Universidade do Porto

Faculdade de Engenharia

FEUP

Recolha de Lixo numa Cidade

Relatório Final

Agentes e Inteligência Artificial Distribuída 2013/14

4º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

Vítor Silva – 201201624 – ei12166@fe.up.pt

Jorge Reis – 200800544 – ei08053@fe.up.pt

15 de Dezembro de 2013

ÍNDICE

OBJECTIVO

DESCRIÇÃO DO CENÁRIO E OBJECTIVO DO TRABALHO

Dado um mapa de uma cidade pretende-se simular a recolha de vários tipos de lixo por camiões (agentes) de diferentes especialidades. Os agentes circulam pelas estradas definidas até encontrar lixo que conforme o tipo, este é recolhido ou o agente responsável é notificado da sua localização, procedendo à recolha deste. Existe também em local fixo depósitos de lixo, onde os agentes quando a sua carga está no limite a possam libertar.

Obter uma recolha eficiente do lixo num sistema multi agente, de forma a que este se adapte a uma situação real em que os agentes cooperam entre si para resolver o problema. É espetável que diferentes estratégias como recolha com comunicação entre agentes ou estes possuírem ou não memória tenha impacto na eficiência da recolha do lixo. Para isso serão recolhidos dados de forma contínua para análise.

ESPECIFICAÇÃO

IDENTIFICAÇÃO E CARACTERIZAÇÃO DOS AGENTES

Dadas as características do projecto, escolheu-se para os agentes a arquitetura BDI (*beliefs*, *desires*, *intentions*). Esta representa uma arquitectura deliberativa baseada em estados mentais com origem no raciocínio prático humano, consistindo na implementação de conceitos de "crença, desejo e intenção". As crenças são a informação do estado do agente e da informação que armazena do meio envolvente; os desejos representam os estados que se pretende alcançar; e as intenções representam os componentes deliberativos que levam à decisão do curso de acção a ser tomado a partir do estado em que se encontra. Como tal, os agentes implementados (camiões), terão sempre conhecimento da sua localização, memorizarão os pontos de recolha de lixo à medida que vão passando pelos mesmos, terão capacidade de decisão quando notificados por outros agentes para recolha em determinados pontos, e terão sempre o objectivo de recolher a maior quantidade de lixo no menor espaço de tempo e, implicitamente, percorrendo também trajetos menores.

Resumidamente, estes agentes deverão circular livremente pelo cenário respeitando o mapa de estradas. Estes possuem capacidade de transporte limitada e apenas podem recolher determinado tipo de lixo. Conhecem a localização no mapa do ponto de descarga do lixo, e aquando da utilização deste ficam a saber o seu nível de ocupação.

Foram especificados ao todo cinco tipos de agente com diferentes responsabilidades:

- City
- Collector
- Container
- Burner
- Charts

City BDI

Identificação

Agente que representa a interface principal da aplicação servindo de meio de interação com o utilizador. Contém o mapa de estradas, um painel de configuração global e um painel para lançar novos agentes na aplicação.

Arquitetura

Objetivos:

PerformMapUpdate: Periódicamente recolhe informação sobre os agentes Container, Burner e Collector e procede à sua representação visual no mapa de estradas.

Responsabilidade:

Aquando criado o agente é responsável por inicializar a interface principal da aplicação. Esta inclui a leitura do mapa de estradas através de um ficheiro de texto e seu desenho na Grid.

Lançar novos agentes na aplicação em run-time com argumentos especificados pelo utilizador.

Comportamento:

Periodicamente atualiza a posição dos agentes representados na Grid e informação relevante.

Estratégias: Consultar os agentes e actualizar a representação destes.

Processos de raciocínio: Apenas persegue o seu objetivo se o critério de pausa se encontrar a falso, sendo este estabelecido pelo utilizador na interface principal, caso contrário suspende a sua execução.

Collector BDI

Identificação

Agente representativo de um camião de recolha de lixo. É especialista em um de quatro tipos diferentes de lixo: papel, vidro, metal ou indiferenciado. Têm uma determinada capacidade e têm como objectivo recolher lixo de *Containers* do seu tipo e depositá-lo num *Burner.* São os principais agentes da aplicação.

Arquitectura

Beliefs Principais: o seu tipo de lixo, própria localização, a sua capacidade, ocupação actual, memória de localização de *Burners* e *Containers* (pode ter ou não), e conhecimento das mensagens trocadas com outros agentes, o que permite utilizar os protocolos.

Goals:

PerformPatrol - caso nenhum outro objectivo se sobreponha, patrulha a cidade em busca de lixo do seu tipo nos containers respectivos;

CheckContainer - Ao passar por um *Container* o agente deve verificar se este é do seu tipo e se tem lixo.

GoToBurnerGoal - Quando a sua capacidade máxima é atingida, o objectivo primário passa a ser deslocar-se até um *Burner* a fim de depositar o lixo.

DumpGoal - Sempre que o agente (camião) passa por um *Burner* e tenha lixo consigo, deve aproveitar sempre para descarregá-lo. Isto acontece mesmo que não esteja cheio, por uma questão de eficiência.

Plans:

Wander - Plano responsável por fazer o agente circular pela cidade, abrangendo todos os pontos da mesma de uma forma eficiente, e que é utilizado quando existe o objectivo PerformPatrol.

PickUpWastePlan - Responsável por verificar a existência de Containers nas imediações, e caso existam e sejam do tipo do agente, recolhe o lixo existente até à sua capacidade máxima. Caso o camião já se encontre cheio, é enviada uma mensagem a todos os outros agentes-CollectorBDI com a indicação do tipo de lixo do Container e da sua localização, para que caso possam procedam à recolha do lixo. É o plano responsável pela manutenção do objectivo CheckContainer.

GoToBurnerPlan - Plano responsável por, quando o agente-camião se encontra cheio (e é assim criado o Goal GoToBurnerGoal), se dirigir para o Burner mais próximo a fim de lá depositar o lixo que contém. Caso o agente não possua memória, vagueará pela cidade de uma forma eficiente a fim de encontrar um Burner.

DumpWastePlan - Responsável por depositar o lixo que possui, sempre que passa por um Container do seu tipo de lixo.Dependente do Goal DumpGoal.

Comportamento:

Inicialmente passa por percorrer o mapa da cidade, implementado sobre um grafo, segundo uma pesquisa em profundidade. Verificar a cada ponto/iteração a existência de Containers e Burners, fazendo as verificações descritas anteriormente. É utilizada a comunicação entre os agentes sempre que algum verifica lixo num Container mas não o recolhe por se encontrar cheio ou não ser do seu tipo de lixo. Isto vai despoletar uma troca de mensagens com vista a se decidir qual o agente-camião que se encontra mais próximo e com possibilidade de ir recolher o lixo.

Container BDI

Identificação:

Agente que representa um contentor de lixo. Um pouco à semelhança dos agentes-camião, possui um de quatro tipos diferentes de lixo: papel, vidro, metal ou indiferenciado. Têm uma dada capacidade, e a quantidade actual e lixo vai sendo incrementada simulando um ambiente real.

Arquitectura

Beliefs Principais: o seu tipo de lixo, própria localização, a sua capacidade, e a ocupação actual.

Goals:

PerformCreateWaste - Goal permanente, que representa o objectivo de o valor de lixo do Container aumentar ao longo do tempo, simulando um ambiente real.

Plans:

CreateWaste - Plano responsável pela criação de lixo no *Container*.

Burner BDI

Identificação:

Agente que representa um Depósito/Incinerador de lixo. Responsável por receber lixo vindo dos camiões *CollectorBDI*, desfazendo-se dele ("queimando-o") de seguida.

Arquitectura:

Beliefs Principais: a sua própria localização, e a quantidade de lixo presente.

Goals:

DeleteWaste - Objectivo de se desfazer do lixo que é descarregado no mesmo.

Plans:

EmptyBurner - Plano encarregue de diminuir o lixo presente no *Burner*, simulando a sua incineração.

Charts BDI

Identificação

Agente que responsável por mostrar em "real-time" estatística sobre o lixo presente na cidade num dado momento.

Arquitetura

Objetivos:

ChartUpdate: Periodicamente recolhe informação sobre a quantidade de lixo presente na cidade e atualiza o gráfico.

Responsabilidade:

Aquando criado o agente é responsável por inicializar a interface contendo um gráfico de linhas que represente a evolução da quantidade de lixo total, nos *Collectors*, nos *Burners* e nos *Containers* ao longo do tempo.

Actualizar de 5 em 5 segundos esta informação.

Comportamento: Periódicamente atualiza o gráfico.

Estratégias: Consultar os agentes sobre a quantidade de lixo em cada um.

Processos de raciocínio: Apenas persegue o seu objetivo se o critério de pausa se encontrar a falso, sendo este estabelecido pelo utilizador na interface principal, caso contrário suspende a sua execução.

PROTOCOLOS DE INTERACÇÃO

A interacção dos *Collectors* com os *Containers* e *Burners* processa-se de forma automática, quando o mesmo se encontra numa posição adjacente deles.

Já a interacção entre *Collectors* obedece a um determinado protocolo, baseado num sistema de mensagens. Assim, quando um *Collector* passa por um *Container* que se encontra com a sua capacidade superior a meio e o mesmo não pode recolher (por estar cheio ou não ser do tipo encarregue da recolha desse contentor), este envia uma mensagem para todos os agentes *Collectors*, com o tipo de lixo e localização. Os agentes, ao receberem esta mensagem, irão primeiramente verificar se se trata do seu tipo de lixo, e caso seja, calcular a sua distância ao *Container* referido, enviando uma mensagem para todos os *Collectors* com a sua distância. Ao receberem as mensagens, facilmente conseguem perceber qual o agente que se encontra mais próximo, e é apenas esse que se descola ao *Container*, aumentando significativamente a eficiência do processo de recolha.

De salientar que cada mensagem tem um número de registo, utilizado também nas respostas. Tal facto impossibilita que haja interferências de outras mensagens, sabendo assim cada agente a que mensagem se refere.

DESENVOLVIMENTO

PLATAFORMA/FERRAMENTA UTILIZADA & AMBIENTE DE DESENVOLVIMENTO

SO: Ubuntu e Windows
IDE: Eclipse Kepler

Ferramentas/Frameworks: Jadex e JFreeChart

O *Jadex* foi utilizado pelas suas componentes de SMA para representação de agentes BDI. É uma biblioteca orientada para o desenvolvimento de sistemas de agentes computacionais inteligentes com recurso às linguagens de programação XML e Java. Estes agentes representam componentes activos com capacidade de raciocínio, podendo apresentar um comportamento reactivo ou um comportamento deliberativo.

A característica mais interessante e utilizada desta plataforma é o seu motor de raciocínio BDI, que permite a criação de agentes deliberativos mais complexos e dinâmicos baseados neste tipo de arquitectura, sendo guiados por crenças, objectivos e planos *(belief, goals, plans)*. O Jadex permite ainda criar sistemas multi-agente distribuídos baseados num conjunto de bibliotecas de comunicação.

JFreeChart, é uma biblioteca java para representação de gráficos de linhas e exportação dos mesmos.

ESTRUTURA DA APLICAÇÃO

Todos os agentes são independentes uns dos outros, na medida em que todos executam os seus planos sem estarem dependentes de outros agentes, i.e., é possível inicializar um *Collector*, sem que ainda esteja definido o mapa de estradas (*City BDI*), no entanto a sua execução só faz sentido quando outros agentes estão também inicializados.

A interface é dotada de mecanismos para criar outros agentes e editar o ambiente dos mesmos.

MÓDULOS

Agentes: Especificação dos diversos agentes seus goals, e estratégia de seleção dos mesmos.

Planos: Todos os goals têm um plano associado, sendo estes definidos para cada ação que um agente pode executar.

Lógica: Neste módulo são definidos todos os tipos necessários para se poder modular o problema proposto, sendo constituído principalmente por um singleton que contém toda a informação relevante à aplicação e seu estado e um Grafo para representar o mapa de estradas.

Interfaces: Todas as interfaces e seus componentes externos ao Jadex são especificados neste módulo.

DIAGRAMA DE CLASSES

Ficheiro: class_diagram.png

DETALHES RELEVANTES DA IMPLEMENTAÇÃO

Devido a um bug ainda não identificado a criação dos agentes pode falhar, tanto pela interface como pelo Jadex, sugere-se por isso que caso aconteça se termine a execução e inicie uma nova. Na criação dos agentes dar algum tempo entre cada um (geralmente apenas um ou dois segundos será suficiente) de modo a tentar evitar estes erros.

EXPERIÊNCIAS

Serão apenas considerados para os níveis de sujidade os valores dos containers, pois estes representam o lixo não recolhido.

Para cada uma das seguintes experiências foi utilizado o seguinte set up:

- ❏ 4 Containers
  - ❏ 1 x Comom, cap: 100, pos(1,3)
  - ❏ 1 x Metal, cap: 100, pos(5,2)
  - ❏ 1 x Paper, cap: 100, pos(1,8)
  - ❏ 1 x Plastic, cap: 100, pos(5,8)

- ❏ 4 Collectors
  - ❏ 1 x Comom, cap: 125
  - ❏ 1 x Metal, cap: 125
  - ❏ 1 x Paper, cap: 125
  - ❏ 1 x Plastic, cap: 125

- ❏ 1 Burner
  - ❏ pos(3,7)

- ❏ Mapa inicial carregado por ficheiro

OBJECTIVO DE CADA EXPERIÊNCIA

a) Avaliar quantidade de lixo máxima, minima e média sem fazer uso da comunicação e memória
b) Avaliar quantidade de lixo máxima, minima e média fazendo uso apenas da comunicação
c) Avaliar quantidade de lixo máxima, minima e média fazendo uso da comunicação e memória
d) Avaliar quantidade de lixo máxima, minima e média fazendo uso apenas da memória

RESULTADOS

a) Ficheiro: statistics/no_mem_no_com.png
   - Max: 220/400
   - Min: ~90/400
   - Média: ~170/400
   - Resultado: satisfatório

b) Ficheiro: statistics/no_mem_com.png
   - Max: 270/400
   - Min: 30/400
   - Média: ~180/400
   - Resultado: satisfatório

c) Ficheiro: statistics/mem_com.png
   - Max: 270/400
   - Min: 80/400
   - Média: ~145/400
   - Resultado: bom

d) Ficheiro: statistics/mem_no_com.png
   - Max: 278/400
   - Min: 93/400
   - Média: ~140/400
   - Resultado: bom

CONCLUSÕES

DA ANÁLISE DOS RESULTADOS DAS EXPERIÊNCIAS LEVADAS A CABO

Devido ao elevado número de configurações possíveis torna-se impossível determinar uma solução ótima, no entanto usando um mapa relativamente pequeno com apenas quatro C*ollectors* e quatro *Containers* um de cada tipo é possível verificar algumas vantagens da utilização de memória e comunicação.

Havendo memória o agente é capaz de se dirigir automaticamente (pelo caminho mais curto) para o *burner* quando tem o seu depósito cheio, em vez de vaguear até o encontrar. Resultando num ganho de eficiência.

Havendo comunicação entre os agentes, torna-se possível notificar o agente responsável da existência de lixo, para que proceda à recolha do mesmo, resultando em melhores mínimos de quantidade de lixo.

Como o plano *Wander* (vaguear) faz com que o agente percorra todos os locais no mapa pelo menos uma vez é previsível que num mapa com um número de estradas considerável as vantagens do uso da comunicação e memória sejam ainda maiores, visto este ser interrompido quando por exemplo outro agente envia uma mensagem a notificar da localização de lixo.

MELHORAMENTOS

Existem alguns problemas de inicialização de agentes que acontecem de vez em quando, sendo difícil de replicar o erro. Pensa-se que tenha a ver com a versão usada da plataforma ainda ser versão Beta. No entanto é um ponto a melhorar se for possível contornar o problema.

Seria também interessante estender o mapa, para que este suporta-se estradas de dois sentidos e gravasse em ficheiro as alterações efetuadas.

RECURSOS

BIBLIOGRAFIA

[1] Recursos disponibilizados na página da unidade Curricular de Agentes e Inteligência Artificial Distribuída (AIAD), *http://paginas.fe.up.pt/~eol/AIAD/aiad1314.html,* ano lectivo 2013/14.

[2] Lars Braubach, Alexander Pokahr, Kai Jander, *Jadex Active Components*, *http://siod.net/bin/view/BDI+V3+Tutorial/01+Introduction*, online em Dezembro 2013.

[3] João Lopes, *Jadex wiki*, *http://paginas.fe.up.pt/~ei10009/jadex/doku.php*, online em Dezembro de 2013.

[4] José Pedro Silva, *Tutorial Jadex – Agentes e Inteligência Artificial Distribuída*, Setembro 2012.

[5] Mehdi Dastani, *Jadex: A BDI Reasoning Engine*, *http://www.cs.uu.nl/docs/vakken/map/slides/jadex.pdf*, acedido em Outubro 2013.

SOFTWARE UTILIZADO

Ubuntu 13.10
Windows 8
Eclipse IDE
GIT (Controlo de Versões)
Plataforma Jadex [2]
Google Docs
JFreeChart

DIVISÃO DO TRABALHO EM TAREFAS

Vítor Silva

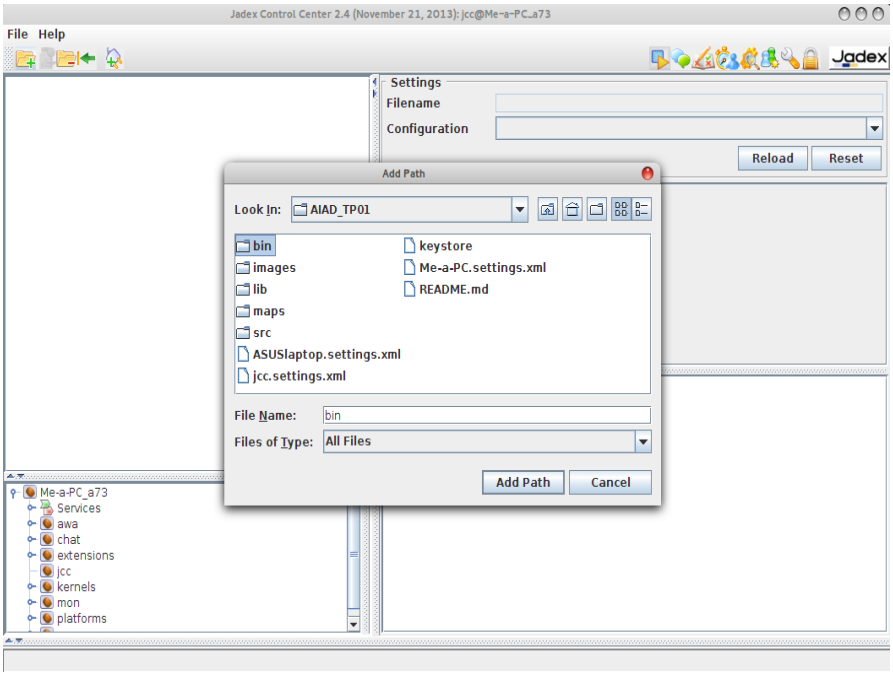
a. Lógica do modelo e fluxo de execução
b. Grafo e operações sobre o mesmo
c. Mapa de estradas e parser de ficheiro de entrada
d. Planos: Wander, MapUpdate, EmptyBurner
e. Agentes: CityBDI, ChartsBDI
f. Interfaces: desenho componentes, interação utilizador aplicação e funcionalidades

Jorge Reis

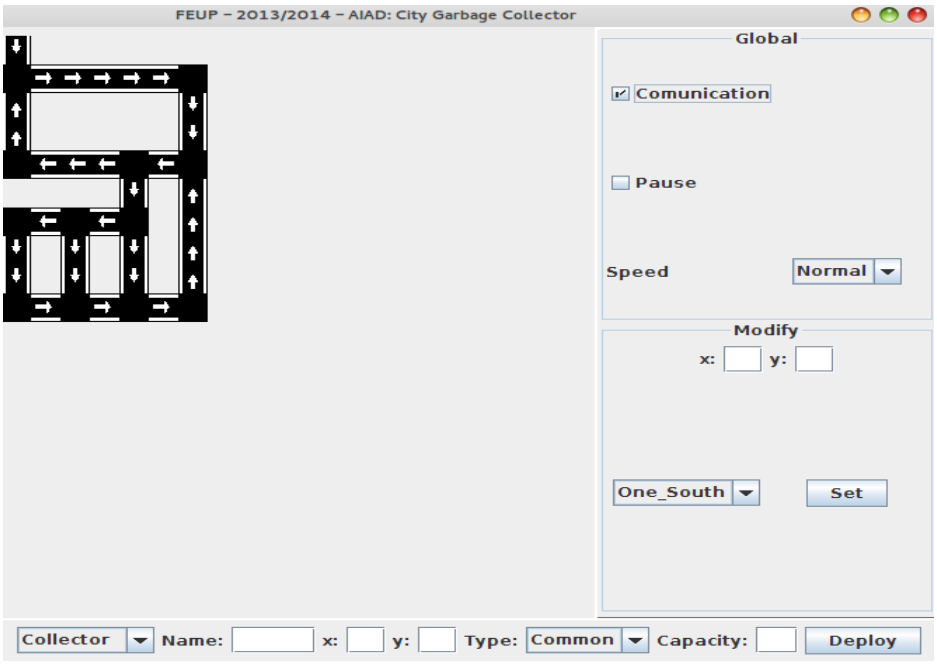
a. Lógica de Recolha e Depósito do lixo com base em agentes BDI
b. Agentes: *CollectorBDI*, *ContainerBDI*, *BurnerBDI*
c. Hierarquia de Goals dos respectivos agentes
d. Planos: *PickUpWaste*, *GoToBurner*, *DumpWastePlan*, *CreateWaste*
e. Configuração de *ChatService* para troca de mensagens
f. Comunicação e respectivo protocolo entre *Collectors*

APÊNDICE (MANUAL DO UTILIZADOR - SUCINTO)

1. Inicialização
    a. Arrancar a plataforma Jadex
    b. Adicionar a pasta bin (presente no root do projeto)



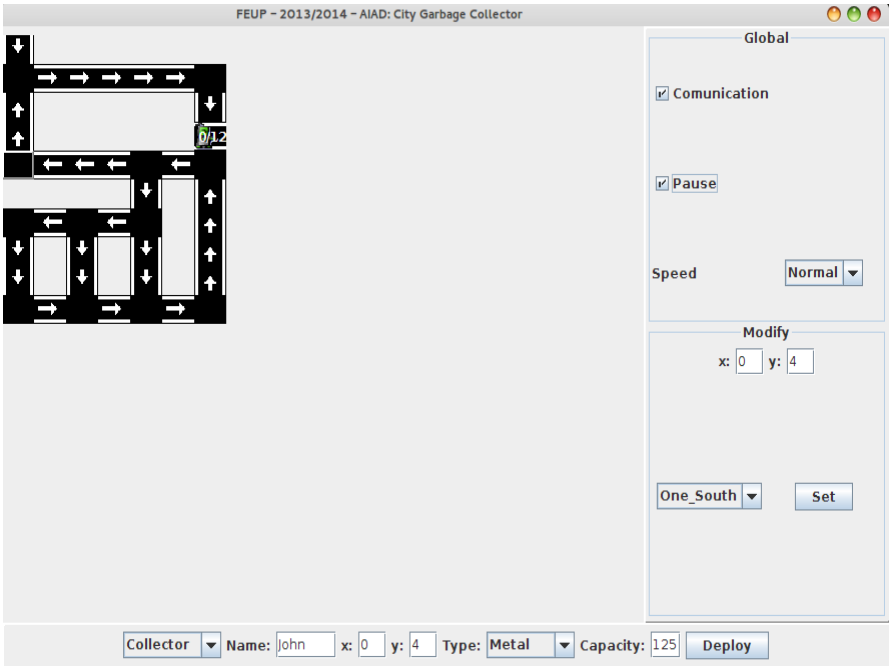
c. Expandir bin → cityGarbageCollector → agent, selecionar CityBDI.class e clicar em start (presente do lado direito), deverá de aparecer a seguinte janela



d. No painel inferior é possível lançar novos agentes, onde é possível escolher a sua função, nome, localização (x,y), tipo de lixo e capacidade. De notar que todos estes campos são opcionais pois existem valores por defeito para os mesmos.

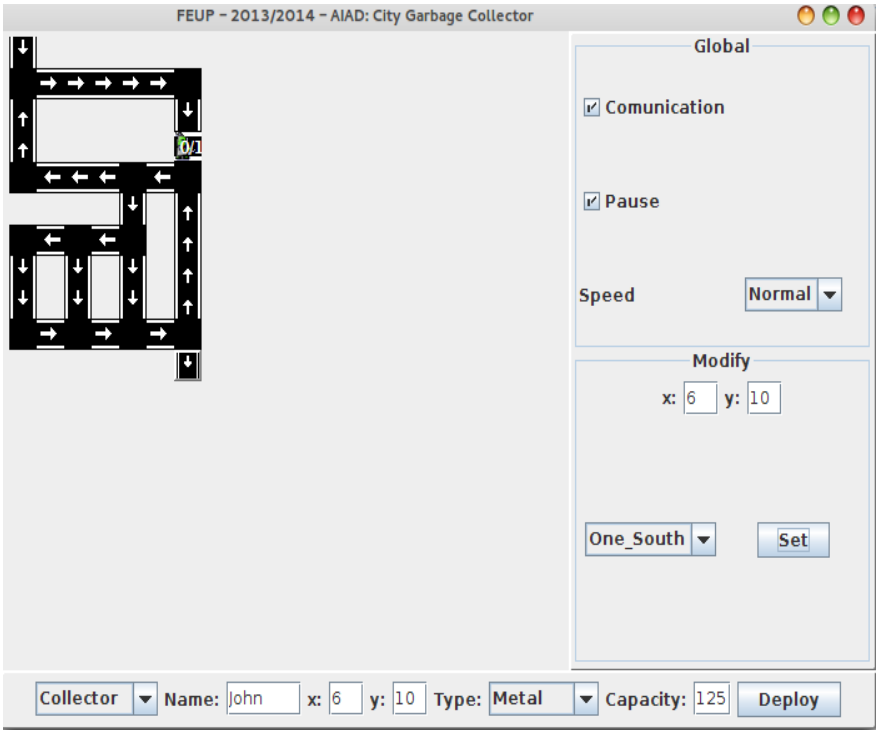
Dica: É possível selecionar uma posição no mapa através de um click para preencher automaticamente os campos x e y.

e. Clicando em "Deploy" deve de aparecer o agente especificado



f. **Dica**: A qualquer momento é possível parar a execução clicando em pause do lado direito, e retomar com um novo click.

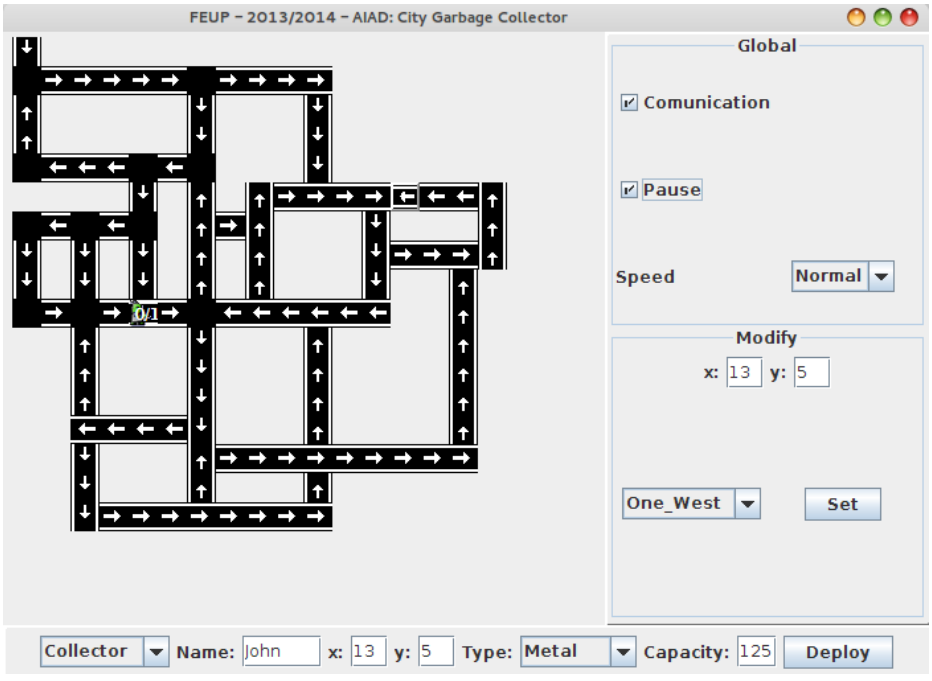
2. Modificar Mapa
   a. Escolher no mapa a posição a modificar do lado direito no grupo "Modify" escolher o tipo de estrada e clicar em "Set"



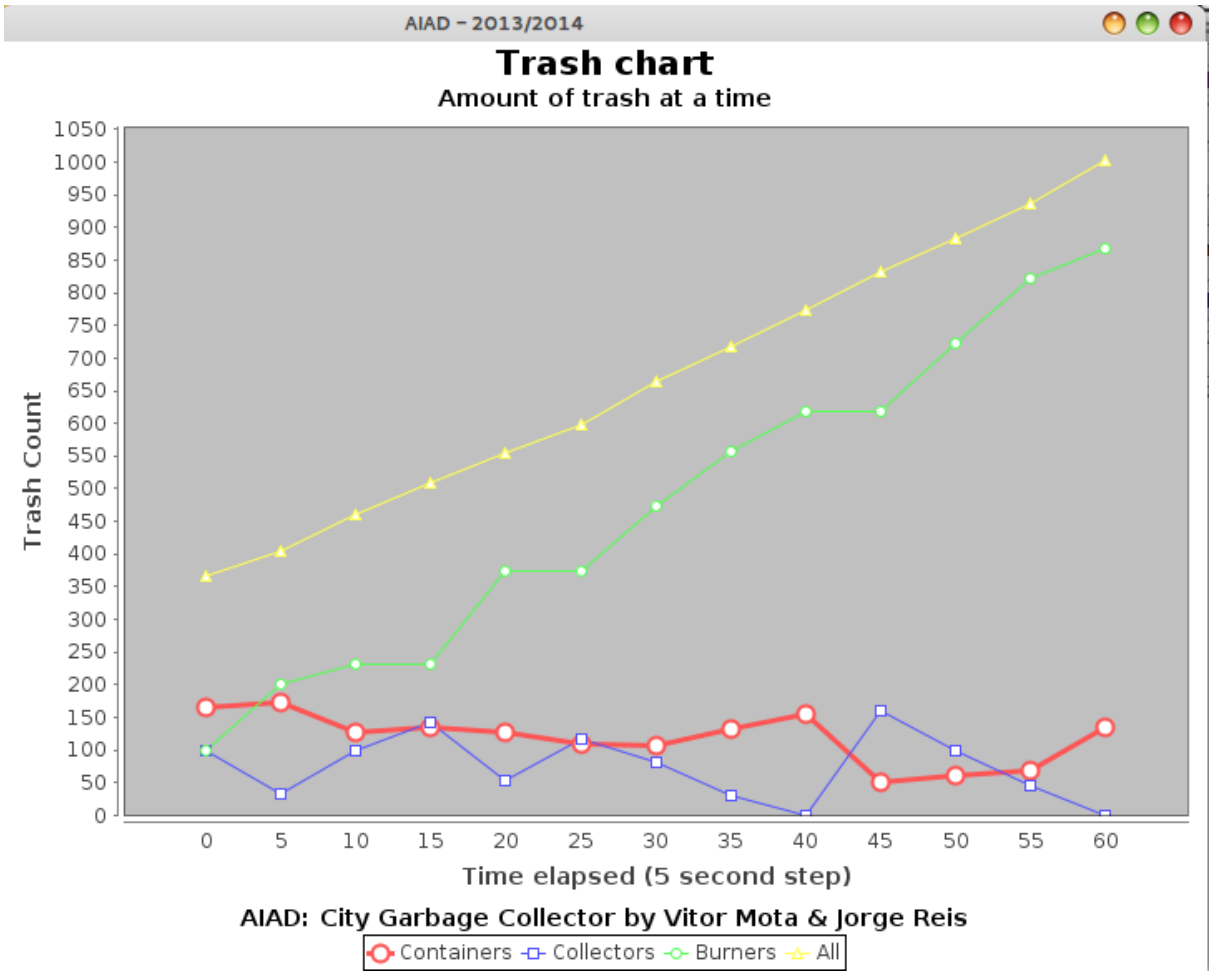
Dica: Pressionando a tecla "S" obtém-se o mesmo efeito que clicando em "Set".

Nota: É da responsabilidade do utilizador garantir que o mapa é ciclico, ou seja não poderão existir estradas sem saída. Para isso sugere-se que antes de modificar o mapa se pause a aplicação usando a check box para o efeito. As setas nas estradas indicam o sentido.

b. Exemplo de mapa válido



3. Interface de estatística
    a. No Jadex escolher o ficheiro ChartsBDI.class e clicar no botão *start* (lado direiro)
    b. Deverá de aparecer a seguinte interface



Nota: Esta é atualizada de 5 em 5 segundos. É possível exportar o gráfico, botão direito do rato → *save as*...