APLICAÇÃO DA TÉCNICA DE SATISFAÇÃO DE RESTRIÇÕES DISTRIBUÍDAS NO SINCRONISMO DE SEMÁFOROS DE UMA MALHA VIÁRIA

Orientando: Mauricio Bruns
Orientador: Jomi Fred Hübner

Roteiro da Apresentação:

- Introdução
- Fundamentação
- Desenvolvimento
- Implementação
- Resultados
- Conclusões

Introdução > Contextualização:

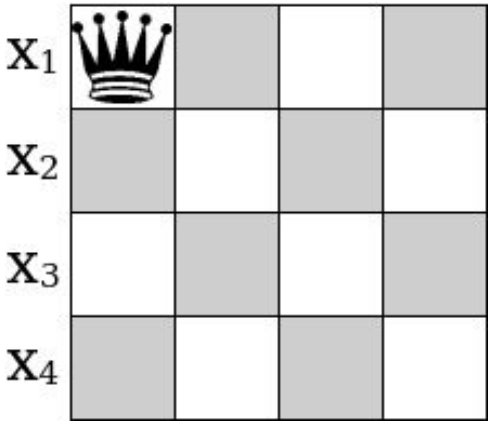
- **Trânsito**
  - Aumento da Demanda
  - Congestionamento, poluição, insegurança
  - Semáforos inteligentes, Sistemas de Informação
- **Problema de satisfação de restrições**
  - Variáveis
  - Domínio
  - Restrições
- **Problemas de satisfação de restrições distribuídas**
  - Conhecimento distribuído

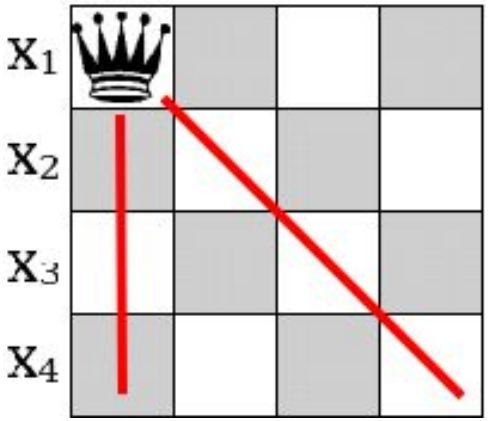
Introdução > Objetivos:

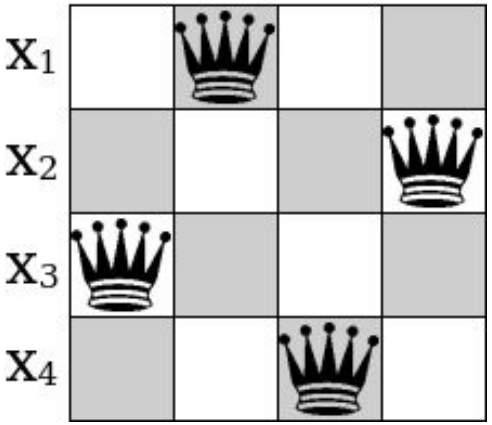
- **Sincronismo de semáforos**
  - Técnica de DCSP
- **DCSP**
  - Variáveis locais
  - Otimização
  - Restrições dinâmicas

Fundamentação:

- **Constraint Satisfaction Problem (CSP)**
  - Definição (TSANG 1993):
    - conjunto de variáveis
    - domínio
    - conjunto de restrições
  - Exemplo:
    - jogo das N-Rainhas

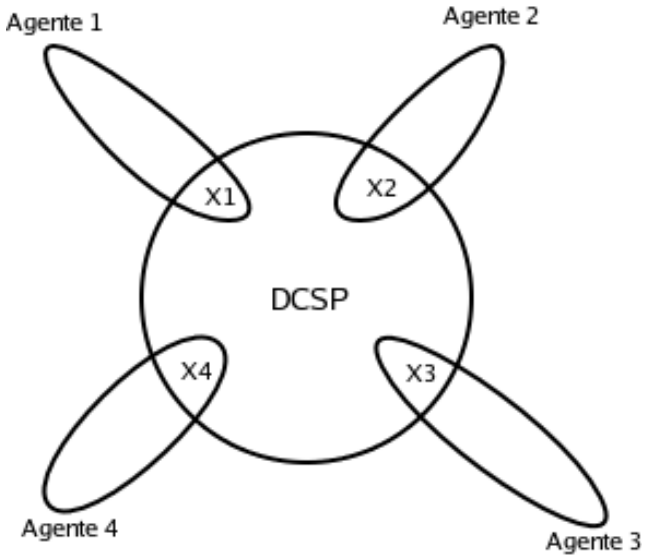






Fundamentação:

- **Distributed Constraint Satisfaction Problem (DCSP)**
  - Variáveis são distribuídas entre agentes automatizados
  - Conhecimento do problema encontra-se distribuído entre os agentes
  - Segurança, restrições geográficas, problemas dinâmicos

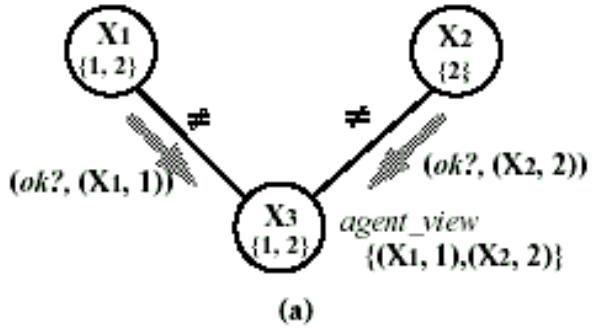


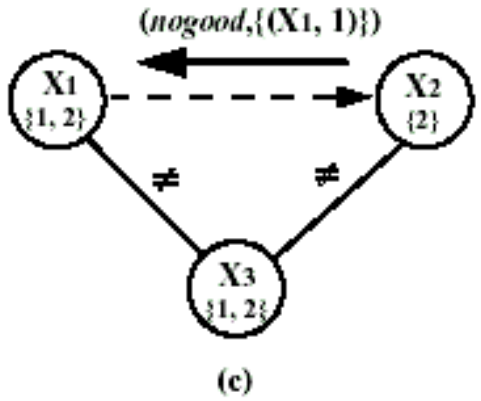
Fundamentação:

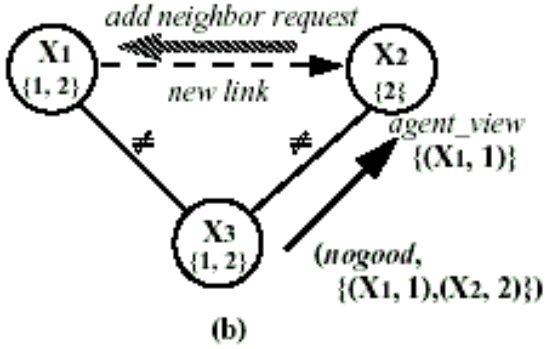
- **Algoritmo Asynchronous Backtracking (AB)**
  - Proposto por Yokoo (2001) e sua equipe para resolver problemas de DCSP
  - Agentes executam concorrentemente, de maneira assíncrona e sem controle global
  - Algoritmo completo, sempre encontrando uma solução caso ela exista
  - Características dos agentes:
    - um identificador único
    - um conjunto de agentes diretamente conectados chamados vizinhos (*outgoing* e *incoming*)
    - uma variável com domínio e um conjunto de restrições

Fundamentação:

- **Algoritmo Asynchronous Backtracking (AB)**
  - Execução do AB:







Fundamentação > Trabalhos Correlatos:

- **SincMobil:**
  - Projeto da Universidade Federal de Santa Catarina
  - Visa disponibilizar informações em tempo real sobre o trânsito urbano e o controle em tempo real dos semáforos para garantir o desempenho ótimo da malha viária
- **Sistema de Controle de Tráfego Urbano Utilizando Sistemas Multi-Agentes:**
  - Trabalho desenvolvido por Schmitz (2002)
  - Visa utilizar a técnica de Sistemas Multi-Agentes no controle de interseções semafóricas
  - Agentes negociam entre si o direito de passagem dos veículos no cruzamento

- **Desenvolvimento de um Algoritmo para Problema de Satisfação de Restrição Distribuída:**
  - Trabalho desenvolvido por Tralamazza (2004)
  - Especificou, implementou e analisou empiricamente os algoritmos para resolução de DCSP propostos por Makoto
  - Propôs alterações no AWC original para adaptação da heurística do menor valor utilizado, ordenação das restrições e não armazenamento de *nogoods* recebidos pelos agentes

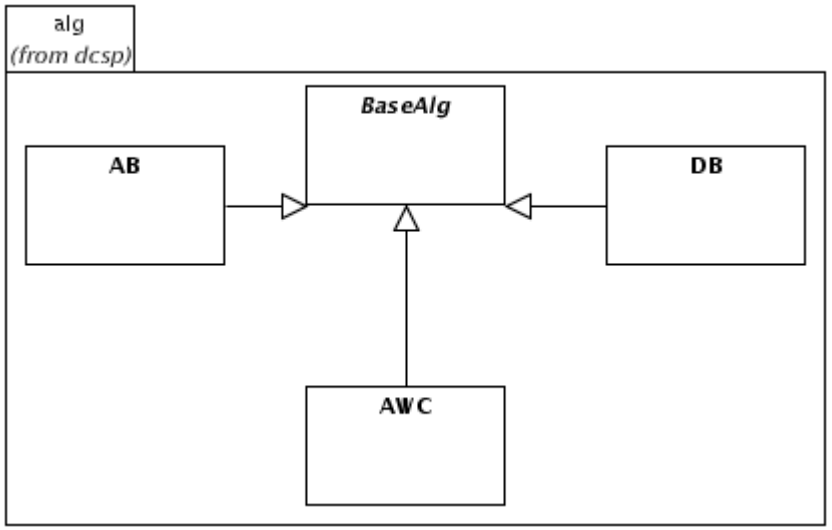
Desenvolvimento > Requisitos Principais

- Implementação de variáveis locais
- Implementação de restrições dinâmicas
- Implementação de otimização de DCSP
- Modelagem de um DSCP para resolução do problema de sincronismo de semáforos viários
- Permitir a criação de malhas viárias que serão utilizadas no simulador
- Obtenção de resultados rápidos para estados dos semáforos

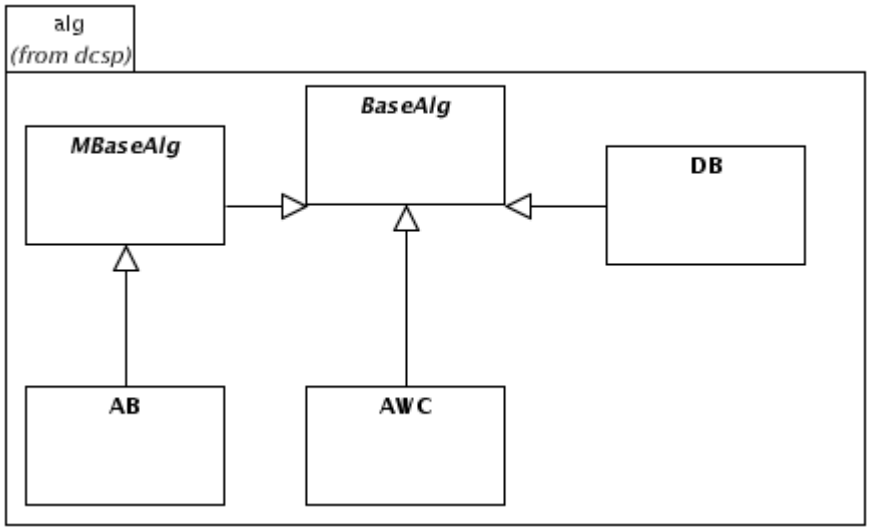
Desenvolvimento > Ferramentas Utilizadas

- Utilização do *framework* DynDCSP desenvolvido pelo grupo de pesquisa em Inteligência Artificial da FURB
- Implementação utilizando Java
- Utilização do ambiente de programação Eclipse

Modificação no pacote *dcsp.alg*

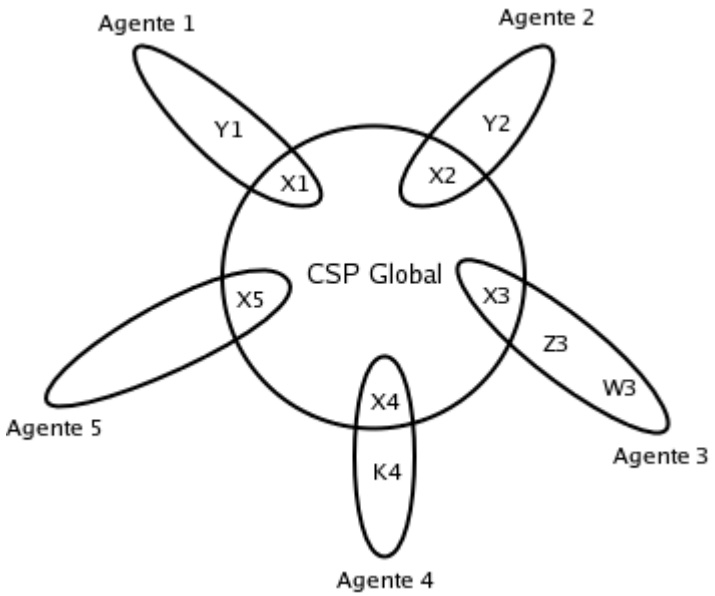


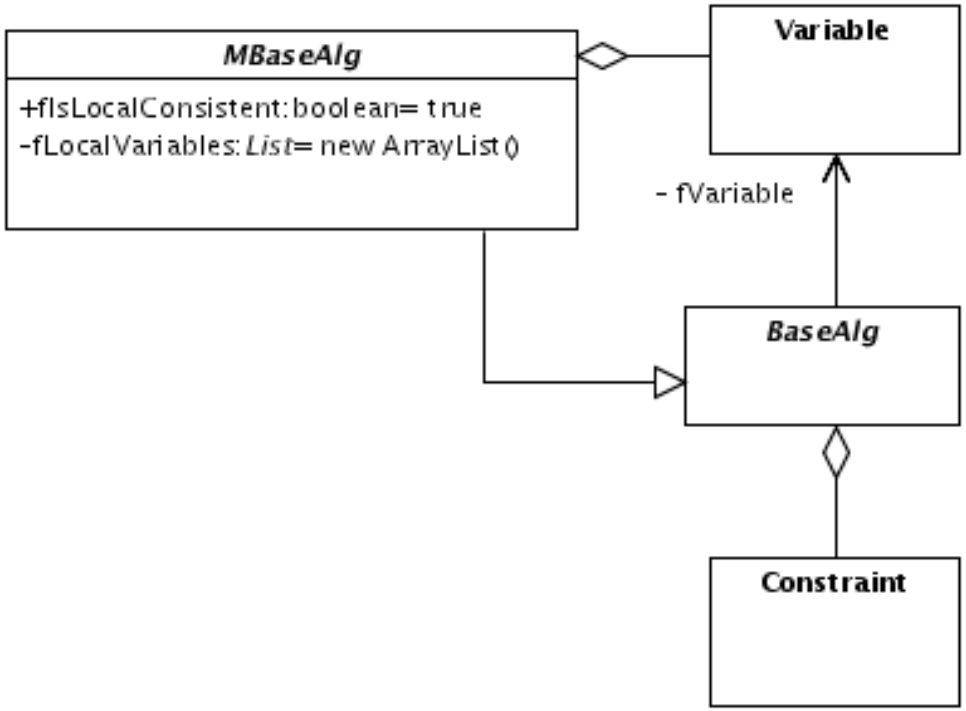
Pacote *dcsp.alg*



Pacote *dcsp.alg* modificado

Variáveis locais





Mensagens:

- addLocalVariables
- addLocalVariable
- putVariableValue
- delLocalVariable

Classe *MBaseAlg*

Variáveis locais

Exemplo adição de variáveis locais:

```
Iterator it = fSemaphores.iterator();
while (it.hasNext()) {
        MSemaphore msem = (MSemaphore) it.next();
        Message mLvar = createMessage( "addLocalVariable" );
        mLvar.put( "receiver", msem.getId() );
        mLvar.put( "localVariable",
              new Variable( "NrCarros" ) );
        mLvar.put( "value",
            new Value( new Double( msem.getQtdeCarros() ) ) );
        tell(mLvar, true);
        try {Thread.sleep(500);} catch (InterruptedException e) {}
}
```

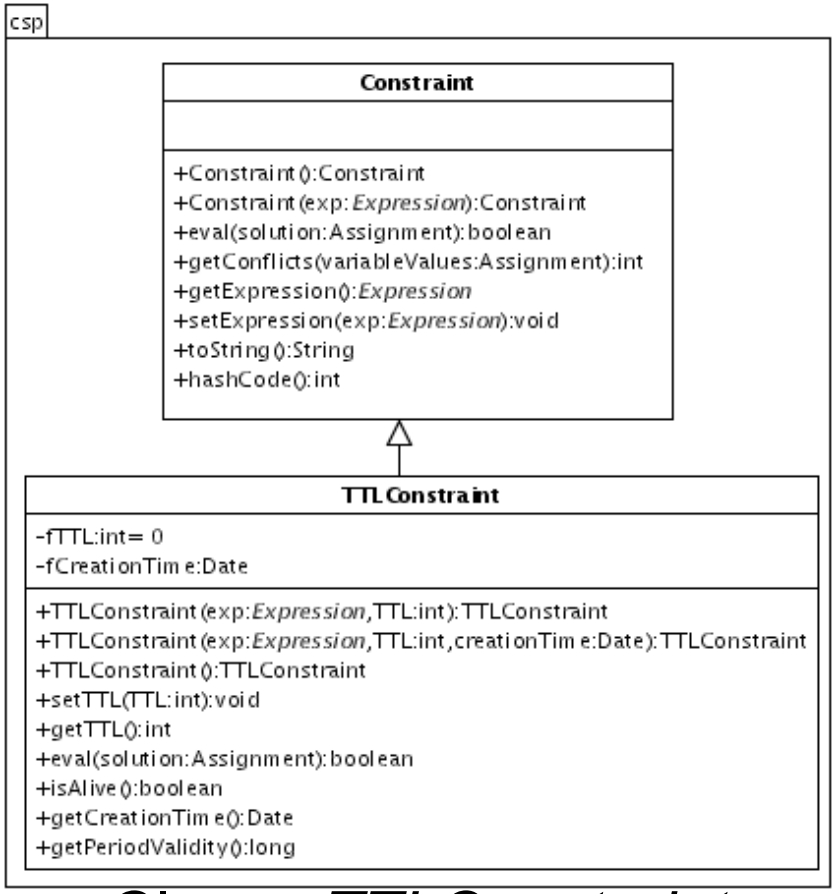
Variáveis locais

Exemplo de restrição sobre variáveis locais:

```
Message mDynConst = createMessage("addDynContraint");
mDynConst.put("receiver", msem.getId());
ConstantExpression ce1 = new ConstantExpression( new Double
                ( msem.getMaxCarrosEspera() ) );
VariableExpression ve1 = new VariableExpression( new Variable
                        ( "AAANrCarros" ) );
GreaterExpression ge = new GreaterExpression( ve1, ce1 );
VariableExpression ve2 = new VariableExpression( new Variable
                        ( msem.getId() ) );
ConstantExpression ce2 = new ConstantExpression( new Integer
                         (MSemaphore.VERDE) );
EqualsExpression ee = new EqualsExpression(ve2, ce2);
ImplicationExpression ie = new ImplicationExpression(ge, ee);

TTLConstraint co = new TTLConstraint(ie, 0);
mDynConst.put("dynConstraint", co);
tell(mDynConst, true);
```

Restrições Dinâmicas



Classe *TTLConstraint*

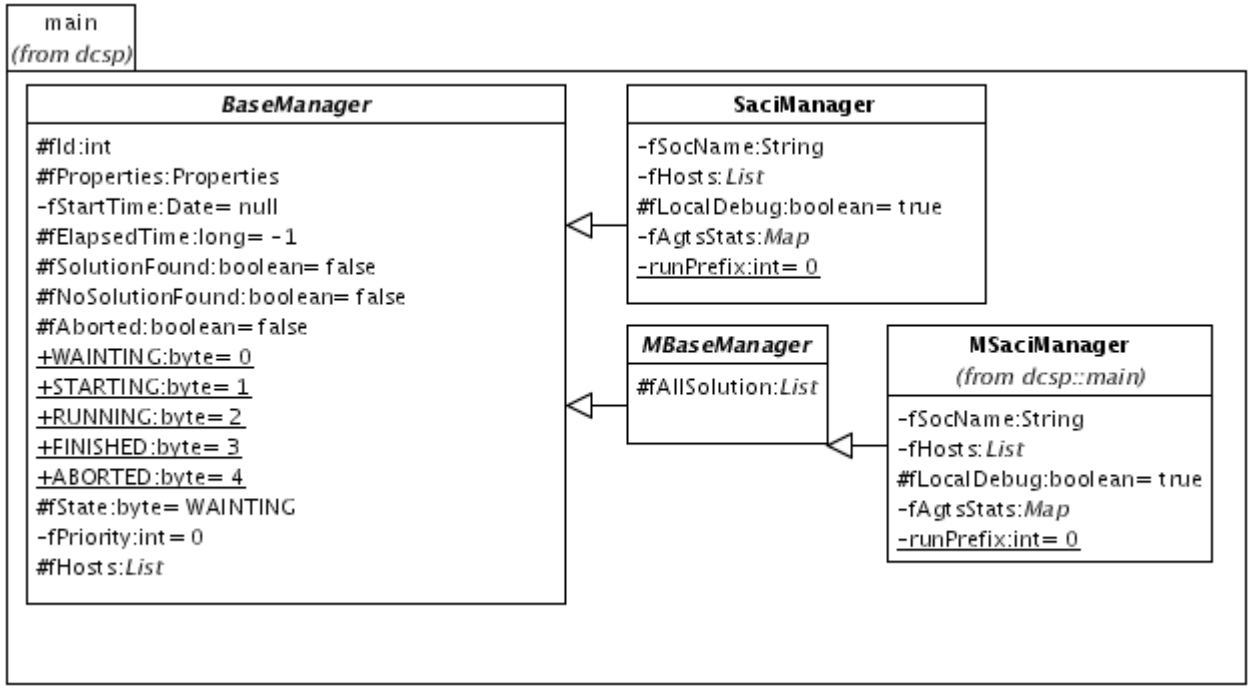
Mensagens:

- addDynConstraints
- addDynConstraint
- delDynConstraint

Características:

- Tempo de vida
- *Thread* que limpa restrições que não tem mais validade

Otimização



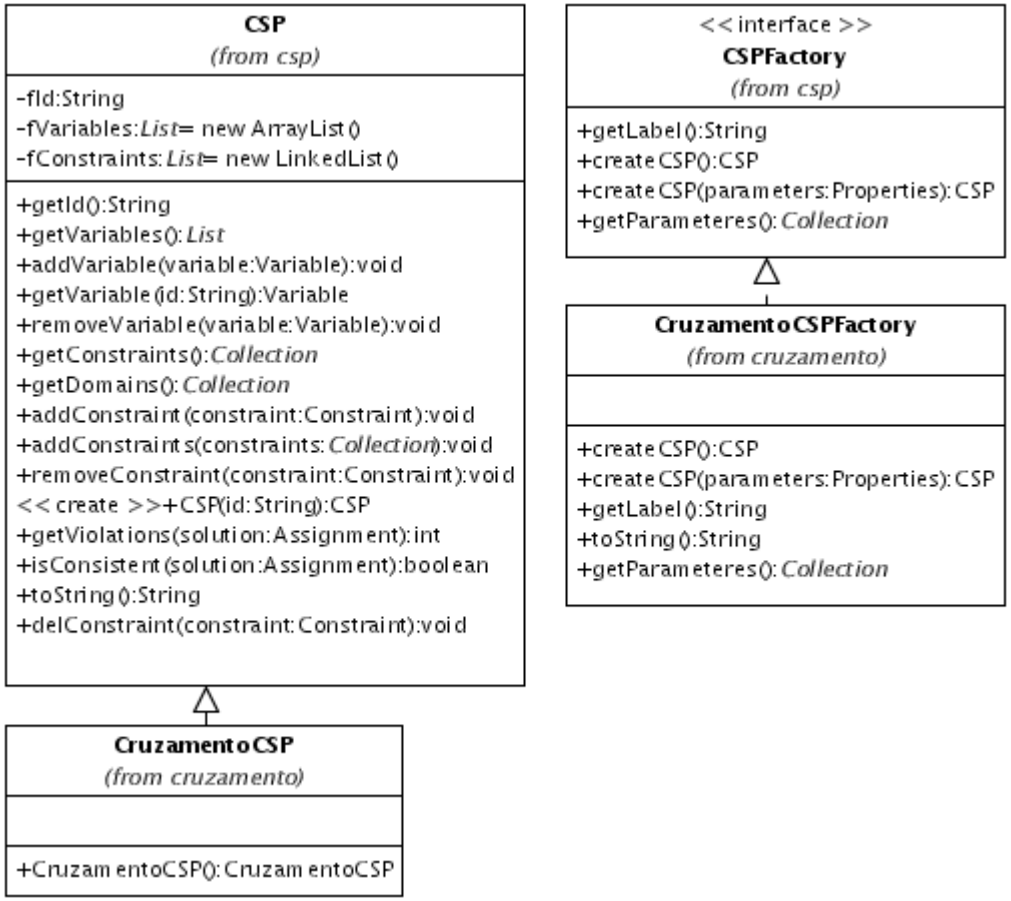
Classes *MBaseManager* e *MSaciManager*

Otimização

Funcionamento da busca por todas as soluções:

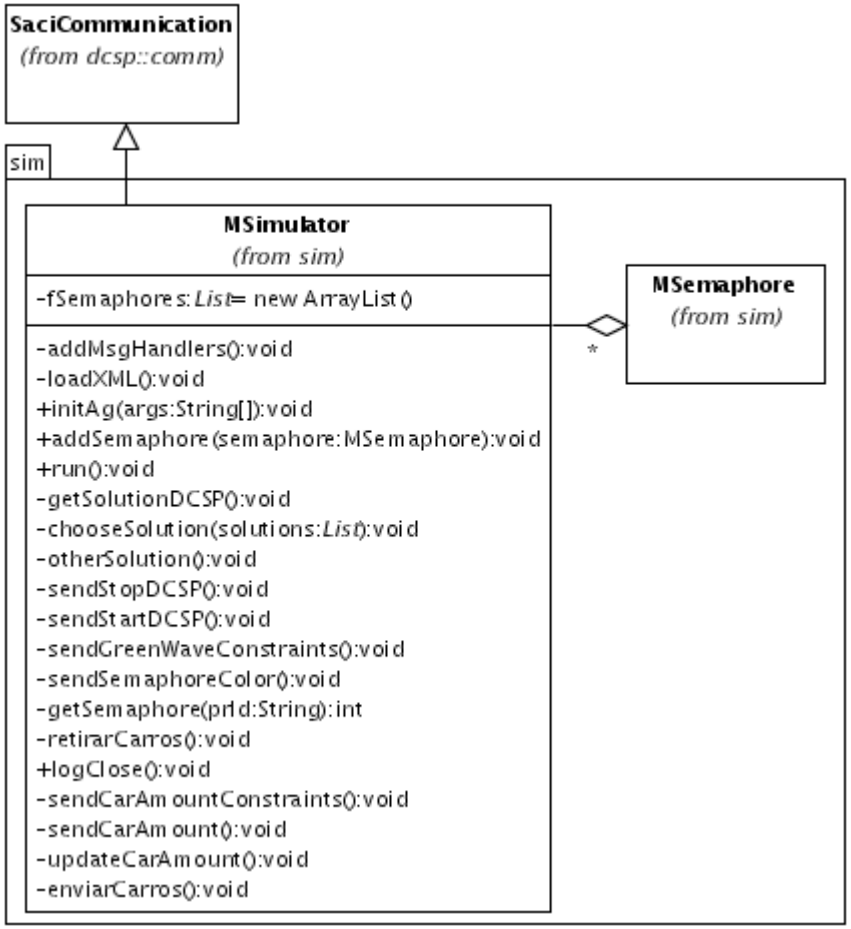
```
enquanto existir solução
        busca por solução
        se encontrou solução
                armazena solução
                envia solução aos agentes como solução inválida
        fim se
fim enquanto
```

CSP do Semáforo



Classes *CruzamentoCSP* e *CruzamentoCSPFactory*

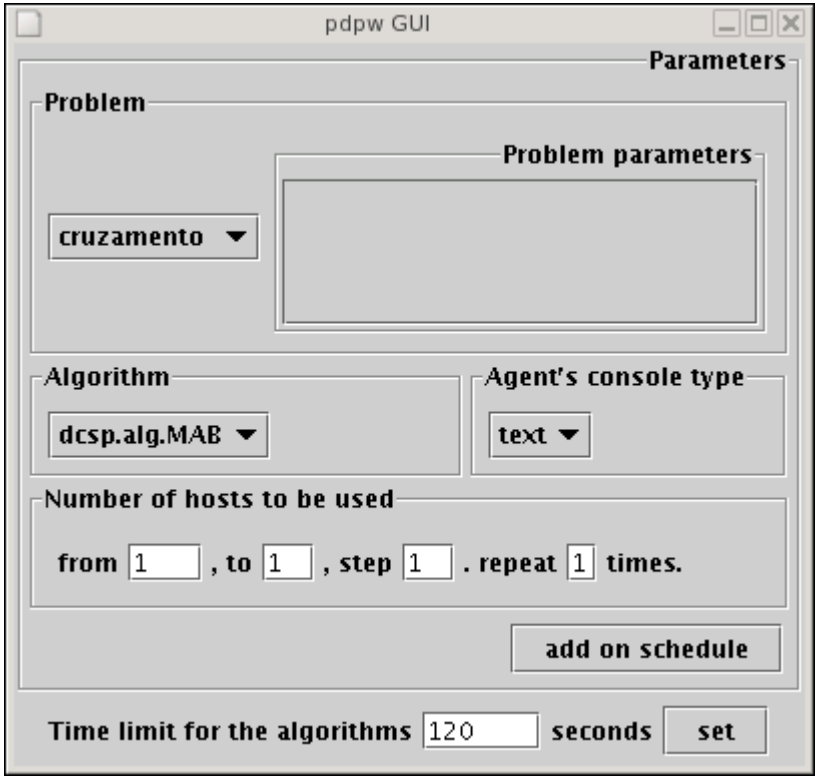
Simulador



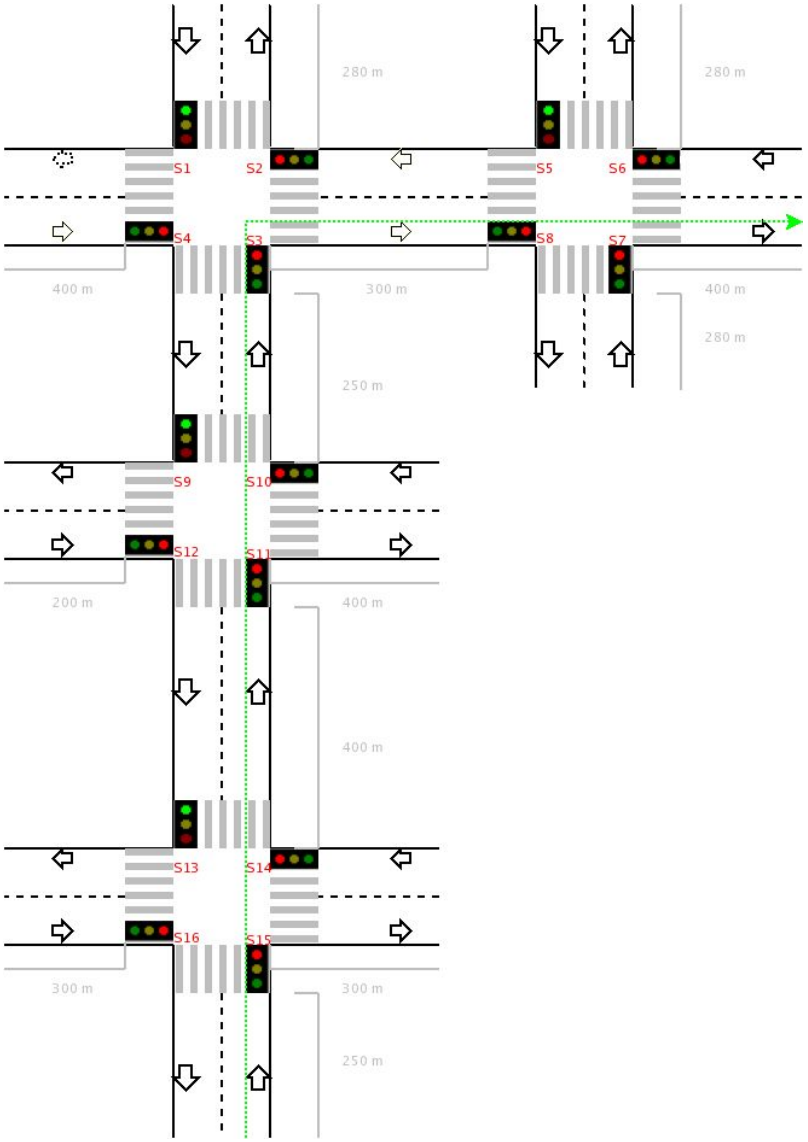
Classes *MSimulator* e *MSemaphore*

Operacionalidade:

- Arquivo XML com definição do ambiente a ser simulado
- Classes para execução do CSP
- Executar SACI
- Executar *framework* DynDCSP
- Escolher CSP do Cruzamento
- Iniciar execução



Ambiente a ser simulado:



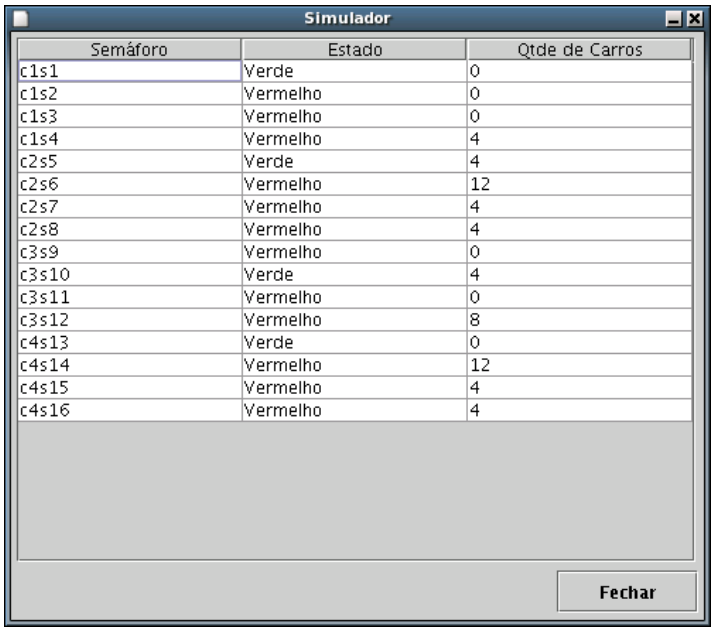
Arquivo XML com definição do ambiente

```
<Cidade>
  <cruzamento id_cruzamento="c1" verde="c1s1">
      <semaforo id_local="c1s1"  id_superior="c3s9" id_esquerdo="c2s8"
    id_direito="c0s0" id_inferior="c0s0" largura="10" velocidade="11"
    comprimento="280" verde="s1" prox_onda="" taxa_entrada="1"/>
      <semaforo id_local="c1s2"  id_superior="c0s0" id_esquerdo="c3s9"
    id_direito="c0s0"  id_inferior="c2s6" largura="10" velocidade="11"
    comprimento="300" verde="s1" prox_onda="" taxa_entrada="2"/>
      <semaforo id_local="c1s3"  id_superior="c0s0" id_esquerdo="c0s0"
    id_direito="c2s8" id_inferior="c3s11" largura="10" velocidade="11"
    comprimento="250" verde="s1" prox_onda="c2s8" taxa_entrada="1"/>
      <semaforo id_local="c1s4"  id_superior="c2s8" id_esquerdo="c0s0"
    id_direito="c3s9" id_inferior="c0s0"  largura="10" velocidade="11"
    comprimento="400" verde="s1" prox_onda="" taxa_entrada="1"/>
  </cruzamento>
...
</Cidade>
```

CSP para os cruzamentos

```
public CruzamentoCSP() {
        super("CruzamentoCSP");
        Integer verde    = new Integer(1);
        Integer vermelho = new Integer(2);
        // creating domains
        Domain semaphoresDomain = new IntegerDomain("semaphoresDomain",
                                verde.intValue(), vermelho.intValue());
        // Cria 4 cruzamentos.
        for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 4; j++) {
                        Variable av = new Variable("c" + (i + 1) + "s" + ((i * 4) +
                                                                        j + 1));
                        av.setDomain(semaphoresDomain);
                        addVariable(av);
                }
        }
        // Cria restrições para todos os semáforos do cruzamento 1
        ArrayList valuesC1 = new ArrayList(0);
        for (int i = 0; i < 4; i++) {
           valuesC1.add(new VariableExpression((Variable) getVariable("c1s" +
                                                                (i + 1))));
        }
        Expression atlC1 = new AtLeastExpression(valuesC1, 1, verde);
        Expression amC1 = new  AtMostExpression(valuesC1, 1, verde);
        addConstraint(new Constraint(atlC1));
        addConstraint(new Constraint(amC1));
        ...
}
```

Visualização do estado dos semáforos



Log com estado dos semáforos

Em Thu Jan 27 13:30:29 BRST 2005 c3s9 iniciou fase Vermelho
Em Thu Jan 27 13:30:30 BRST 2005 c3s10 iniciou fase Verde
Em Thu Jan 27 13:30:34 BRST 2005 c1s1 iniciou fase Vermelho
Em Thu Jan 27 13:30:34 BRST 2005 c1s4 iniciou fase Verde
Em Thu Jan 27 13:30:34 BRST 2005 c2s5 iniciou fase Vermelho
Em Thu Jan 27 13:30:34 BRST 2005 c2s8 iniciou fase Verde

Resultados

- Implementação de variáveis locais, restrições dinâmicas e busca por todas as soluções
- Sincronismo dos semáforos.
- Utilização de uma função auxiliar para decidir qual semáforo ficaria verde caso o CSP não fosse resolvido a tempo
- *Framework* DynDCSP implementa algoritmos mas utilizados para resolver um DCSP

Conclusões

- Modelagem utilizando CSP para o problema de sincronismo de semáforos viários
- Implementação de variáveis locais, restrições dinâmicas e obtenção de todos os resultados possíveis
- DCSP pode ser utilizado em vários tipos de problemas sem alteração nos algoritmos

Conclusões > Extensões

- **Simulador:**
  - enriquecimento da simulação
  - priorização de veículos especiais
  - desenvolvimento de uma interface gráfica para o simulador
  - definição dinâmica da onda verde
- ***Framework*:**
  - implementação de algoritmos para otimização de DCSP
  - implementação de variáveis locais com domínio

Bibliografia

- SCHMITZ, Marcelo. **Sistema de controle de tráfego urbano utilizando sistemas multi-agentes**. 2002. 62 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- TRALAMAZZA, Daniel. D**esenvolvimento de um algoritmo para problema de satisfação de restrição distribuída**. 2004. 37 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- TSANG, Edward. **Foundations of constraint satisfaction**. London: Academic Press, 1993. ISBN 0-12-701610-4.
- YOKOO, Makoto. **Distributed constraint satisfaction**. Berlin: Springer, 2001.