# EFFICIENT CLEANING OF A CITY

## JADE

**MULTI-AGENTS SYSTEM**

Roger Pes Buenestado

Joan Solé Gavaldà

Ferran Mata Arcas

David Perelló Cruz

Francesc Llaó Garcia

UNIVERSITAT ROVIRA I VIRGILI

SMA – 2010-2011

# Index

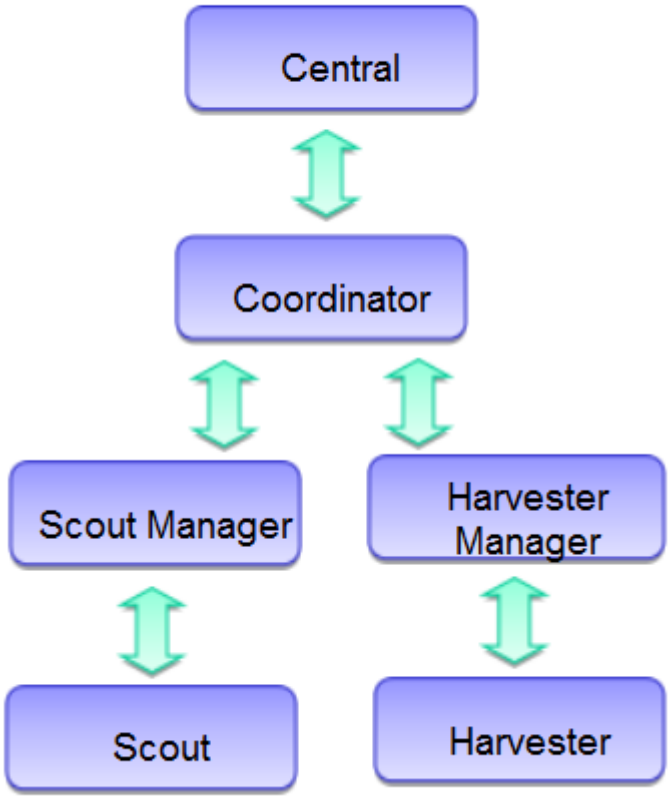
# Coordinator

## Objective

Simulation of a efficient garbage collection system, based on Agent technology

## Structure

Our solution is based on a hierarchical structure:




This structure brings to us a better development organization into 5 different Agent types. Each member will work on one of them (except the coordinator that will work on both the Coordinator and Central Agent).
This structures simplifies each Agent, compared to non hierarchical structure.

# Main Policy

Our bigger efforts in order to provide an efficient and intelligent solution can be resumed as follows:

- Goals, paths and strategy is **computed** again on **every turn**
- Simulation is **protected** against cheating
- We consider **risky movements**, like including on our path an undiscovered zone.

As a result of this:

- Reactions to changes in the dynamic environment are good.
- Our score is true.
- Benefits of risk are high

Some bad consequences:

- High computing requirements
- In dense mazes, it may appear as Agents are constantly changing of goals, due they are avoiding blocked paths.

# Central Agent Roles

The Central Agent has to maintain the game control, security control against unhallowed movement orders, strict turn counting and notifying, discovered map and public data distribution, execute Agent desires and maintain statistical data.

## Turn Control

Based on a ticker behavior, timed as the game parameter “Time to Think”.
Each timeout this agent updates public information, and notyfies that a new turn is about to begin to the coordinator.
It resets also the movement registry of all agents.

## Public Game Information

A copy of the private game information is maintained with each new discovered part of the map.
Each turn this public information is updated and distributed through the Coordinator Agent to all Agents as a serialized object. So each turn they have an updated info, containing the result of their previous actions.

## Movement

Movement ins modelled to an independent package, that includes data definition, message passing, and message rely on intermediate Agents, to provide an unique an easy to implement solution for movement order propagation.

### Movement

Conceptual Data Class, contains an order of type GO, PUT, or GET.
Also contains the Agent that wants to do this action.
And the direction of its objective. That is, where he wants to move, from he wants to get garbage or to where he wants to drop garbage.

### Movement Sender

Implements communication behaviours, ready to use by all Agents. They only have to instantiate the class and call the Send method.

### Movement Rely

A communication class that install a method on Manager Type Agents, that rely all movement orders from Agents above, to the coordinator Agent, and so to the Coordinator Agent to the Central Agent.

## Security

Each time a movement order is received, several checking are necessary to assure our score legacy:

- Agent has not moved yet on this turn
- While Moving
    - Destination is an empty Street
    - Destination is inside the map
    - Direction is UP/DOWN or SIDE/SIDE
- While Getting Garbage
    - Agent can carry that garbage type
    - Agent is not full
    - There is Garbage on the building
    - There is a building to get it from
    - The agent is not carrying garbage of different type
- While dropping garbage
    - Destination is a recycling Center
    - Destination accepts the garbage
    - Agent has enough garbage to drop

## Statistics

In order to deliver a brief of the game results on simulation end. That is, we reached the maximum turns or there is no more garbage to collect and recycle.

- Earned and maximum points
- Collected and total garbage
- Turns needed to finish
- Discovered buildings with garbage
- Total buildings with garbage
- Time to discover all buildings
- Number of moving turns for each Agent

## Results

Or results are fast and points to time balanced. Almost all the mazes are entirely solved. Some of them left incomplete due to harvesters blocking paths when inactive. See our tests results at the end of this document for conclusions:

EASY MAZE

Simulation Finished
**********

Points earned: 65
Max points: 90
Percentage earned: 72.22222222222223%
Garbage collected: 35
Total garbage: 35
Turns to finish: 161
Buildings discovered: 4
Total Garbage Buildings: 4
Percentage of Garbage Buildings discovered :100.0%
Turns to find all garbage buildings: 67
Agent S6: 90 movements
Agent H2: 108 movements
Agent H5: 85 movements
Agent S1: 65 movements
Agent S4: 81 movements
Agent S3: 122 movements

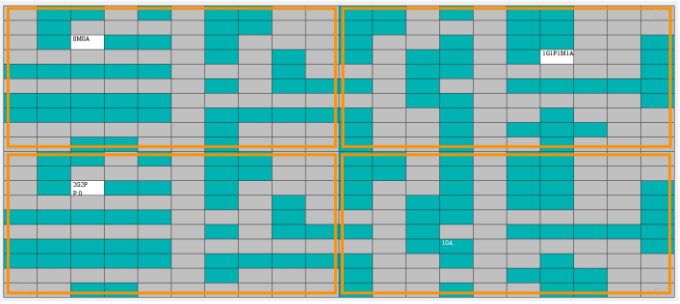
HARD MAZE

Simulation Finished
**********

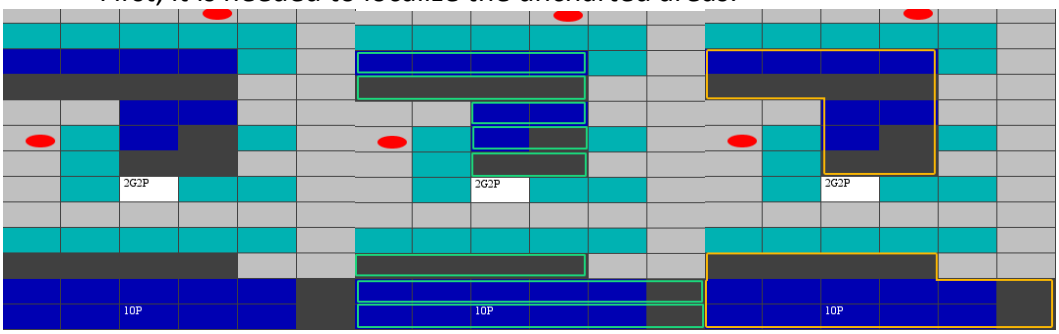
Points earned: 184
Max points: 258
Percentage earned: 71.31782945736434%
Garbage collected: 104
Total garbage: 104
Turns to finish: 365
Buildings discovered: 7
Total Garbage Buildings: 7
Percentage of Garbage Buildings discovered :100.0%
Turns to find all garbage buildings: 116
Agent S5: 146 movements
Agent S4: 133 movements
Agent H1: 188 movements
Agent H3: 237 movements
Agent H6: 217 movements
Agent S2: 122 movements

# Scout Manager & Scouts

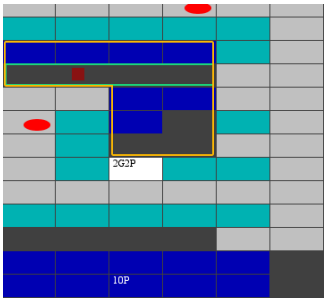
- **Scout Manager:**
  - The purpose of the Scout Manager agent is, as its name indicates, to manage the scouts to discover the map, and doing it in an efficient and fast and manner.
  - Scout Manager uses *ProtocolContractNeInitiator* to prepare the communications with the scouts (scout agents).
  - Steps for each turn:
    - It starts searching and saving the scouts and dividing the map into N quadrants, where N is the number of scouts rounded up to a pair. Each scout is assigned to a quadrant. This steps is done only at the start.



    - The map is again divided into quadrants, but this time is divided into N*2 quadrants, where N is the number of the scouts. Why N*2? It just works, it was determined by doing tests and seems to be a good number for the actual size of the map.
    - For each of those quadrants, it is determined a point of an uncharted area, and this point will be submitted by a contract net. How those points are determined? It is better explained graphically:
      - First, it is needed to localize the uncharted areas:




        1. It is determined, row to row, the rectangles of the quadrant that are uncharted.
        2. Those rectangles are grouped if they are in contact.
      - Second, choose a point from one of these areas:



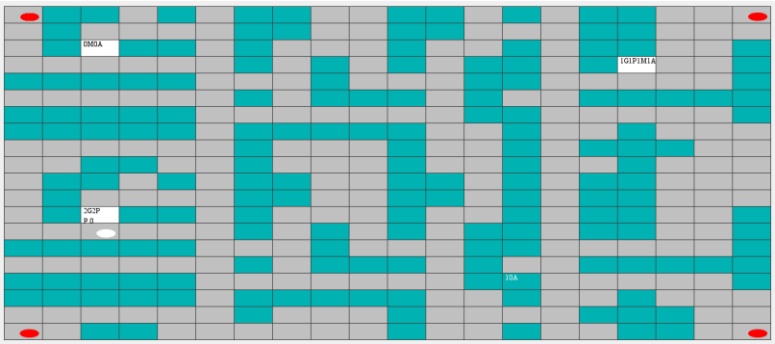

1. The target row will be numRows / 2 of the area.
2. The target column will be the centre of the target row.

- Notice that this method is not strictly exact, but it is almost and it works. Besides, the next turn, the point will move again to a more accurate point of the same area, and at the end, the area is always entirely discovered efficiently.
- Center of gravity is not used here because it can end up returning a value that is already discovered.
- To resolve this algorithm, the classes Rectangle and Point are used. They are simple and self-explanatory.

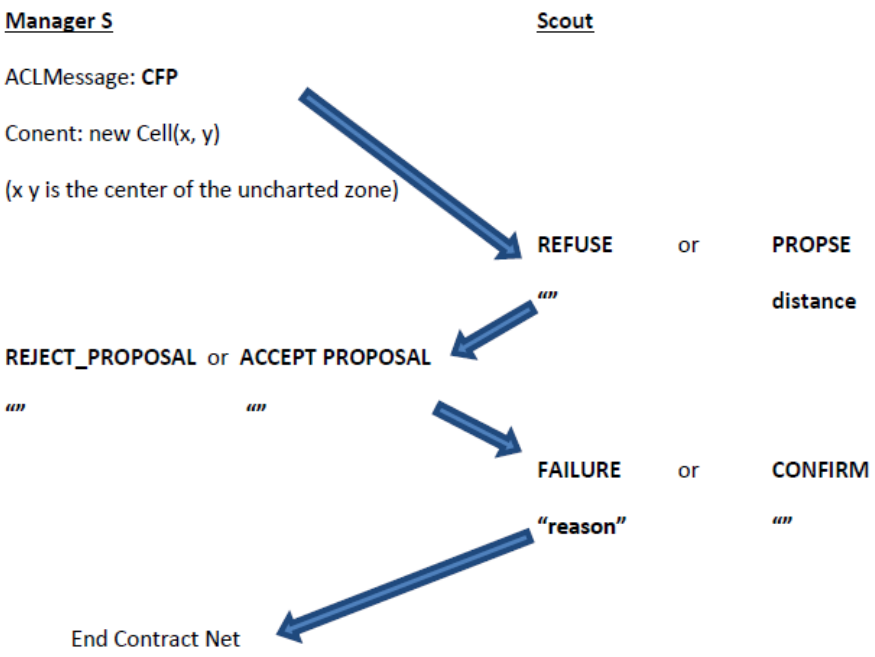
■Let's see what happens when the contract net has responses:

- Let pass all the responses that are proposals and discard the refuses.
- Check if the scout is proposing a distance to a point that is in a quadrant assigned to it.
- If various scouts are assigned to the same quadrant, accept the one with the shortest distance/path to the target cell.

- Unusual strategies:
  - When just one quadrant is entirely discovered, all quadrants become merged. At this point, the contract net has full sense as well all scouts can go to any point of the map.
  - When the map is completely discovered, the scouts are sent to the corners so they can't interfere with the harvesters.



  - If no scouts can move, due, for instance, to a conflict with a harvester, they are sent to the corners.



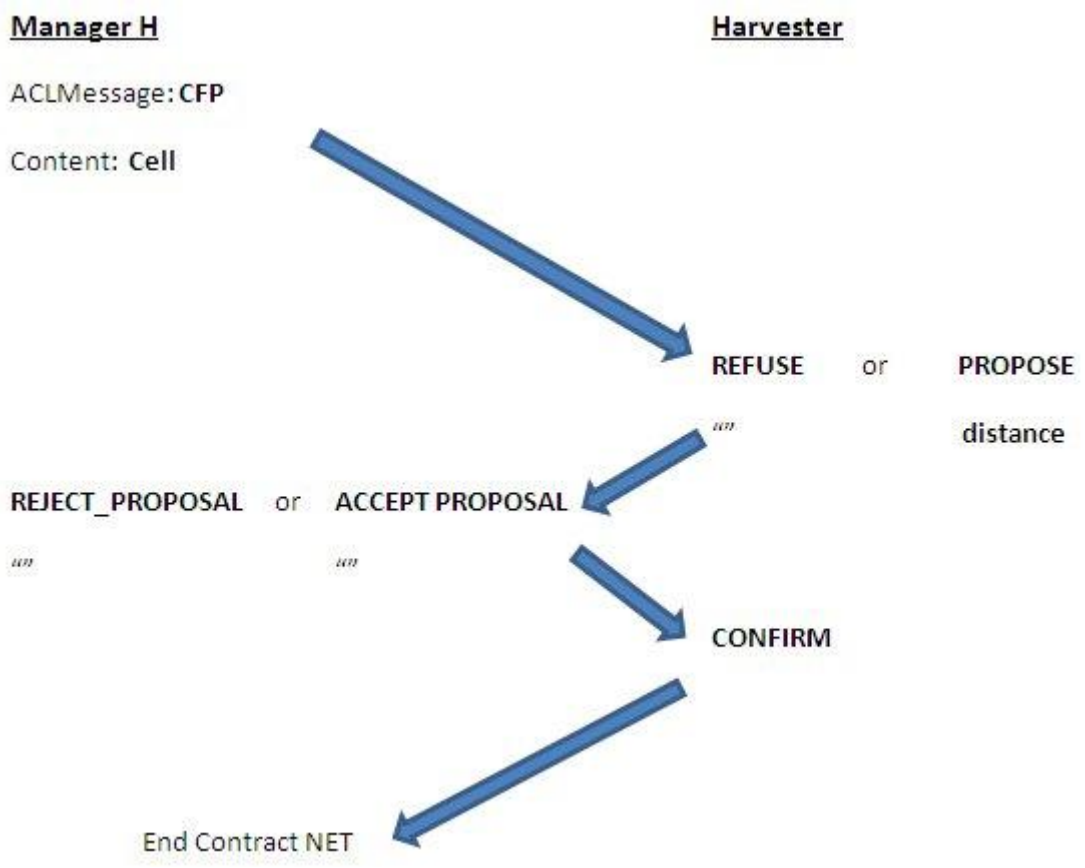

- ScoutManagerUtils: This class provides a series of utilities that can be used by the ScoutManagerAgent and the ProtocolContractNetInitiator:
  - ●divideCity: Divides the map into x quadrants, where x is the number of scouts (+1 if that number is odd)
  - ●chooseUnchartedPointToSendScout: Taking a quadrant and a scout, returns the nearest uncharted point to the scout. Not used.
  - ●chooseUnchartedPointInAQuadrant: It will return a point in the biggest uncharted area of the quadrant or the centre of the area nearest to the last point.
  - ●joinQuadrants: If just one quadrant is completely discovered, it will join all the quadrants. And so there will be just one quadrants and its dimensions will be the same as the map's.
  - ●mapNotEntirelyDiscoveredYet: Returns true if the map is not entirely discovered yet.
  - ●chooseCornerPoint: It will return a point in the corner of the map to send a scout.

- **Scouts:**
  - They explore the map. These vehicles will move through the city discovering the buildings that have garbage. They cannot harvest garbage. They can move, horizontally or vertically, 1 cell per turn.

  - Scouts uses *ProtocolContractNetResponder* to analize the communications with their manager (ScoutManager).
    - Scouts receives undiscovered cells and they calculates the shortest path between themselves position and this cell. Then scouts brings the number of steps of this path. Once scouts brings them distances, scout manager accepts the proposals and if scout manager accept proposal of one scout, this scout brings the next movement of them path.

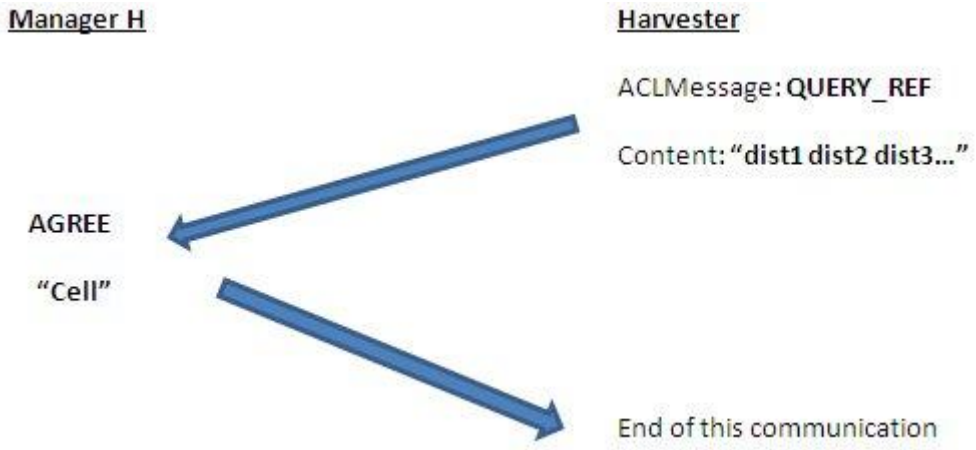
# Harvester Manager & Harvesters

- **Harvester Manager:**
  - That agent have the next functions:
    - Bridge between Coordinator and Harvester: Coordinator indicates to other agents when is the new turn, and that information passes for Harvester Manager to Harvesters. Be the same with movements orders.
    - Detect garbage in buildings: Harvester Manager search in discovered map which cell have garbage.
    - Choose the best Harvester for recollect garbage: Assign garbage to the nearest harvester, if he can carry.
    - Choose recycling center for downloading garbage: When Harvester load all garbage or is full, Harvester Manager assign he the best recycling center for drop garbage. Harvester Manager choose recycling center comparing between distance and points.
  - Harvester Manager uses FIPA ContractNet for that communication, and for do it use ProtocolContractNeInitiator and ProtocolContractNetResponder behaviours. Use this communication to prepare the communications with their harvesters. Each turn search garbage and assign it to the Harvesters. That protocol explained in follow scheme:

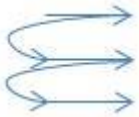



In first step the content have Cell to indicate where is garbage. Distance is integer and is the distance between the Harvester and Cell.

- ReceivingFinishLoad: Harvester informs Harvester Manager that he finishes load garbage or is full For do it Harvester send FinishLoad to Harvester Manager, and Manager deciding the cell of recycling center for download garbage. FIPA Query is used for that communication, but this communication is blocking because Harvester need know where he download garbage. SetProtocol is mark with "Query" for all messages. Protocol explained in follow scheme:

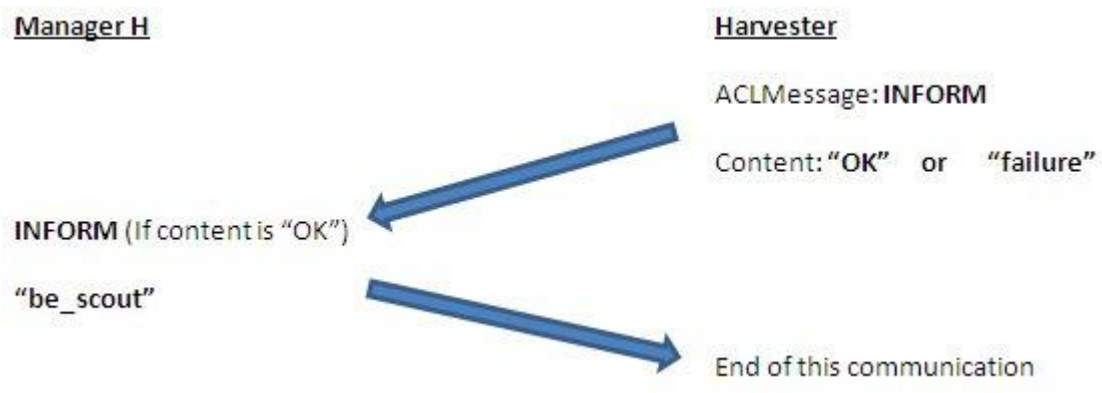


In first step Content have the distance of the recycling buildings ordered from left to right and up to down.



- DistanceList: Use object DistanceList for put all the distance between Harvester and recycling center. DistanceList is List of Integers.

- ReceiveFinishDownload: Harvester informs Manager that he finishes download garbage. That communication send to Harvester "Be_scout", but that harvester wait for other ContractNet. SetProtocol is mark with "Finish_work" for all messages. Follow scheme explain this communication:



- **Harvesters:**
  - They harvest garbage and bring it to a recycling center. They can move, horizontally or vertically, 1 cell per turn. In order to harvest garbage, they must be situated in a cell adjacent to the building containing garbage (horizontally, vertically or diagonally) and remain there for some time (1 turn per garbage unit).

  - Harvesters uses also the *ProtocolContractNetResponder* to communicate with their manager (HarvesterManager). Each harvester receives several requests to know the distance between us and the garbage of a point and harvesters return this distance if possible. First, harvesters only accept one request and after control if is possible calculate the distance. Harvesters control if possible catch the type of garbage at a point and if in this moment harvesters are transport garbage, control that this garbage are the same type which are the harvesters. The others case, Harvesters refuse the request to go catch garbage.

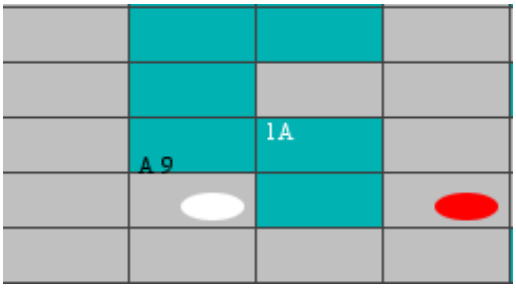
  - When a ManagerHarvester accept a proposal by the harvesters, the harvesters are directed to where to collect garbage, or, if the garbage is at its perimeter they pick it up

  - For each turn, the harvester control if is full and then they send a list of distances to points of recycling to know where go to recycling. Also control if harvesters catch all garbage of a cell but they hasn't full, then they communicate this to know where go to recycling garbage. Control if harvester is going to recycling garbage if they have download or only move. For this last point, the harvesters, look at your perimeter to know if possible download garbage, else they move.

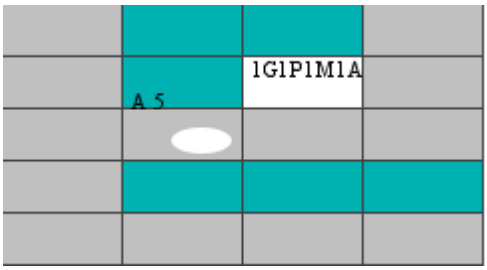
  - Finally, Harvesters use two other protocols to communicate with HarvesterManager:
    - SendFinishLoad: This is to announce that it has finished picking up garbage at a specified point and propose different distance, for items of recycling, to inform us where to go to Recycle

    - SendFinishDownload: This is to announce that it has completed recycling the garbage at a specified point.

- **Examples**




Harvester loading garbage in diagonal




Harvester downloading garbage in diagonal

# A* Path finding

We have implemented the A* algorithm to find the optimal path of two points. For each turn harvesters and scouts copy the map of the game and they works with this copy. This new map works as a matrix, not with cells as original map.

It has been created a simply method to get the path between two points, just add the 2 points it returns us distance between them. If path is null, hence, no path between two points, A* start returns 10000 steps and if one coordinator obtains 10000 of another agent, it rejects this proposal immediately.

Scouts and Harvesters try two types of path and gives the shortest path.

- **First path**:  Agent creates a path using only discovered cells.
- **Second path**: Agent creates a path between two points and it can moves over undiscovered cells.
    Weight of discovered cells are 1.
    Weight of undiscovered cells are 2.

- Finally, agents give the shortest path of these two alternatives.

Noteworthy second path is very important with two fundamental reasons:

- Scouts can discover dark zones faster crossing the dark zones (if one cell was building… bad luck).
- When one scout discovers garbage, harvesters can go to this point and it assumes scout behavior, harvester can cross over undiscovered zones.

# Tests Conclusions

**Some tests to evaluate the correct functionality.**

- Number of scouts and harvesters on a easy maze city:
  - Under 2 agents of each type, there is a significant impact on performance
  - Over 3 agents of each type, there is no more increasing performance
  - More than 8 Agents on a 20x20 city, provokes a lot of blocks
- City topology and size
  - Open topologies are always solved (square cities)
  - Easy Mazes are almost completely solved with good score
  - Extremely complex mazes are solved at almost the turn limit or not solved
- Distribution of the garbage
  - Dense garbage zones provoke a lot of blocking paths
  - Dispersed garbage are easy to handle
- Number of recycling centers
  - Recycling centers are crucial entities as the less they are, the more blocking problems arise
  - On well distributed cities, with high accessibility to recycling centers, the easiest the solution.

# Improvements

- Implement an unblock strategy for inactive Agents (done with scouts, todo with harvesters)
- Improve message performance, bypassing the hierarchical structure. With such turn propagation or movement rely
- Inactive harvesters being like scouts when inactive
- Multiple harvesters going to a single garage point at a time

# References

Source code SVN:
https://ia2-jade.googlecode.com/svn
Project Activity:
http://code.google.com/p/ia2-jade/updates/list
Wiki Documents:
http://code.google.com/p/ia2-jade/w/list
Downloads:
http://code.google.com/p/ia2-jade/downloads/list
Public Documents:
SlideShow:
https://docs.google.com/present/edit?id=0AXii9lX2DEWHZGcyeDliMzlfMjE0dmg5dHozag&hl=en_US
FinalDoc:
https://docs.google.com/document/d/1nsNf538AU2-bsVNZgfRyAO7_69IF4dwpKZufMNYvcg4/edit?hl=en_US