

1 Exercici 1

Identifiqueu els agents que hi ha al sistema i definiu el seu tipus. Descriviu les variables i accions que pot desenvolupar cada agent.

En el sistema multi-agent s'han definit 3 agents: `sheep`, `wolf` i `grass patch`.

L'agent **Sheep** i l'agent **Wolf** tenen les següents **variables**:

- `pos`. Vector d'enters. Posició de l'instància de l'agent a la quadrícula per on es mou (grid). Només s'utilitzen la primera i la segona posició, corresponents a les coordenades `x` i `y`.
- `model`. Classe `WolfSheepPredation` que extén de la classe `Model` de la llibreria `mesa`. Conté els paràmetres els que s'inicialitza el sistema. També manté còpies de totes les instàncies dels agents i de la quadrícula a on resideixen (grid).
- `moore`. Variable que defineix el tipus de moviment de l'agent. Si és `True` es podrà moure en totes les 8 direccions; d'altra banda, només es podrà moure cap a dalt, a baix, dreta i esquerra.
- `energy`. Enter. Determina el nivell d'energia d'un agent. Si aquest agent té energia major o igual a 0, tindrà una certa probabilitat de reproduir-se amb èxit. Si la seva energia és negativa, morirà en aquella ronda (step).

L'agent **GrassPatch** té les següents **variables**:

- `pos`. Vector d'enters. Posició del tros d'herba a la quadrícula (grid). Només s'utilitzen la primera i la segona posició, corresponents a les coordenades `x` i `y`.
- `model`. Classe `WolfSheepPredation` que extén de la classe `Model` de la llibreria `mesa`. Conté els paràmetres els que s'inicialitza el sistema. També manté còpies de totes les instàncies dels agents i de la quadrícula a on resideixen (grid).
- `fully_grown`. Booleà. Determina si el tros d'herba ha crescut al màxim o encara no. Les ovelles només poden menjar aquell tros d'herba que ja ha crescut al nivell màxim.
- `countdown`. Enter. Temps restant per a que el tros d'herba torni a créixer al nivell màxim.

A continuació, definirem les accions que cada agent pot desenvolupar. Comencem per les **accions** de **Sheep**:

- Mou l'ovella a una cel·la adjacent. Es crida a la funció `random_move()` que té en compte el valor de la variable `moore` per saber si els moviments en diagonal estan inclosos.

- L'ovella es menja l'herba de la cel·la on es troba si ha crescut al nivell màxim (l'ovella es suma l'energia corresponent al guany de menjar-se un tros d'herba) i, a continuació, marca la instància de l'agent `GrassPatch` com a rasurat (li posa la seva variable `fully_grown` com a `False`).
- L'ovella es mor si no té suficient energia (`energy` és menor que 0).
- L'ovella es reproduïx amb una certa probabilitat d'èxit si té un nivell mínim d'energia (`energy` és major o igual a 0).

Accions de **Wolf**:

- Mou el llop a una cel·la adjacent. Es crida a la funció `random_move()` que té en compte el valor de la variable `moore` per saber si els moviments en diagonal estan inclosos.
- El llop es menja, si n'hi ha, una ovella de la cel·la on es troba en aquell moment (el llop es suma l'energia corresponent al guany de menjar-se una ovella) i, a continuació, la mata (elimina la instància de l'agent `Sheep` de la quadrícula i del model).
- El llop es mor si no té suficient energia (`energy` és menor que 0).
- El llop es reproduïx amb una certa probabilitat d'èxit si té un nivell mínim d'energia (`energy` és major o igual a 0).

Accions de **GrassPatch**:

- Fa créixer l'herba si no està al nivell màxim (`fully_grown` és `False`) i si s'ha exhaurit el temps d'espera necessari per a que creixi (`countdown` és menor o igual a 0). És a dir, si es compleixen ambdues condicions, es marca el tros d'herba amb el nivell màxim de creixement i llest per a ser menjat per una ovella (posa `fully_grown` a `True`). Altrament, redueix el temps de espera (redueix `countdown` en una unitat).

2 Exercici 2

Executeu una simulació del sistema analitzar l'impacte dels paràmetres del model en les poblacions dels agents. Modifiqueu els agents `sheep` i `Wolf` afegint la variable `edat` que s'incrementarà en cada iteració (ex. cada iteració / step pot equivaler a 1 mes). Els agents podran reproduir-se si tenen més d'1 any i poden arribar a una edat màxima de 20 anys. Executeu la nova simulació obtenint la gràfica de l'edat mitjana de cada tipus d'agent. (Ajuda: veure `DataCollector` a l'arxiu `model` i la funció `get_breed_count()`).

Primer, he executat bastantes vegades la simulació amb els paràmetres originals (l'anomenaré cas 1):

Cas 1 (sense modificar del codi):

Cas 1	
Variable	Valor
Grass enabled	True
Grass Regrowth Time	20
Initial Sheep Population	100
Sheep Reproduction Rate	0.04
Initial Wolf Population	50
Wolf Reproduction Rate	0.05
Wolf Gain From Food Rate	20
Sheep Gain From Food	4

I he pogut observar que dos tipus de situacions es repetien constantment:

- **Situació 1:** Al principi, el nombre d'ovelles decau molt ràpidament (degut a que els llops se les mengen i, en algun cas, moren per fam) i el nombre de llops augmenta. A continuació, els llops es moren ja que no aconsegueixen trobar les ovelles restants. Finalment, com les ovelles restants tenen suficient menjar per a sobreviure, es reproduïxen i aconsegueixen establir la seva població al voltant dels 50 individus.
- **Situació 2:** Al principi, el nombre d'ovelles decau molt ràpidament (degut a que els llops se les mengen i, en algun cas, moren per fam) i el nombre de llops augmenta. A continuació, els llops aconsegueixen aniquilar les ovelles, però, finalment, moren de fam ja que no hi ha ovelles restants. Finalment, tota l'herba creix al màxim però no tenim agents del tipus Sheep ni Wolf.

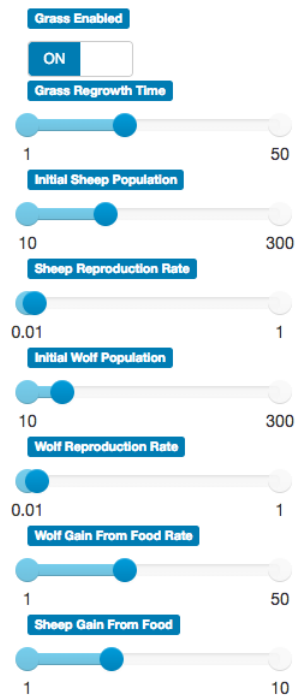


Figura 1: Cas 1 - Valors de les variables.

Les gràfiques obtingudes per a una simulació en concret de les situacions anteriors són les següents:

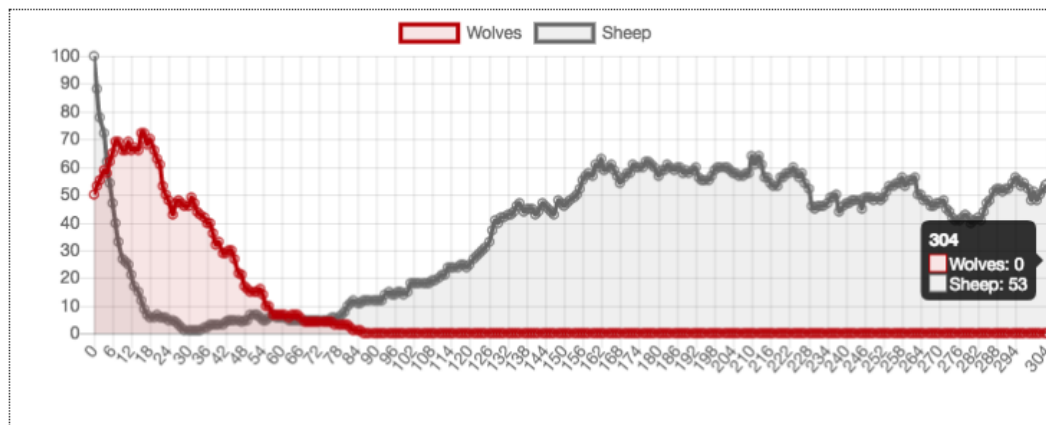


Figura 2: Cas 1 - Situació 1. Gràfic de l'evolució de les ovelles i els llops.

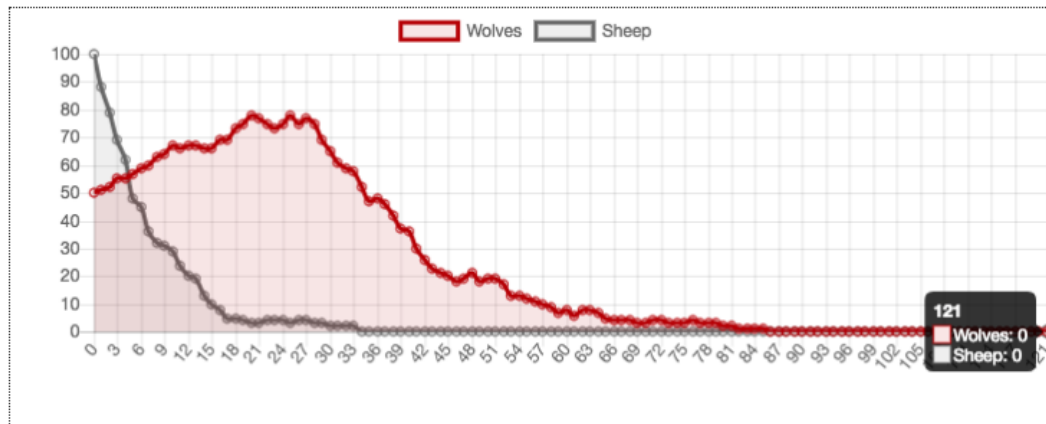


Figura 3: Cas 1 - Situació 2. Gràfic de l'evolució de les ovelles i els llops.

Cas 2 (sense modificar del codi):

Jugant amb els paràmetres he pogut comprovar que només modificant la tasa de reproducció de les ovelles, s'aconsegueix arribar a una situació bastant més estable.

De totes maneres, les fluctuacions del nombre d'ovelles i llops són evidents i bastant brusques. Cal dir, també, que en alguna de les múltiples execucions de la simulació d'aquest cas, he observat que s'ha arribat a la situació d'aniquilar totalment les ovelles.

Cas 2	
Variable	Valor
Grass enabled	True
Grass Regrowth Time	20
Initial Sheep Population	100
Sheep Reproduction Rate	0.23
Initial Wolf Population	50
Wolf Reproduction Rate	0.05
Wolf Gain From Food Rate	20
Sheep Gain From Food	4

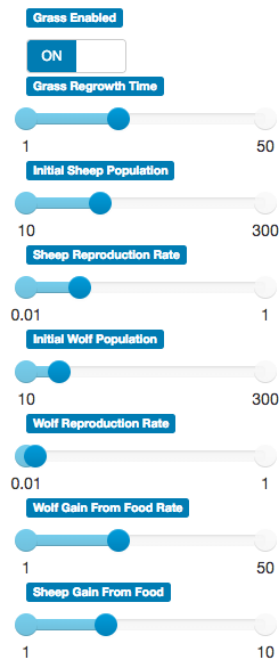


Figura 4: Cas 2 - Valors de les variables.

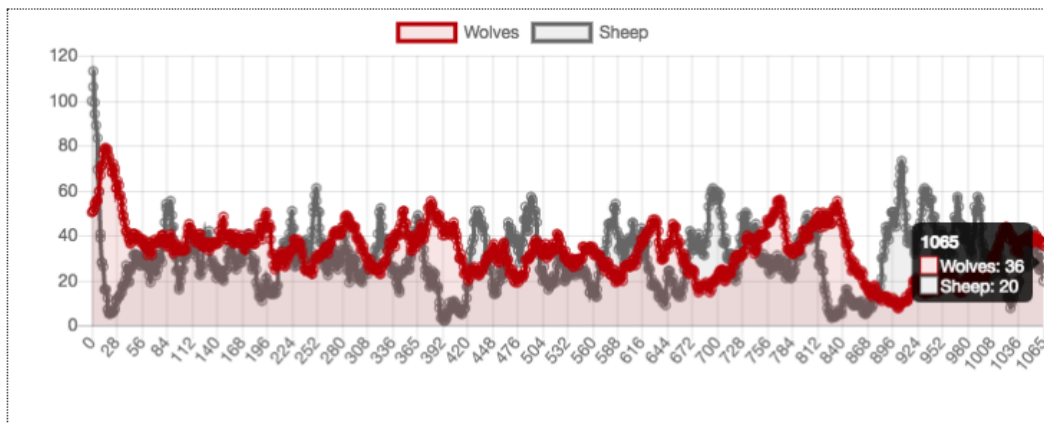


Figura 5: Cas 2 - Gràfic de l'evolució de les ovelles i els llops.

Cas 3 (sense modificar del codi):

Llavors, he modificat pràcticament el valor de totes les variables, buscant que pogués mantenir-se estable, dintre d'uns marges de variabilitat. I he trobat que la següent combinació dóna uns resultats bastant interessants:

Cas 3	
Variable	Valor
Grass enabled	True
Grass Regrowth Time	31
Initial Sheep Population	100
Sheep Reproduction Rate	0.09
Initial Wolf Population	18
Wolf Reproduction Rate	0.01
Wolf Gain From Food Rate	20
Sheep Gain From Food	10

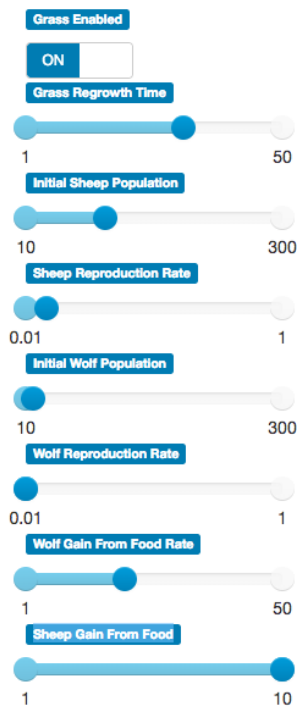


Figura 6: Cas 3 - Valors de les variables.

Al gràfic, es pot observar que el nombre d'ovelles i llops fluctuen, però menys brúscament que al cas anterior. Això, és degut, en part, a que s'ha reduït la població inicial de llops i la seva probabilitat de reproducció. També, s'ha incrementat el guany de menjar de les ovelles i la seva probabilitat de reproducció.

A més, per tal de compensar el guany de menjar de les ovelles, s'ha incrementat el temps de l'herba de tornar a estar al nivell màxim per a ser menjada.

El resultat és que les fluctuacions són més suaus i permeten mantenir l'ecosistema més estable.

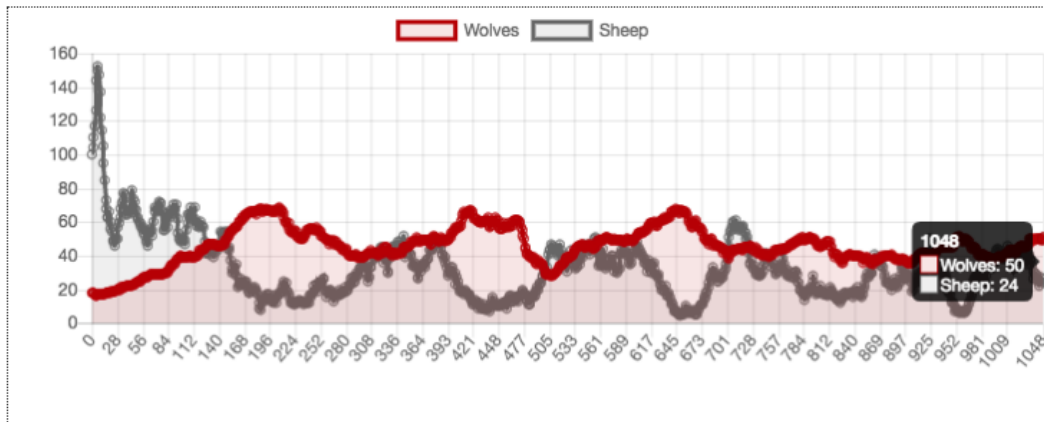


Figura 7: Cas 3 - Gràfic de l'evolució de les ovelles i els llops.

Modificacions del codi:

Ara, he aplicat les modificacions que se'ns demana a l'enunciat: afegit la variable edat per als agents Sheep i Wolf. És a dir, a partir d'ara, les ovelles i els llops només es podran reproduir si tenen més d'un any de vida i, al cap de 20 anys, moriran.

He considerat, tal i com es recomanava a l'enunciat, que una iteració equival a un mes de vida.

Els canvis aplicats al codi són els següents:


```

... wolf_sheep/agents.py

Hunk 1 : Lines 4-11
4 4
5 5     from wolf_sheep.random_walk import RandomWalker
6 6
7 7 + YEAR = 12
8 8 + MAX_LIVED_YEARS = 20 * YEAR
9 9
10 10     class Sheep(RandomWalker):
11 11         ...

Hunk 2 : Lines 19-32
17 19     def __init__(self, pos, model, moore, energy=None):
18 20         super().__init__(pos, model, moore=moore)
19 21         self.energy = energy
20 22 +         self.age = 0
21 23
22 24     def step(self):
23 25         ...
24 26         A model step. Move, then eat grass and reproduce.
25 27         ...
26 28         self.random_move()
27 29 +         self.age += 1
28 30         living = True
29 31
30 32         if self.model.grass:

Hunk 3 : Lines 42-53
38 42         grass_patch.fully_grown = False
39 43
40 44         # Death
41 45 -         if self.energy < 0:
42 46 +         if self.energy < 0 or self.age > MAX_LIVED_YEARS:
43 47             self.model.grid._remove_agent(self.pos, self)
44 48             self.model.schedule.remove(self)
45 49             living = False
46 50 -         if living and random.random() < self.model.sheep_reproduce:
47 51 +         if living and random.random() < self.model.sheep_reproduce and self.age > YEAR:
48 52             # Create a new sheep:
49 53             if self.model.grass:
                self.energy /= 2

```

Figura 8: Modificacions al codi. Arxiu agents.py

Hunk 4 : Lines 66-76	
62 66	def __init__(self, pos, model, moore, energy=None):
63 67	super().__init__(pos, model, moore=moore)
64 68	self.energy = energy
69 +	self.age = 0
65 70	
66 71	def step(self):
67 72	self.random_move()
73 +	self.age += 1
68 74	self.energy -= 1
69 75	
70 76	# If there are sheep present, eat one
Hunk 5 : Lines 86-96	
80 86	self.model.schedule.remove(sheep_to_eat)
81 87	
82 88	# Death or reproduction
83 -	if self.energy < 0:
89 +	if self.energy < 0 or self.age > MAX_LIVED_YEARS:
84 90	self.model.grid._remove_agent(self.pos, self)
85 91	self.model.schedule.remove(self)
86 92	else:
87 -	if random.random() < self.model.wolf_reproduce:
93 +	if random.random() < self.model.wolf_reproduce and self.age > YEAR:
88 94	# Create a new wolf cub
89 95	self.energy /= 2
90 96	cub = Wolf(self.pos, self.model, self.moore, self.energy)

Figura 9: Modificacions al codi. Arxiu agents.py

```
... wolf_sheep/model.py

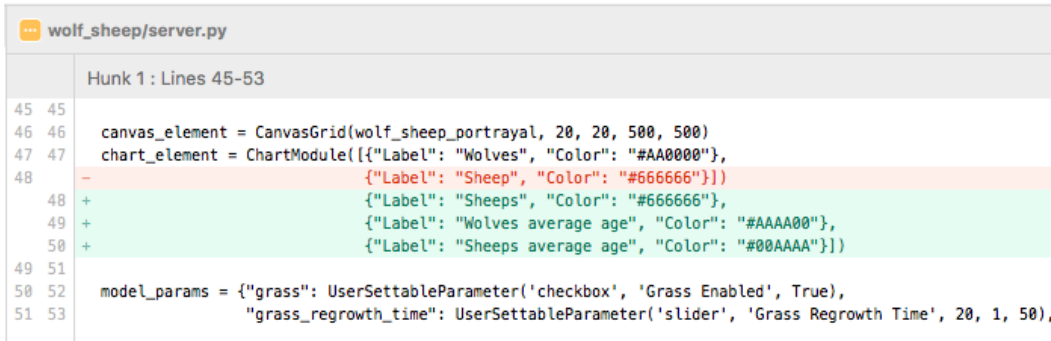
Hunk 1 : Lines 79-97
79 79         self.grid = MultiGrid(self.height, self.width, torus=True)
80 80         self.datacollector = DataCollector(
81 81             {"Wolves": lambda m: m.schedule.get_breed_count(Wolf),
82 82             - "Sheep": lambda m: m.schedule.get_breed_count(Sheep)}}
82 82         + "Sheeps": lambda m: m.schedule.get_breed_count(Sheep),
83 83         + "Wolves average age": lambda m: m.schedule.get_breed_avg_age(Wolf),
84 84         + "Sheeps average age": lambda m: m.schedule.get_breed_avg_age(Sheep)}}
85 85         +
```

Figura 10: Modificacions al codi. Arxiu model.py

```
... wolf_sheep/schedule.py

Hunk 1 : Lines 77-95
77 77         Returns the current number of agents of certain breed in the queue.
78 78         '''
79 79         return len(self.agents_by_breed[breed_class])
80 80         +
81 81         + def get_breed_avg_age(self, breed_class):
82 82         +         '''
83 83         +         Returns average age for a specific breed.
84 84         +         '''
85 85         +         total_age = 0
86 86         +         agents = self.agents_by_breed[breed_class]
87 87         +
88 88         +         for agent in agents:
89 89         +             if hasattr(agent, 'age'):
90 90         +                 total_age = total_age + agent.age
91 91         +
92 92         +         if total_age == 0:
93 93         +             return 0
94 94         +         else:
95 95         +             return round(float(total_age) / len(agents), 1)
```

Figura 11: Modificacions al codi. Arxiu schedule.py



```

45 45 canvas_element = CanvasGrid(wolf_sheep_portrayal, 20, 20, 500, 500)
46 46 chart_element = ChartModule([{"Label": "Wolves", "Color": "#AA0000"},
47 47 - {"Label": "Sheep", "Color": "#666666"}])
48 48 + {"Label": "Sheeps", "Color": "#666666"},
49 49 + {"Label": "Wolves average age", "Color": "#AAAA00"},
50 50 + {"Label": "Sheeps average age", "Color": "#00AAAA"}])
49 51
50 52 model_params = {"grass": UserSettableParameter('checkbox', 'Grass Enabled', True),
51 53 "grass_regrowth_time": UserSettableParameter('slider', 'Grass Regrowth Time', 20, 1, 50),

```

Figura 12: Modificacions al codi. Arxiu server.py

Cas 1 (amb el codi modificat):

Si es torna a executar els casos anteriors, es pot observar que, per als paràmetres del primer cas, es segueix obtenint els dos tipus de situacions: o bé, els llops es mengen les ovelles i després moren de fam (falta d'energia), o bé, els llops es moren i les ovelles aconsegueixen sobreviure obtenint una població mitjana al voltant dels 50 exemplars.

També es pot veure, al gràfic de la situació 1, que la mitjana de l'edat de les ovelles es troba als voltants dels 40 mesos (3 anys i 4 mesos). Cal dir, que la causa principal de mort de les ovelles és degut a la falta de menjar i que molt poques ovelles moren com a conseqüència d'haver arribat als 20 anys de vida (a partir de la iteració 240). Els llops també moren per falta de menjar.

Per tal de visualitzar aquesta informació he afegit `print` al codi (no es mostra a les imatges anteriors de les modificacions).

En el gràfic de la situació 2, les ovelles es moren perquè els llops se les mengen i els llops es moren tots de fam (no arriben a la iteració 240).

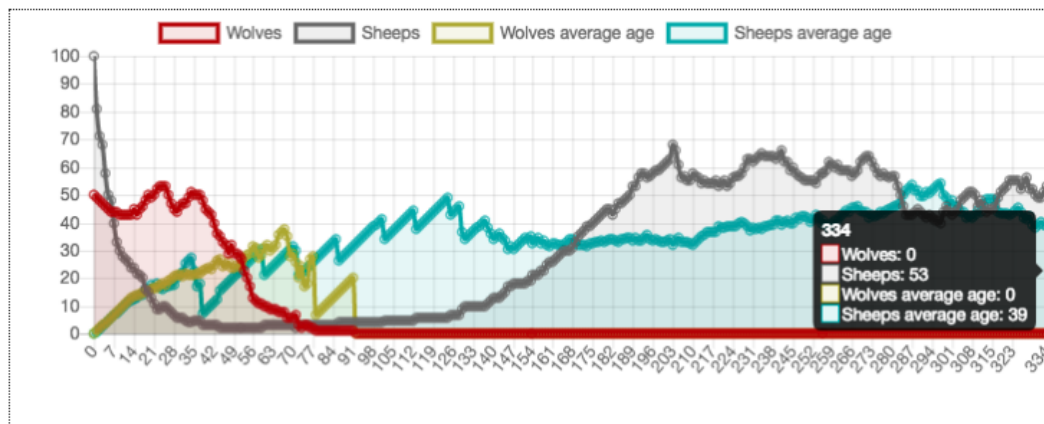


Figura 13: Cas 1 - Situació 1. Gràfic de l'evolució de les ovelles i els llops.

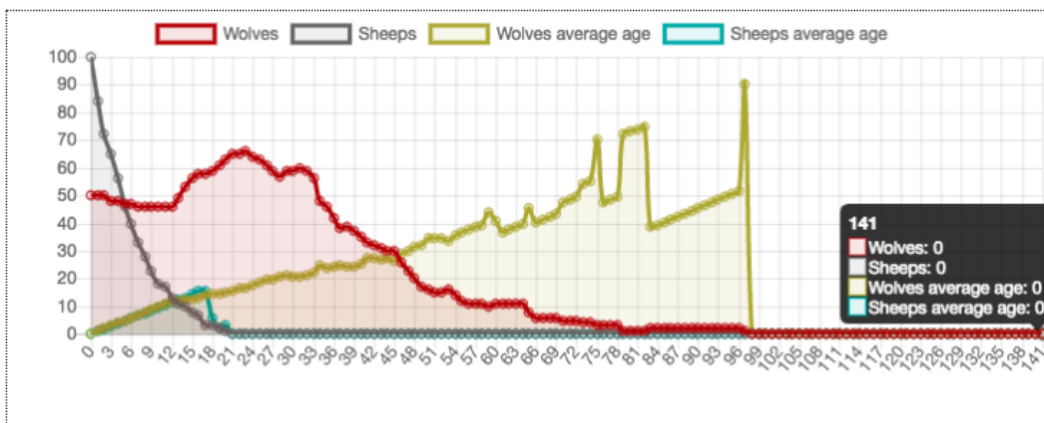


Figura 14: Cas 1 - Situació 2. Gràfic de l'evolució de les ovelles i els llops.

Cas 2 (amb el codi modificat):

En aquest cas, es veu com les ovelles es mantenen en una mitjana d'edat al voltant dels 10 mesos, mentre que els llops tenen una mitjana d'edat molt més alta (al voltant dels 30 mesos, és a dir, 2 anys i 6 mesos). De totes formes i, tal i com he comentat al cas anterior, els agents (ovelles o llops) que es moren per arribar al seu màxim d'edat és molt reduït, la gran majoria moren de fam.

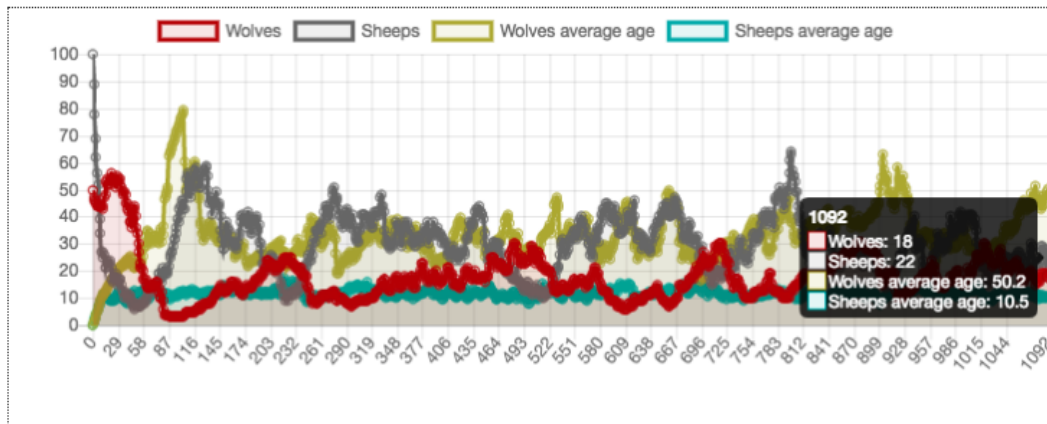


Figura 15: Cas 2 - Gràfic de l'evolució de les ovelles i els llops.

Cas 3 (amb el codi modificat):

En aquest cas, s'observa que la mitjana d'edat dels llops ha augmentat considerablement (al voltant dels 100 mesos, és a dir, 8 anys i 4 mesos) i la mitjana d'edat de les ovelles es manté al voltant dels 20 mesos (1 any i 8 mesos).

Tot i això, es segueix mantenint la dinàmica que els agents es moren de fam i no de vells.

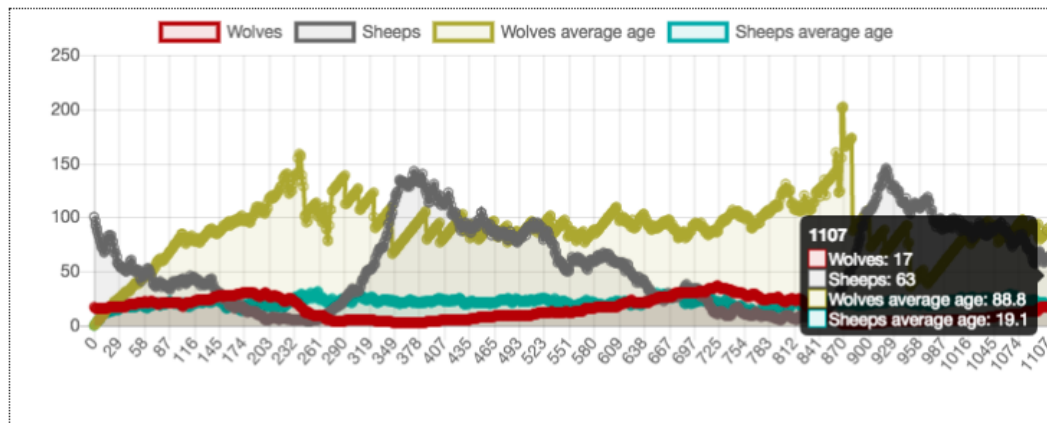


Figura 16: Cas 3 - Gràfic de l'evolució de les ovelles i els llops.

Cas 4 (amb el codi modificat):

Per tal d'intentar observar els efectes d'haver afegit la variable edat, he modificat tots els paràmetres deixant-los al mínim, excepte el del guany del menjar per a les ovelles, que l'he col·locat al màxim. D'aquesta manera, he reduït, expressament, al màxim la vida i la supervivència dels llops, i he augmentat la supervivència, però no reproducció de les ovelles. Això, hauria de permetre que un agent Sheep tingués menjar suficient i sempre disponible per a la seva supervivència i, així, només hauria de morir de vellesa.

Cal remarcar que m'he centrat en el cas de mantenir vives les ovelles i no els llops, ja que és molt més senzill d'aconseguir i el sistema resultant és molt més simple. Només cal centrar-se en 2 agents: Sheep i GrassPatch, ja que l'agent Wolf queda eliminat ben al principi.

En executar la simulació, a la figura 19, sí que es veu que a la iteració 240 hi ha una davallada de 10 ovelles (són les ovelles inicials de partida), però a la llarga se'n reproduïxen més que no pas moren d'envelliment (a la primera fase pràcticament cap mor de fam). Finalment, tal i com es pot observar a la figura 20, quan s'estabilitza la situació, moltes de les ovelles tornen a morir per falta de menjar. De totes maneres, cal dir, que, un cop estabilitzat, sí que hem aconseguit un ratio més elevat de morts per envelliment (aproximadament una tercera part). Això queda reflectit a la mitjana d'edat de les ovelles, que ara es troba al voltant dels 95 mesos (7 anys i 11 mesos).

També, a la figura 18, es pot observar com durant els primers 29 mesos, la mitjana d'edat augmenta linealment ja que no hi ha cap ovella que es mori menjada pels llops (en aquesta simulació, els llops moren sense haver aconseguit menjar cap ovella) i perquè tampoc aconsegueixen reproduir-se (recordar que tenim la condició de que no es poden reproduir si no tenen 12 mesos com a mínim de vida).

Cas 4	
Variable	Valor
Grass enabled	True
Grass Regrowth Time	1
Initial Sheep Population	10
Sheep Reproduction Rate	0.01
Initial Wolf Population	10
Wolf Reproduction Rate	0.01
Wolf Gain From Food Rate	1
Sheep Gain From Food	10

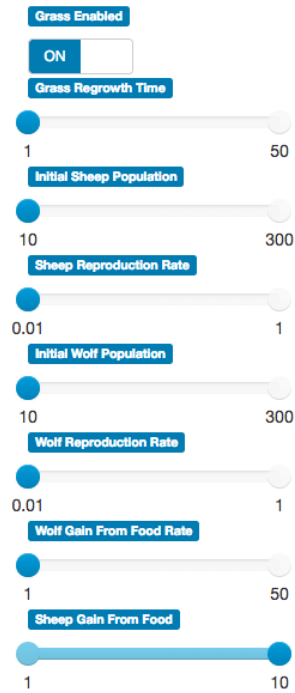


Figura 17: Cas 4 - Valors de les variables.

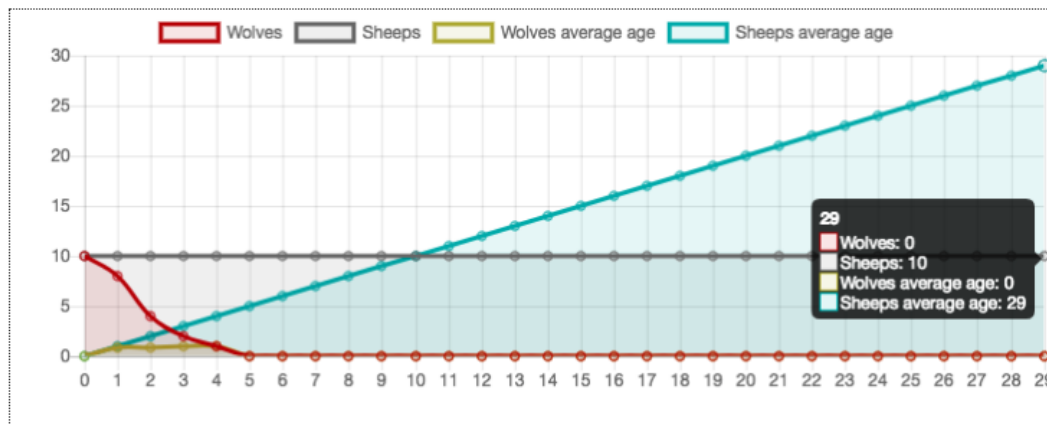


Figura 18: Cas 4 - Gràfic de l'evolució de les ovelles i els llops. Zoom per a les 29 primeres iteracions.

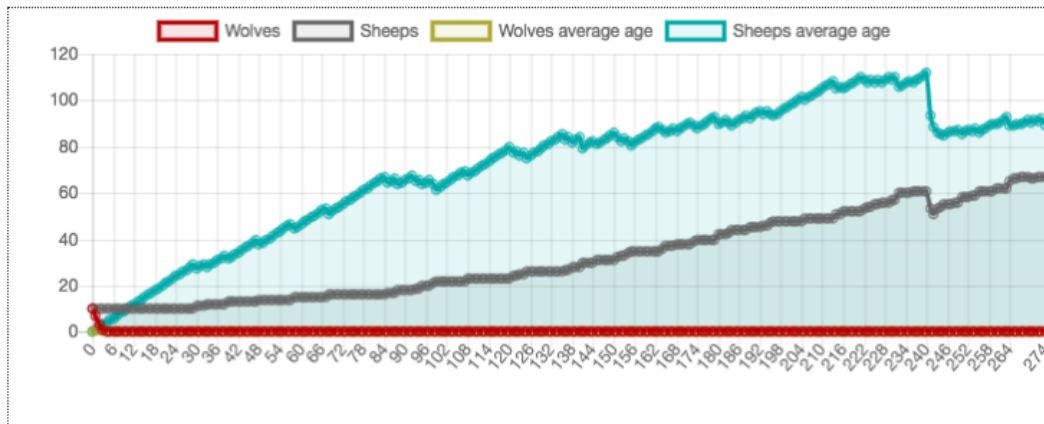


Figura 19: Cas 4 - Gràfic de l'evolució de les ovelles i els llops. Zoom per a les 274 primeres iteracions.

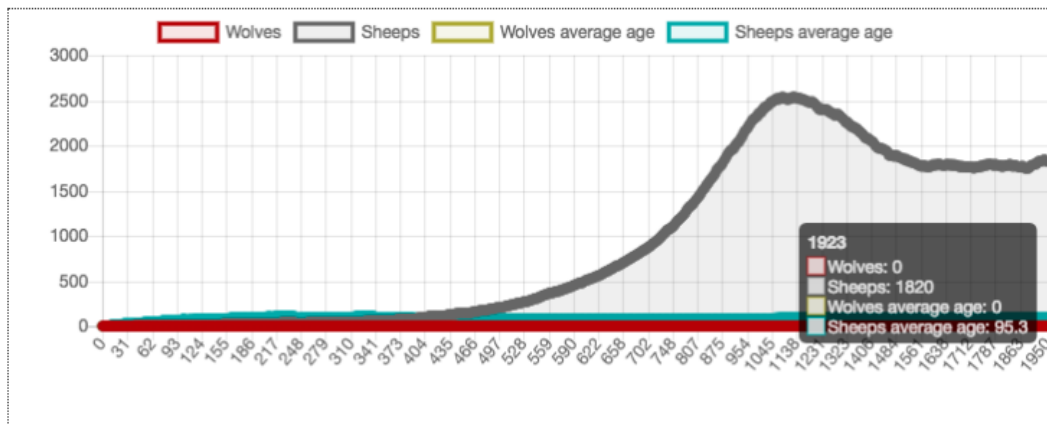


Figura 20: Cas 4 - Gràfic de l'evolució de les ovelles i els llops. Simulació completa.

3 Exercici 3

En un sistema multi-agent on els agents sheep i Wolf poden comunicar-se entre ells, penseu 2 tipus d'accions podrien realitzar per fer més realista el sistema, quin protocol FIPA utilitzarien i que informació es passarien.

Es podria pensar en un sistema de comunicació centralitzada a nivell de tipus d'agent (o Sheep o Wolf, suposem que GrassPatch no té capacitat de reaccionar als estímuls externs ni intel·ligència), els agents d'un tipus en particular es podrien comunicar a través d'una pissarra informant de la seva posició, posició dels altres agents al seu voltant i comunicació d'on són les possibles preses o depredadors i, en el cas de les ovelles, els trossos d'herba per a poder alimentar-se.

Els agents d'un tipus concret podrien accedir a aquesta informació i utilitzar-la per prendre decisions per fer més eficient la seva distribució a la quadrícula. A més, disposar d'informació de varis agents (en alguns casos redundant) milloraria la robustesa i riquesa de coneixement del sistema a l'hora de prendre decisions. Així, podríem definir dos tipus de comunicacions:

- Una ovella es vol comunicar amb altres ovelles per saber si es pot menjar un tros d'herba. Utilitzaria el FIPA-Request per a preguntar si hi ha alguna altra ovella a la mateixa casella que estigui més necessitada que ella (tingui menys energia i no sigui vella).

En aquest cas, fins i tot, es podria afegir algun tipus de lògica fent que si una altra ovella no es troba a la mateixa casella però molt aprop i realment està molt més necessitada que l'actual, li indiqui la coordenada a la qual s'ha de moure i li cedeixi l'herba a l'espera que en els següents torns se la pugui menjar.

- Un llop es vol comunicar amb altres llops per saber si pot menjar-se una ovella. Utilitzaria el FIPA-Request per a preguntar si hi ha algun altre llop a la mateixa casella que estigui més necessitat que ell (tingui menys energia i no sigui vell).

En aquest cas, no tindria massa sentit que un llop cedís una ovella a algun altre llop que es trobés aprop de la casella i més necessitat, ja que les ovelles van canviant de posició i, a no ser que s'afegís algun tipus d'intel·ligència als llops per a permetre seguir-les, es podria perdre el seu rastre en els torns següents.