de DFST
  - Variáveis ordenadas por nome ou grau

grafo:

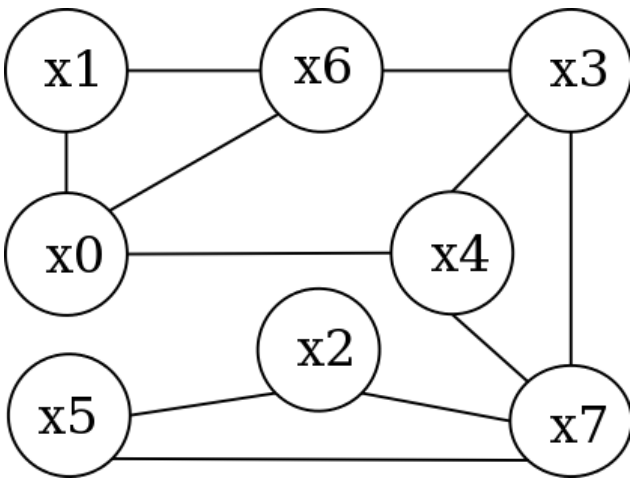


árvore:



Coloração de Grafo

- Atribuir cores aos vértices, minimizando restrições violadas
- Grafo utilizado:
  - 8 vértices $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$
  - 11 arestas (restrições)



Coloração de Grafo - Restrições de Cores

- Restrições binárias
  - Especificadas através de valores que devem ser evitados

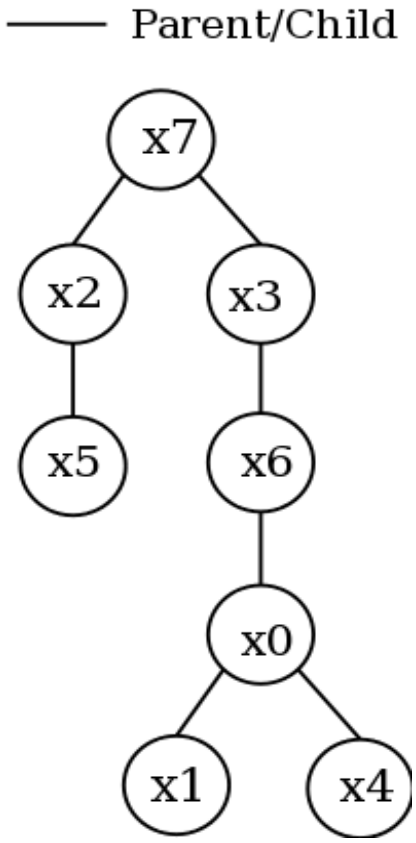
x_0	x_4
2	1
0	1
1	0

Funções de custo

$$f(x_0, x_4) = \begin{cases} 1 & \text{if } (x_0 = 2, x_4 = 1) \text{ or } (x_0 = 0, x_4 = 1) \text{ or} (x_0 = 1, x_4 = 0) \\ 0 & \text{otherwise} \end{cases}$$

Coloração de Grafo - Ordenação em Árvore

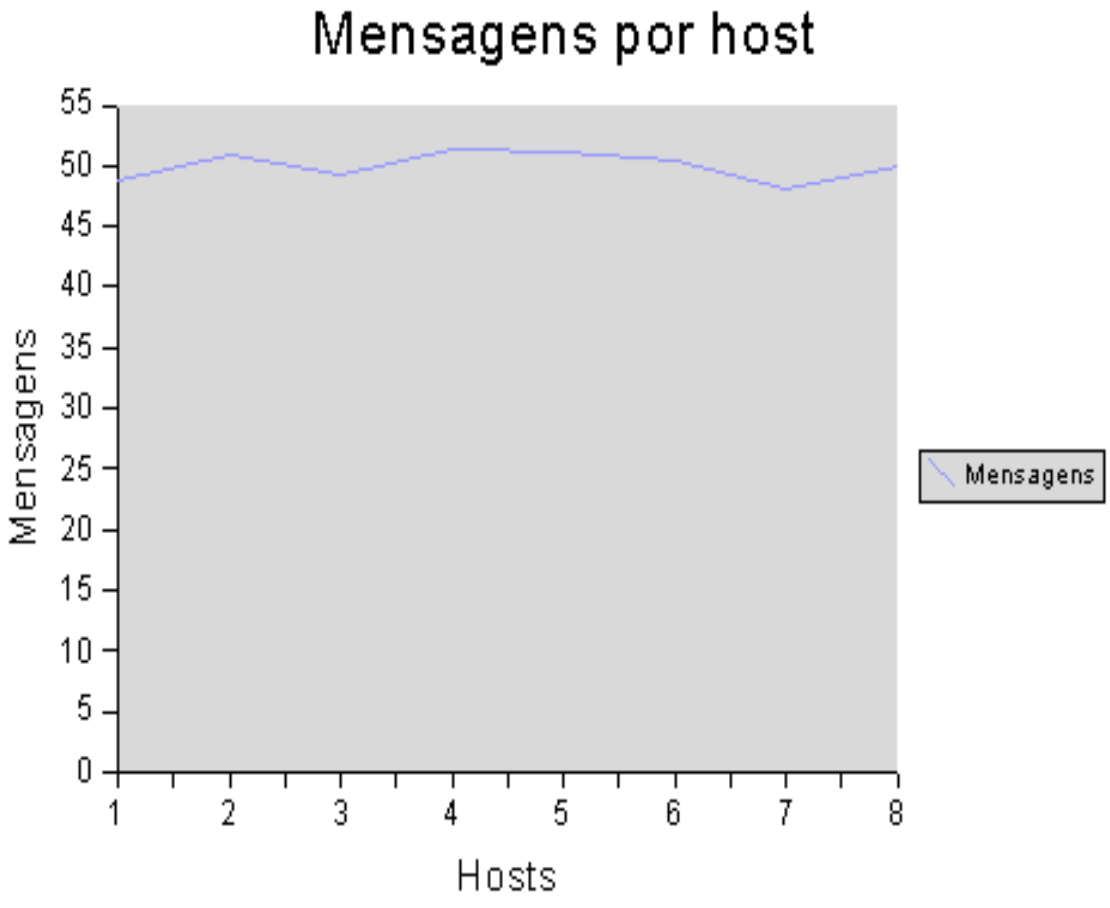
- Ordenação dos agentes em árvore (grau do vértice)



- Classe definida: `GraphColoring8VerticesCOP`

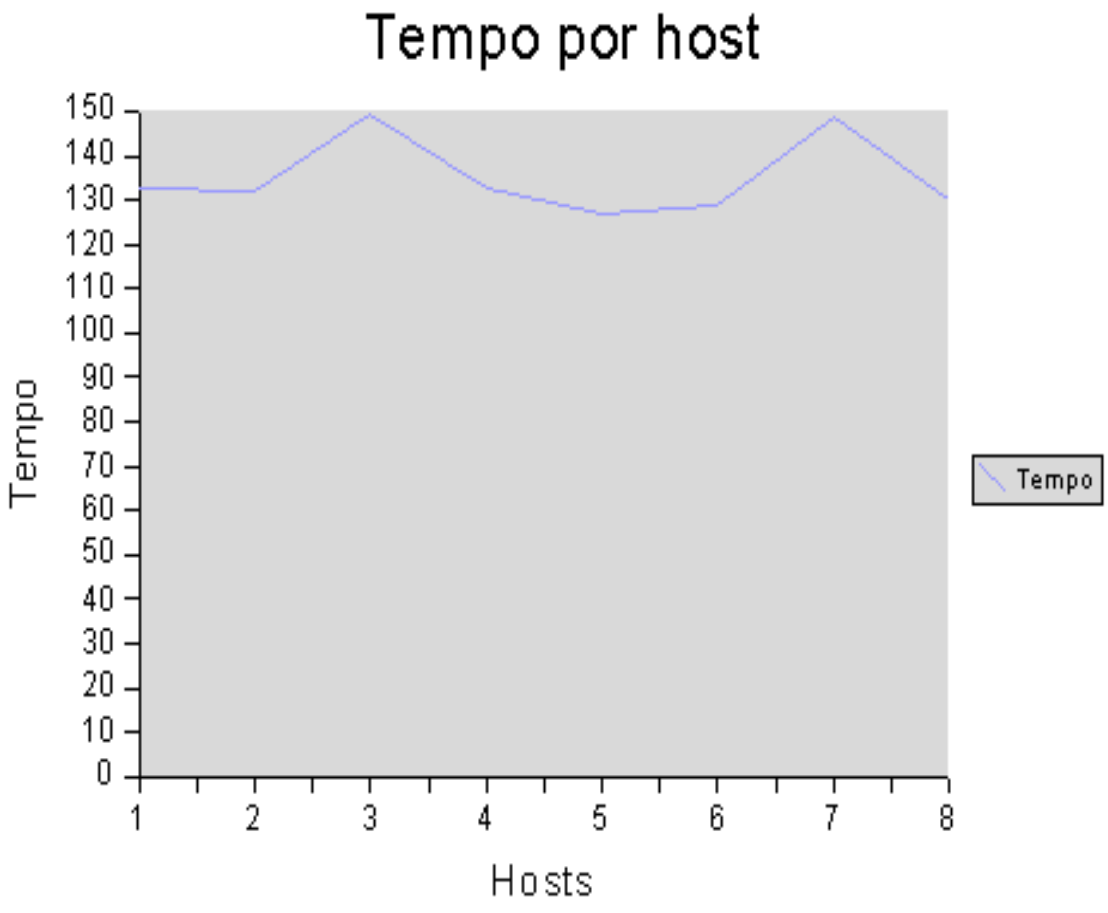
Coloração de Grafo - Resultados

- 10 execuções com 1, 2, ..., 8 *hosts*



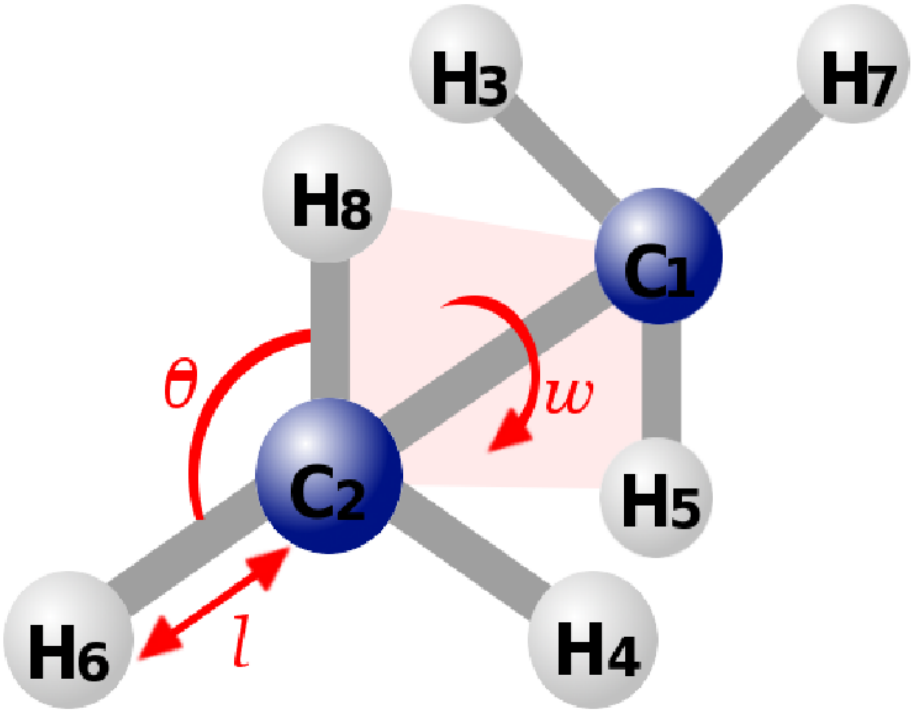
Coloração de Grafo - Resultados

- 10 execuções com 1, 2, ..., 8 *hosts*



Modelagem Molecular - Conceitos Avançados

- Átomos interagem entre si, produzindo energia



Estiramento

Torção

Dobramento

Átomos não ligados

Modelagem Molecular - Conceitos Avançados

- Cada tipo de interação/energia é definida por uma equação [Leach, 1996]

Equações de energias

$$E = E_{str} + E_{bend} + E_{tors} + E_{non-bonded}$$

- Modelo molecular é determinado quando a energia é mínima

Modelagem Molecular - Abordagem COP/DCOP

- Cada interação transformada em variável
  - Exemplos:

Estiramento	Dobramento	Torção
C1-H3	C1-C2-H4	H3-C1-C2-H4

  - Discretização dos domínios
- Interações entre átomos não ligados não é considerada
  - Modelo molecular aproximado

Funções de custo

Equações que definem energia em cada interação

Modelagem Molecular - Resultados

- Grafo de restrições denso
  - Árvore linear
  - Paralelismo prejudicado
- Teste com domínio reduzido
  - Solução correta
  - Elevado tempo de processamento, inviabilizando abordagem

Conclusões

- Flexibilidade de alterações do *framework* dynDCSP
- Implementação do Adopt realizada com sucesso
  - Primeira implementação efetivamente distribuída [Pearce, 2005]
- Adopt eficiente para problemas com grafo de restrições esparço
  - Inviável para modelagem molecular (na abordagem utilizada)

Extensões

- Depuração da implementação do algoritmo Adopt
  - Em algumas execuções com grafo diferente do utilizado, a execução pára (não determinismo)
  - Provável causa: alguma mensagem não enviada em função das otimizações
  - Dificuldade: depuração de *threads*
- Outras abordagens para problema da modelagem molecular
  - tipo de interação $\rightarrow$ variável
  - mensagem VALUE enviando valor calculado (equação)
- Domínios contínuos

GRUPO DE INTELIGÊNCIA ARTIFICIAL DA FURB (2005).

DynDCSP project.
Technical report, Blumenau.

Leach, A. R. (1996).
Molecular modelling.
Longman, Essex.

Modi, P. J. (2003).
Distributed constraint optimization for multiagent systems.
PhD thesis, Department of Computer Science, University of Southern California, Los Angeles, California.

Pearce, J. (2005).
Distributed constraint optimization problem (dcop).
Technical report, Los Angeles, California.

Russel, S. J. and Norvig, P. (2003).

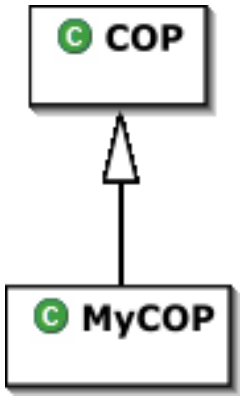
Artificial intelligence.
Prentice Hall, New Jersey, 2nd edition.


Tsang, E. (1993).
Foundations of constraint satisfaction.
Academic Press, London.

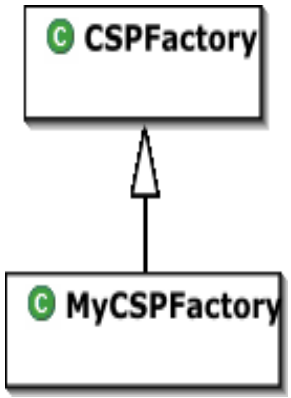

Yokoo, M. (2001).
Distributed constraint satisfaction.
Springer, Berlin.

framework dynDCSP - Operacionalidade

1. Definir um COP → estender classe `COP`



2. Definir um *Factory* → estender classe `CSPFactory`



framework dynDCSP - Operacionalidade

3. Alterar arquivo de configuração (incluir *factory*)

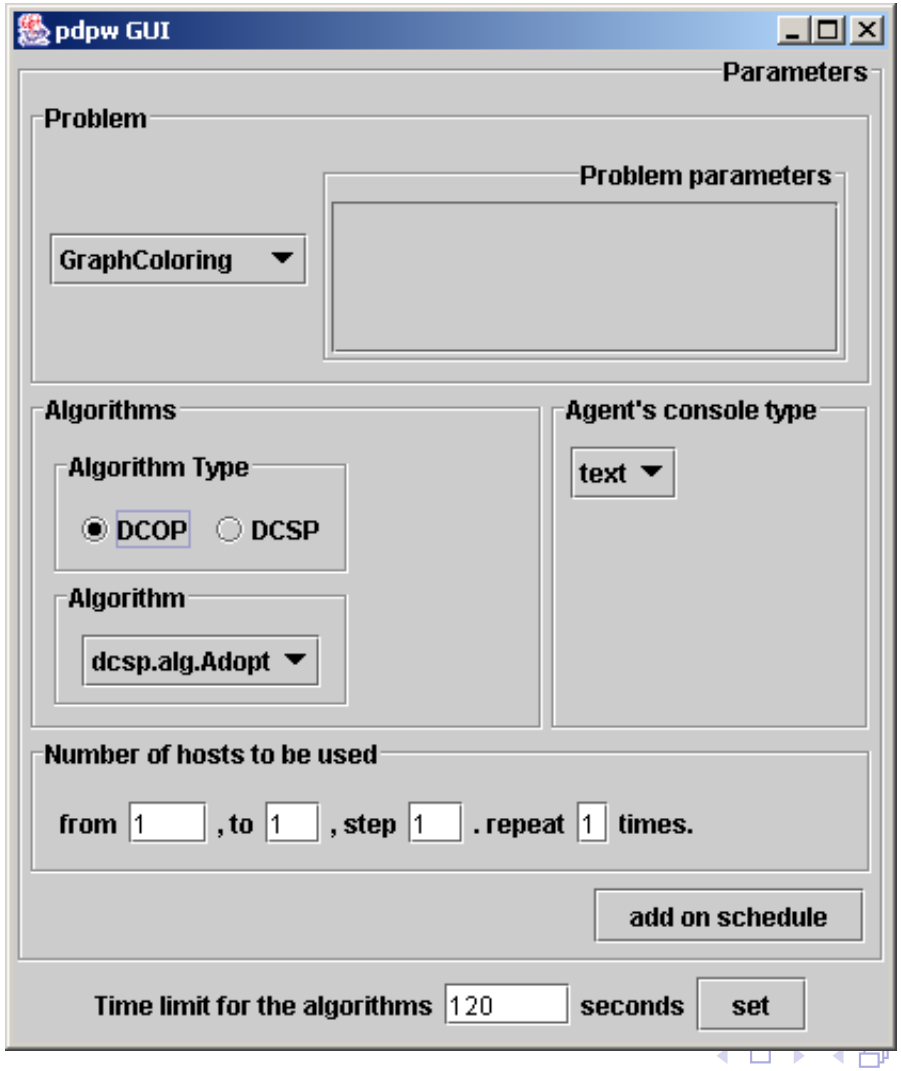
```
+problems
    helloCSP.HelloCSPFactory
    mSquare.MagicSquareCSPFactory
    nqueens.NQueensCSPFactory
    zebra.ZebraCSPFactory
    helloAdopt.HelloAdoptCSOPFactory
    fepk.FePKCSOPFactory
    molecules.WaterCSOPFactory
    molecules.EthaneCOPFactory
    graphColoring.GraphColoring8VerticesCOPFactory
```

4. Iniciar infraestrutura SACI
   - `$ant saci`

5. Executar *framework*
   - `$ant pdpw`

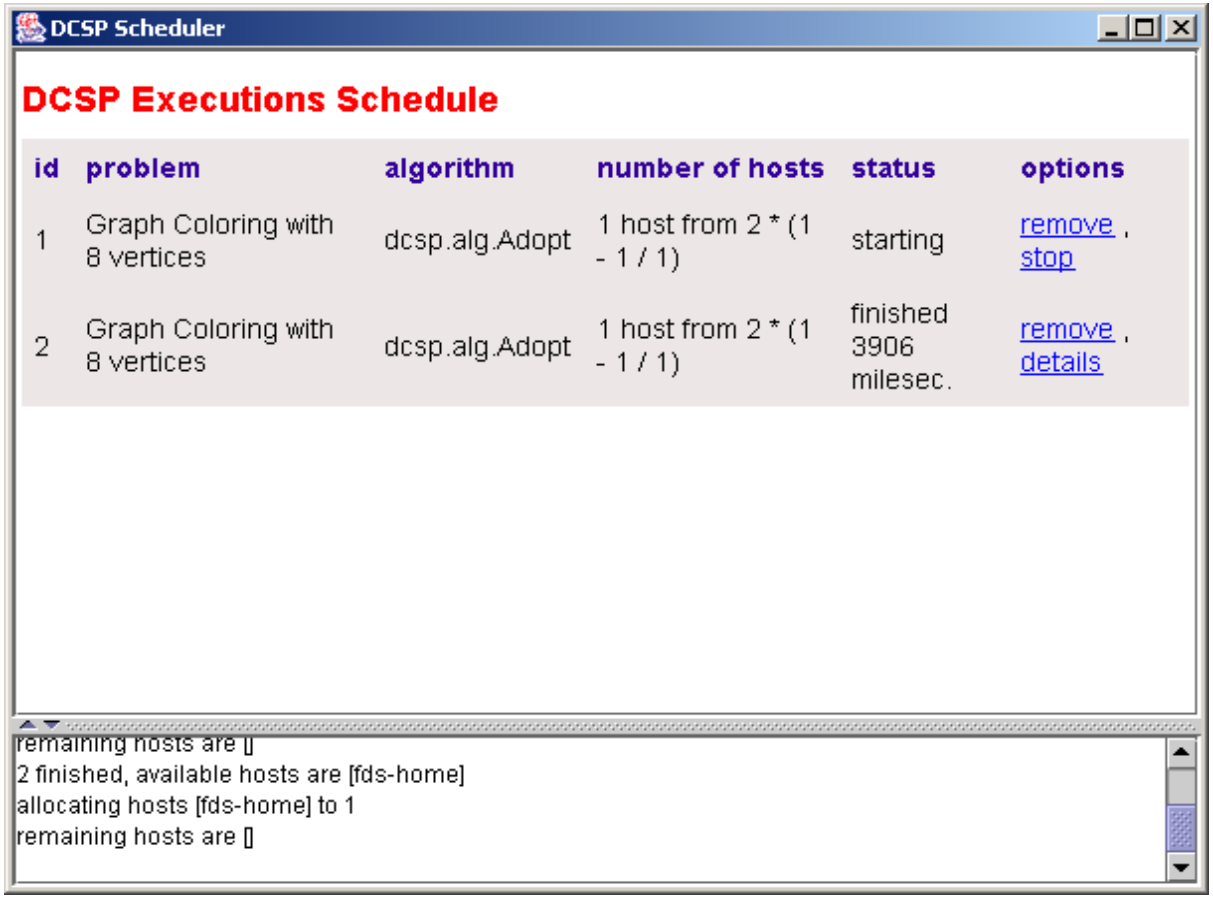
Algoritmo Adopt - Operacionalidade

6. `pdpwGUI`: Selecionar problema e algoritmo



Algoritmo Adopt - Operacionalidade

7. `SchedulerGUI`: Acompanhar execução



Algoritmo Adopt - Operacionalidade

8. `ManagerGUI`: Visualizar resultados