# Proyecto de Simulación

Mauricio L. Perdomo Cortés. C-412

November 30, 2020

Facultad de Matemática y Computación
Universidad de La Habana

# 1 Orden del Problema

**Agentes:**

La orden de nuestro trabajo se encuentra en un pdf que acompaña a este en el repositorio de GitHub.

# 2 Principales Ideas seguidas para la solución del problema.

En el proyecto modelaremos el ambiente mediante la clase Environment, esta representará el estado actual del ambiente, guiará la simulación y reunirá distintas estadísticas. En el ambiente podremos encontrar distintos tipos de elementos entre los que se encuentran Niños, Obstáculos, Suciedad, Corrales y Agentes (Roboces de limpieza). Cada uno de estos elementos está modelado mediante una clase que podemos encontrar en el paquete env. En cada turno nuestro ambiente le indica al robot que realice una acción, luego mueve a los niños y genera suciedad y por último en caso de ser requerido aplica un cambio aleatorio del ambiente, esto consiste en tomar todos los elementos del ambiente y ubicarlos un una posición aleatoria del ambiente. Luego de haber ejecutado todas las acciones del turno, se comprueba si se cumple la condición de parada del ambiente y en consecuencia del resultado se procede al siguiente turno o se termina la simulación (La condición de parada también es chequeada después de la acción del agente para evitar que el robot termine de limpiar todo y sea el momento de aplicar la variación aleatoria lo que no resultaría en la terminación de la simulación).

# 3 Modelos de Agentes:

En el proyecto contamos con la implementación de dos agentes distintos uno implementado siguiendo un patrón reactivo, buscando acercarnos a la Arquitectura de Brooks, y otro que busca un enfoque más proactivo planeando recorridos para cumplir su objetivo.

- El primer agente sigue un enfoque totalmente reactivo, basa su próxima acción en los datos obtenidos del ambiente actual y no tiene en cuenta para nada acciones pasadas o posibilidades futuras. El agente contiene una lista de reglas las cuales están conformadas por una condición y

una acción. Estas reglas estan ordenadas y cada vez que el agente debe realizar un acción va de una en una por estas reglas hasta encontrar la primera que cumpla la condición teniendo en cuenta el estado actual del ambiente, la acción asociada a esta regla es la seleccionada por el agente. En el caso de que no se cumpla ninguna condición el agente decidirá no hacer nada.

- El segundo agente sigue un enfoque más proactivo planeando una estrategia para lograr su objetivo. Este agente tiene como fin dejar a todos los niños en el corral y limpiar toda la casa, para esto tiene dos objetivos fundamentales que cumplir, poner a los niños en el corral y limpiar la casa. Cada determinado tiempo indicado por un parámetro o cuando se cumple determinada condición el agente planea nuevamente su estrategia, al planear decide si se concentrará en limpiar la suciedad o en llevar niños al corral, en caso de que decida llevar niños al corral creará una lista ordenada para decidir a que niño trasladará en cada momento. Como vemos este proceso para decidir qué hacer plantea objetivos que no pueden ser transformados directamente a acciones del tipo traládate a la derecha, suelta a un niño, etc; para decidir la acción como tal a realizar por el agente usaremos el mismo método que con el agente anterior, entonces segun el objetivo que tengamos se construirá un lista de reglas que se ejecutarán como mismo explicamos con el agente anterior.

# 4 Ideas seguidas para la implementación:

## 4.1 Ambiente

El ambiente es modelado mediante la clase Environment, esta se encarga dados los parámetros de inicio de crear el ambiente inicial colocando los elementos en el ambiente, cada elemento está modelado a su vez por una clase que contiene la información e interacciones del elemento. La ejecución de una corrida ocurre mediante el método run del ambiente el cual comezará a ejecutar las acciones necesarias hasta que se cumpla la condición de parada.

### 4.2 Agentes

Los agentes fueron implementados mediante clases que heredan de la clase Agent que representa la interfaz mediante la cual el ambiente interactúa con los agentes. El método más importante de estas clases es el método action que es el que le indica al agente que realice una acción. En el caso de BrooksA el método action toma la percepción del ambiente y comprueba que regla se cumple para realizar la acción; en el caso de ProactiveAgent toma una percepción y primero el agente decide si debe planear o no, y luego de esto decide que acción realizar en base a la percepción y a los objetivos determinados en la fase estratégica.

## 5 Resultados de las simulaciones:

Se tomaron 10 ambientes distintos y se realizaron 60 simulaciones en cada uno (30 por cada agente). De cada par agente ambiente se contó la cantidad de veces que el agente fue despedido, la cantidad de veces que logró limpiar completamente la casa y que porcentaje del ambiente representaba la media de casilla sucias en una corrida. Podemos observar los datos obtenidos en las siguientes tablas.

| Fil. | Col. | T | % de Obs. | % de Suc. | Niños | Despedido | Completado | % Suc. |
|---|---|---|---|---|---|---|---|---|
| 9 | 6 | 74 | 25 | 35 | 4 | 0 | 6 | 32.04 |
| 5 | 9 | 68 | 42 | 13 | 3 | 1 | 27 | 24.46 |
| 8 | 10 | 118 | 16 | 16 | 4 | 0 | 22 | 20.52 |
| 7 | 5 | 33 | 50 | 31 | 3 | 0 | 11 | 29.54 |
| 10 | 5 | 22 | 7 | 36 | 3 | 0 | 0 | 31.38 |
| 6 | 8 | 81 | 7 | 39 | 4 | 0 | 6 | 30.08 |
| 5 | 6 | 48 | 29 | 11 | 4 | 4 | 24 | 23.81 |
| 5 | 7 | 60 | 29 | 17 | 4 | 2 | 27 | 25.22 |
| 6 | 9 | 96 | 22 | 10 | 2 | 0 | 28 | 14.84 |
| 5 | 5 | 100 | 0 | 0 | 4 | 0 | 30 | 14.71 |

Table 1: BrooksA

| Fil. | Col. | T | % de Obs. | % de Suc. | Niños | Despedido | Completado | % Suc. |
|---|---|---|---|---|---|---|---|---|
| 9 | 6 | 74 | 25 | 35 | 4 | 0 | 0 | 37.22 |
| 5 | 9 | 68 | 42 | 13 | 3 | 10 | 20 | 28.59 |
| 8 | 10 | 118 | 16 | 16 | 4 | 18 | 12 | 30.07 |
| 7 | 5 | 33 | 50 | 31 | 3 | 0 | 1 | 40.19 |
| 10 | 5 | 22 | 7 | 36 | 3 | 0 | 0 | 37.97 |
| 6 | 8 | 81 | 7 | 39 | 4 | 0 | 0 | 34.13 |
| 5 | 6 | 48 | 29 | 11 | 4 | 24 | 6 | 34.43 |
| 5 | 7 | 60 | 29 | 17 | 4 | 28 | 2 | 36.73 |
| 6 | 9 | 96 | 22 | 10 | 2 | 0 | 29 | 15.53 |
| 5 | 5 | 100 | 0 | 0 | 4 | 2 | 28 | 19.35 |

Table 2: ProactiveAgent

Luego de observar los datos se llega a la conclusión de que el agente totalmente reactivo es mucho más efectivo en este escenario, lo cual era esperado dado el alto grado de variabilidad que tiene el ambiente teniendo en cuenta que los niños se encuentran en movimiento y que cada cierta cantidad de turnos el ambiente cambia completamente.

# 6 GitHub link al código del proyecto

Código Fuente