

Multi-Agent Programming Contest Scenario Description (2015 Edition)

<http://www.multiagentcontest.org/>

Tobias Ahlbrecht Jürgen Dix Federico Schlesinger

May 18, 2015

New in 2015: Since the Scenario for 2015 is brand new, we recommend to read the complete documentation.

Contents

1	Introduction	2
2	Visualization	2
3	Teams & Vehicles	4
4	Items and Facilities	5
5	Money and Jobs	5
6	Agent Actions	6
6.1	Actions' Parameters	8
6.2	Actions' Failure Codes	10
7	Percepts	12

1 Introduction

In the following, we provide a detailed description of the Multi-Agent Programming Contest 2015 scenario.

Our scenario consists of two teams of agents moving through the streets of a realistic city. The goal for each team is to earn as much money as possible. Money is rewarded for completing certain jobs. Jobs comprise the acquisition, assembling, and transportation of goods. These jobs can be created by either the system (environment) or one of the agent teams. There are two kind of jobs: priced and auctioned. A team can accept an auctioned job by bidding on it. The bid amount of money is the reward. If both teams bid, naturally the lowest bid wins. If a job is not completed in time, the corresponding team is fined. Priced jobs have a reward defined upfront, that is given to the first team to complete that job. The teams have to decide which jobs to complete and how to do that, i.e. where to get the resources and how to navigate the map considering targets like shops, warehouses, charging stations, storage facilities.

A team consists of different types of agents. The agents differ in their speed, how they move around the city, battery charge, the volume of goods they can carry, and which tools they can employ to craft other goods. Currently we have 4 types: cars, trucks, motorcycles, and drones.

Goods can be bought, crafted, given to a teammate, stored, delivered as part of a job completion, recovered from a storage facility, and dumped. These action may happen at their respective specific locations/facilities. The crafting of an item requires the use of other goods, some which serve as prime matter and some which serve as tools. Since each kind of agent can only handle a subset of the tools, the crafting of some goods require the explicit collaboration of 2 or more agents.

Agents possess a battery charge that gets decreased as they move around from one place to another. They need to make sure they never run out of charge, and therefore should plan their visits to the charging stations accordingly. Moving from one place to another, as well as recharging the battery at a charging station, are actions that are carried on only partially on each step, and may require several steps for completion.

Tournament points are distributed according to the amount of money a team owns at the end of the simulation. To get the most points, a team has to beat the other, as well as surpass a certain threshold. If a team has a large debt, points are deducted.

Figure 1 shows an overview of a map with different kinds of agents and facilities.

2 Visualization

We provide a visualization tool where all relevant information of what goes on during a simulation is easily accessible. The visualization tool is provided in two variants: the first can connect to *MASSim* via RMI, to display simulation infor-

3 Teams & Vehicles

We currently define four roles (see Table 1), where each role describes the routes that the vehicle is allowed to take, its speed, its battery capacity, its load capacity, and the tools it is capable of using to manufacture new items. The teams consist of 16 agents (4 Cars, 4 Drones, 4 Motorcycles and 4 trucks).

Car	Speed:	3
	Routes:	roads
	Battery capacity:	500
	Load capacity:	550
	Tools:	tool1, tool2
Drone	Speed:	5
	Routes:	air (moves in straight line from one place to another).
	Battery capacity:	250
	Load capacity:	100
	Tools:	tool1
Motorcycle	Speed:	4
	Routes:	roads
	Battery capacity:	350
	Load capacity:	300
	Tools:	tool1, tool3
Truck	Speed:	1
	Routes:	roads
	Battery capacity:	3000
	Load capacity:	1000
	Tools:	tool2, tool3

Table 1: The different roles.

4 Items and Facilities

Items, *a.k.a. products*, are the elements that the agents can acquire, manipulate, and move around. Items possess an id and a volume (which effectively limits the amount of items that a single agent can carry, or that may be stored at a storage facility). Some items may be manufactured by the agents, using other items as prime matter and as tools. Which items serve as prime matter and/or tools depend on the particular item being manufactured (i.e., it is possible for an item to serve as prime matter in some cases, and as a tool in others).

There are many facilities distributed along the map. Each belongs to one of five kinds: **shop**, **storage**, **charging station**, **workshop** and **dump location**. Agents can perform different actions at each of these, more details are given below. Independently of its type, all facilities have a unique id and an associated location (defined by its longitude and latitude).

Shops offer different items for agents to buy. The item selection, current stock, and prices, may vary from one shop to another. Charging stations have a limited number of charging slots (when these are in use, other incoming agents are placed in a queue). Number of slots, price, and speed of charging may vary from station to station. Storage facilities offer agents the possibility to store items temporarily, either to free their load capacity and/or to let teammates pick them up at a later step. Total capacity and price may vary from storage to storage. Dump locations let agents get rid of items that they do not need anymore, to free their load capacity. Workshops are used for the manufacturing of items.

5 Money and Jobs

The goal of the game is to earn during the simulation the biggest possible amount of money (and more than the opponents). The system keeps track of the balance: agents are always allowed to "pay" for all priced actions (independently of their current amount of money) possibly generating a negative balance. Details on the cost of the actions are given below.

The way to earn money is through jobs. There are two kinds of jobs: *auction* and *priced*. Both kinds of jobs are completed by delivering all the items specified by the job at a particular storage location, also specified by the job. Delivery of items can be carried on in parts and by different agents, as long as the job remains active. However, no partial rewards are given for partial deliveries.

Auction jobs are started with an auction period, during which teams may place a bid to be assigned the job. The team that has placed the lower bid is assigned the job at the end of the auction period. The winning bid, however, must be lower than a certain threshold (defined by the job but not disclosed to the teams). If the job is assigned, the team that won the bid has a certain number of steps to complete the job and earn the bided price. Otherwise, the fine that the job specifies is taken from that team's balance.

Priced jobs, on the other hand, have a fixed price that is paid to the first

team to complete it. There is also a limited number of simulation steps for the completion of the job. Partial deliveries by each team are counted separately. If the job is not completed by any of the teams in the assigned time, it is simply cancelled.

Many Jobs are posted by the system throughout the simulation. Teams are also allowed to post jobs: these look indistinguishable from system-created jobs to the other team, so a team may collaborate with the other, without being aware of it. Of course in this case the posting team pays the price upon completion and gets the delivered items. It also gets the fined amount if this is an auction job that the other team bid for and didn't complete (this is an indirect way in which a team may earn money).

Items that were delivered for a job that wasn't completed by that team, may be recovered from the storage location once the job is finished (either by completion from the other team, or timed out).

6 Agent Actions

In this section we present all the actions that are available for the agents. Availability of an action for a particular agent depends on that agent's location and context. Table 2 presents a condensed version of the general characteristics. Following, we present an explanation of every action.

goto Move the agent from one location to another. The location can be specified either explicitly (by giving the latitude and longitude), or referencing the id of a facility. Depending on the distance to the destination and the speed of the agent, it is likely that the agent needs several simulation steps to complete the route; in this case it is enough (and preferred) for the agent to specify the destination just the first time: successive execution of this action will simply continue advancing on the pre-established route.

buy Only possible at a **shop**. The agent buys the specified number of units of the specified item id. The availability of the item and price paid for each unit is dependent on the particular shop. The buying agent must have enough free capacity to receive the items it intends to buy.

give Give to a teammate agent the specified number of units of the specified item id. This action can be performed at any place in the map, but both the giving and the receiving agents must be on the same spot. This action requires explicit coordination: the receiving agent must execute the **receive** action at the same time in order to succeed.

receive Receive items from teammate agents, as long as the agent has enough free capacity. See **give**.

store Only possible at a **storage** location. Store a number of units of the specified item, paying a cost. Items can be retrieved later by any agent of

the same team. Storage capacity at each facility is limited, so enough free capacity at the given **storage** location is required.

retrieve Only possible at a **storage** location. Retrieve a number of number of units of the specified item, previously stored by the team at this **storage** location.

retrieve_delivered Only possible at a **storage** location. Retrieve a number of number of units of the specified item, previously delivered as part of a job competition. If the job was completed and the reward was paid, the poster team of the job may use this action to get hold of the items that the other team delivered. If a job was cancelled or was completed by the other team, an agent can use this action to recover items that were delivered towards a partial (i.e. unsuccessful) completion of that job.

dump Only possible at a **dump** location. The agent gets rid of a number of number of units of the specified item, in order to increase its free capacity. Depending on the particular **dump** location, a cost might be associated with this action.

assemble Only possible at a **workshop**. The agent manufactures a single unit of the specified item, and gets hold of it. The agent may be assisted by teammates, that can contribute prime matters and tools operation.

assist_assemble Only possible at a **workshop**. The agent supports a teammate in the manufacturing of an item. Prime matter may be taken from it automatically as required by the product being manufactured. The agent may also contribute with the operation of tools it is able to operate (as defined by its role), in this case it is required that the agent is actually holding such tool.

deliver_job Deliver items required to complete the specified job. Each job has a **storage** location associated, and the delivery of items is only accepted at such location. Partial order deliveries are accepted; they can be useful to free carrying capacity of an agent during the process of completing a job.

charge Only possible at a **charging station**. The agent enters the charging station. If there is an available slot, the agent starts charging immediately, otherwise it is placed in a queue and automatically moved to an available slot at a later step, when it becomes available. Even when the agent is already at a charging slot, this action may require many steps until a full charge is achieved; if an action other than **charge** or **continue** is executed, the agent loses its place in the charging-slot / queue, and may be forced to wait before being able to continue charge. Once a full charge is available, the agent is removed from the slot (i.e. it cannot block a slot indefinitely). The cost and charging are dependent on the particular station.

bid_for_job Place a bid for the specified *auction job*, that has to be in the action stage.

post_job Create a new job offering, that the other team may choose to complete in order to get the reward. This is another way to get hold of items besides buying at shops or manufacturing.

continue The agent continues the execution of an ongoing action (**goto** or **charge**).

skip The agent doesn't perform any action, or continues execution of an ongoing action.

abort The agent cancels the execution of any ongoing action.

The result of an action is perceived explicitly by the agent, i.e., the information is sent to it in the next percept. Last action result is **successful** when the action is effectively executed; for more information on failure of actions see section ??.

6.1 Actions' Parameters

Some actions require the use of parameters parameters always take the form **key=value** and are separated from each other by a single space. Table 2 shows the parameters related to each action.

For most actions parameters are self-explanatory. The **post_job** action is more complex, so its parameters are explained in detail:

type Either **auction** or **priced**, defines the type of job to post.

max_price For auction jobs, the maximum bid price that will be accepted (i.e., if all bids are higher than this value, the job will be considered cancelled at the end of the bidding period).

fine For auction jobs, the fine that the team that wins the auction will have to pay if the job is not completed in time.

price For priced jobs, the amount of money that will be paid upon completion.

active_steps The number of steps that the job will remain active an accepting delivery of items for its completion. (for auction jobs, the active steps start counting after the bidding period).

auction_steps For auction jobs, the number of steps for the bidding period.

storage The id of the storage were items shall be delivered.

itemX and textttamountX (Where X is a number) this is the format used to specify the list of required items (with item id and number of items of that id). Numbers have to be consecutive, starting from 1. The ordering is not important, but all item ids used shall be distinct.

goto	Parameters: (optional if agent is already en-route) - option 1: facility=fac.id - option 2: lat=51.805 lon=10.3355 Current location: Any Cost: Battery charge
buy	Parameters: item=item.id amount=10 Current location: Shop Cost: Money according to item price at that shop (and amount of items bought).
give	Parameters: agent=id1 item=item.id amount=10 Current location: Any (receiver agent at same location) Cost: -
receive	Parameters: - Current location: Any (giver agent at same location) Cost: -
store	Parameters: item=item.id amount=10 Current location: Storage Cost: Money according to storage location's price
retrieve	Parameters: item=item.id amount=10 Current location: Storage Cost: -
retrieve.delivered	Parameters: item=item.id amount=10 Current location: Storage Cost: -
dump	Parameters: item=item.id amount=10 Current location: Dump location Cost: Money according to dump location's price
assemble	Parameters: item=item.id Current location: Workshop Cost: Items needed to build the product will be taken from the agent
assist.assemble	Parameters: assembler=agentId1 Current location: Workshop (assembler agent at same location) Cost: Some items needed to build the product may be taken from the agent
deliver.job	Parameters: job=job.id Current location: Storage Cost: -
charge	Parameters: - Current location: Charging Station Cost: Money according to charging station's price (if charge was effectively increased)
bid_for.job	Parameters: job=job.id price=100 Current location: Any Cost: (If bid is won and job is not completed, the fine is deduced from the teams' money).
post.job	Parameters: - - option 1: type=auction max_price=110 fine=50 active_steps=30 auction_steps=10 storage=storage.id item1=item.id1 amount1=10 item2=item.id2 amount2=5 ... - option 2: ttype=priced price=110 active_steps=30 storage=id.storage1 item1=item.id1 amount1=10 item2=item.id2 amount2=5 ... Current location: Any Cost: -
continue	Parameters: - Current location: Any (effect varies according to context) Cost: Depending on context
skip	Parameters: - Current location: Any (effect varies according to context) Cost: Depending on context
abort	Parameters: - Current location: - Cost: -

Table 2: The different actions.

6.2 Actions' Failure Codes

Actions can fail due to diverse reasons. The possible causes of failure for each action are presented in Table 2. The perceived last action result will be the prefix **failed_** followed by the failure reason. Here we explain what each possible failure means:

failed_location The agent is not in the right location/facility needed to execute this action.

failed_unknown_item The item id given as parameter does not correspond to an item in the simulation.

failed_unknown_agent The agent id given as parameter does not correspond to an agent in the simulation.

failed_unknown_job The job id given as parameter does not correspond to a job in the simulation.

failed_unknown_facility The facility id given as parameter does not correspond to a facility in the simulation.

failed_item_amount The number of items specified as parameter for this action is bigger than the number of items available.

failed_capacity There is not enough available capacity (for an agent or storage) to perform the action.

failed_wrong_facility The agent is not in a facility that is of a kind different than the one needed to execute this action. (similar to **failed_location**).

failed_tools The agent (or group of agents) does not count with the necessary tools (in the hold of agents that can use them) in order to manufacture the product.

failed_item_type The agent is attempting to manufacture an item that cannot be manufactured by agents.

failed_job_status The status of the job is not the right for the execution of this action (e.g. trying to complete a job that has been already completed or cancelled, bidding for a job after the bidding time has passed, etc.).

failed_job_type The agent is bidding for a job that is not of auction type, or trying to posting a job of an unknown type.

failed_counterpart The counterpart agent failed to correctly execute its part, for an action that requires coordination (e.g. **give**, **assist_assemble**).

failed_wrong_param Some parameter had the wrong format (e.g. numeric parameters)

failed_unknown_error An unknown error was produced during the execution of this action.

failed The action was unknown, so it couldn't be executed.

successful_partial Some items were delivered towards the completion of a job, but the job haven't been fully completed yet.

useless No items were delivered towards the completion of a job.

Besides each action's intrinsic failure possibilities, an uncertainty factor has been introduced that lets actions fail randomly with a 1 percent probability. In this case the action is considered as the **skip** action (and the perceived result will be **failed_random**).

7 Percepts

In every step, the agents get these percepts:

- state of the simulation, i.e. the current step,
- state of the team, i.e. the current money and jobs,
- state of the vehicle, i.e. its internals as described above,
- information about the other vehicles, i.e. the identifier, position, team and role,
- facilities, i.e. general information and specifics if the facility is nearby,
- jobs, i.e. the type, requirements and other specifics.

Please refer to the protocol description for the details about percepts.