Relatório da Atividade dos Agentes Coletores da Disciplina Inteligência Artificial

João Marcello Mendes Moreira

Email: joaomarcello.mm@gmail.com

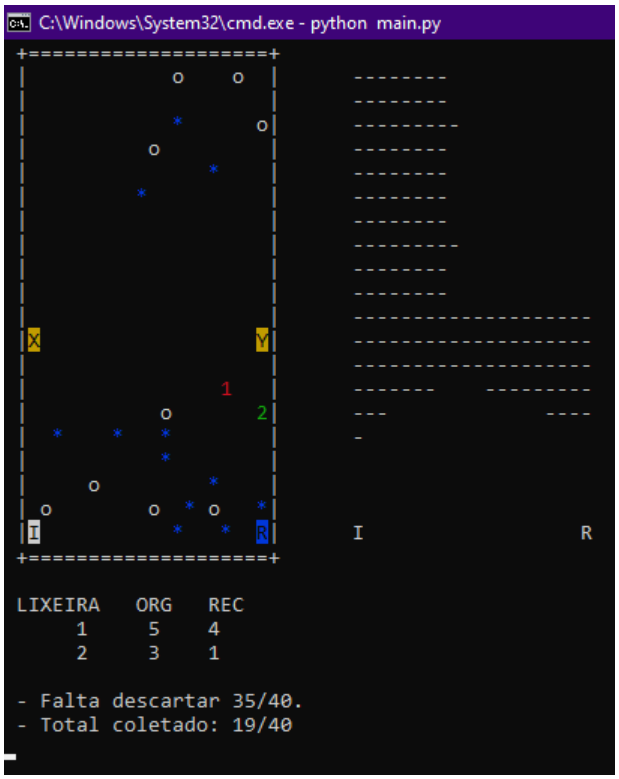
Professor: Tiago Bonini Borchatt

Resumo da Atividade: Simular robôs que coletam lixo. O robô R1 coleta e leva pra lixeira enquanto o robô 2 verifica as lixeiras, pega algum lixo e o leva para o incinerador, se o lixo for orgânico ou para o reciclador caso o lixo seja reciclável.

Implementação

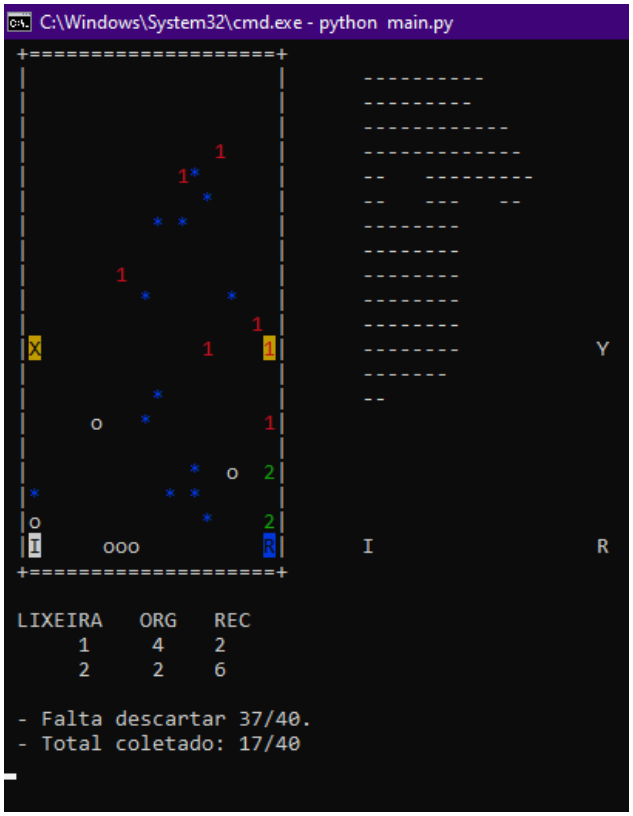
Utilizou-se a linguagem Python para realização da atividade. A Figura 1 mostra a saída do programa. Além do espaço, o programa mostra a quantidade de lixo de cada tipo contido em cada lixeira, assim como a quantidade de lixos coletados e descartados. À direita, é possível observar o mapeamento do agente R1, ou seja, os lugares por onde ele passou.

Figura 1 - Saída do programa. Os pontos 1 e 2 representam os agentes R1 e R2, respectivamente. Observa-se também as lixeiras (X/Y - Amarelo), o incinerador (I - Branco) e o reciclador (R - Azul). Os lixos são representados por 'o' (orgânico) e '*'(reciclável)



O programa foi implementado de modo que seja possível incluir mais de dois agentes como ilustra a Figura 2.

Figura 2 - Saída do programa com vários agentes inclusos no espaço.



Algumas constantes podem ser editadas no arquivo main.py da atividade para mudar a execução do programa. Abaixo, a descrição de cada uma:

- LINES: a quantidade de linhas do espaço. Deve ser um número inteiro positivo.
- COLUMNS: a quantidade de colunas do espaço. Deve ser um número inteiro positivo.
- AMOUNT_TRASH: a quantidade de lixo que deve ser gerado no espaço. Deve ser menor ou igual a LINES*COLUMNS-4.
- SHOW_EXECUTION: booleano que informa se deseja visualizar a execução do programa.
- SHOW_COLOR: booleano que informa se o programa deve reproduzir as cores ou não. Se houver algum problema na visualização das cores, coloque esta variável para False.
- SHOW_FIRST_AGENT_MAPPING: booleano que informa se deve ou não mostrar o mapeamento do primeiro agente inserido no programa. Caso o programa apresente alguma quebra na visualização, coloque esta variável para False.

- DELAY: o tempo de espera em segundos que o programa deve fazer a cada passo. Aumente o valor desta variável para o programa executar mais lentamente.

Segue algumas notas sobre a implementação do programa:

1 - Assumiu-se que os robôs não podem ocupar o mesmo espaço que outro.

2 - Para pegar um lixo, o robô precisa estar no espaço que o lixo se encontra (as vizinhanças não contam).

3 - Para soltar o lixo, o robô também precisa estar no local onde deve soltar o lixo, ou seja, nas lixeiras, no incinerador ou na recicladora.

4 – Como no enunciado da atividade assume-se que os robôs sabem as posições das lixeiras, do incinerador e do reciclador, não foi implementado as arquiteturas Reagente Simples e Modelo para o robô R2 uma vez que ele sempre sabe aonde ir, já que não é necessário mapeamento pra saber as posições dos lugares onde ele recolhe e descarta os lixos.

5 – Assumiu-se que o ambiente envia a informação para os robôs se existe ou não lixo no local. Assim eles sabem quando concluíram suas tarefas. Se não fosse feito isso, os agentes do tipo Reagente Simples não saberiam quando parar, já que não mapeiam o espaço.

6 – O lixo no ambiente é gerado aleatoriamente no espaço. Só é possível especificar a quantidade de lixo que será gerado, mas não o tipo. Assim, o programa pode gerar mais lixo orgânico do que reciclável, ou vice-versa e isso influencia no tempo de execução do programa.

7 – Assumiu-se que o robô R2 não sabe o conteúdo das lixeiras, apesar de saber suas posições. Desse modo, ele sempre tem que ir até a posição das lixeiras pra saber se tem algum lixo nelas.

8 – Para o robô R1:

- Reativo Simples: não mapeia o espaço e faz movimentos aleatórios. Se encontra algum lixo no espaço atual ou vizinhança, pega e leva pra lixeira mais próxima (como ele sempre sabe sua própria posição e a posição das lixeiras, é possível fazer o cálculo mesmo não mapeando o espaço).
- Modelo: faz um mapeamento do espaço a cada movimento guardando as posições onde já esteve e onde tem lixo. Se ele sabe a posição de algum lixo, segue imediatamente para aquele local recolhe-lo (se encontrar algum outro lixo enquanto foi buscar aquele que estava buscando, recolhe este novo lixo

primeiro). Se não sabe onde tem algum lixo, movimenta-se horizontalmente pelo espaço e caso chegue nas bordas, avança para baixo. Se chega no final do espaço, volta pra primeira posição (a posição 0x0) e faz tudo novamente enquanto houver lixo. Se encontra algum lixo, leva pra lixeira mais próxima. Não diferencia entre pegar lixo reciclável ou orgânico, sempre pegando o primeiro que encontrar. Se ele já estiver carregando algum lixo e encontra algum outro no caminho, ele salva sua posição para recolher depois, já que os agentes só podem carregar um lixo por vez.

- Objetivo: tem o intuito de recolher os lixos o mais rápido possível. Faz a mesma movimentação da arquitetura Modelo, mas, além de mapear o espaço e guardar a informação de onde tem lixo, utiliza o mapeamento para evitar percorrer uma linha que já foi completamente verificada. Se encontra algum lixo leva pra lixeira mais próxima e também não prioriza um tipo específico de lixo, pegando sempre o primeiro que encontrar ou o mais próximo caso saiba a posição de algum.
- Utilidade: tem o intuito de recolher os lixos o mais rápido possível, mas priorizando os lixos recicláveis. Movimenta-se como a arquitetura Objetivo. Se sabe onde tem algum lixo, vai até o local recolhê-lo. Contudo, se enquanto estiver indo buscar um lixo ele encontra um outro lixo reciclável no caminho, ele ignora o primeiro que estava buscando e recolhe este novo. Se o novo lixo encontrado for orgânico ele apenas guarda sua posição para buscar quando não souber mais a posição de nenhum lixo reciclável. Porém, se ele estiver indo buscar um lixo orgânico e encontra algum outro lixo orgânico que está mais próximo que o primeiro, ele vai atrás do mais próximo. Se for reciclável, ele ignora o orgânico e vai atrás do reciclável.

9 – Para o robô R2:

- Objetivo: o agente verifica a lixeira mais próxima e se tiver algum lixo, ele calcula qual está mais próximo entre o incinerador e o reciclador e pega o lixo de acordo com o resultado. Se robô estiver mais próximo da recicladora, pega da lixeira um lixo reciclável, caso contrário pega um lixo orgânico. Se a lixeira estiver vazia, ele segue para a outra lixeira e fica nesse loop até todos os lixos terem sido descartados. Se o robô está na lixeira mais próxima do reciclador e não há lixo reciclável nela, ele recolhe lixo orgânico, se houver, e leva para o incinerador. Se

o robô está na lixeira mais próxima do incinerador e não há lixo orgânico nela, ele recolhe lixo reciclável, se houver, e leva para o reciclador.

- Utilidade: ele sempre dará prioridade em pegar um lixo reciclável da lixeira, mesmo que esteja mais perto do incinerador. Se ele está numa lixeira que não tem lixo reciclável, mas tem lixo orgânico, ele recolhe o lixo orgânico e leva para o incinerador, mesmo que a outra lixeira tenha algum lixo reciclável (já que ele não guarda o conteúdo das lixeiras, ele não lembra do conteúdo das lixeiras).

Resultados

Para avaliar a solução do problema, mediu-se o tempo (em segundos) de conclusão do mesmo de duas maneiras: (1) visualizando a execução do programa para checar se tudo corria como o esperado e (2) sem a visualização. As Tabelas 1 e 2 mostram o tempo em segundos para execução do programa utilizando diferentes tipos de agentes para ambos os métodos com o programa, respectivamente. As colunas são os tipos de arquitetura do robô R1 e as linhas a arquitetura do robô R2. Somente um agente de cada tipo foi incluído no espaço em ambos os testes, assim como pedia o enunciado da atividade, mudando apenas o tipo de arquitetura utilizada. O programa foi executado cinco vezes para cada teste e utilizou-se como resultado a média e o desvio padrão dos tempos. Testes foram feitos com o robô R1sendo do tipo Reagente Simples, mas por conta da movimentação aleatória dessa arquitetura, as execuções do programa eram demoradas e por isso não foram incluídas no resultado. A configuração do programa para o método (1) foi:

- Espaço: 20x20
- Quantidade de lixo: 40

A configuração do programa para o método (2) foi:

- Espaço: 100x100
- Quantidade de lixo: 500

Tabela 1 – Média de tempo em segundos da execução do programa visualizando a execução.

R1/R2	Modelo	Objetivo	Utilidade
Objetivo	109,24 ± 13,27	80,042 ± 2,39	75,808 ± 3,46
Utilidade	134,545 ± 18,01	72,548 ± 6,00	73,763 ± 4,83

Tabela 2 – Média de tempo em segundos da execução do programa sem a visualização.

R1/R2	Modelo	Objetivo	Utilidade
Objetivo	18,94 ± 1,57	4,037 ± 0,16	3,997 ± 0,06
Utilidade	19,091 ± 1,40	4,064 ± 0,08	3,807 ± 0,12

Nota-se analisando as Tabelas que quando o robô R1 utiliza a arquitetura Modelo, o tempo de execução do programa aumenta consideravelmente. O robô R1 e o robô R2, ambos com a arquitetura Utilidade, obtiveram o melhor desempenho geral. Mas vale notar que na Tabela 1 a combinação R1/Objetivo e R2/Utilidade obtiveram o melhor tempo, mas possivelmente foi pelo fato dos lixos serem gerados aleatoriamente e isso influencia na execução do programa (como pode ser notado pelo alto valor do desvio padrão nesse teste).