UNIVERSITE PIERRE & MARIE CURIE
PARIS

MASTER IAD

SSC TME 2
Coordination spatiale adaptative

12 mars 2006

Par:
ALEXANDRE BARGETON
BENJAMIN DEVÈZE

TABLE DES MATIÈRES

CHAPITRE 1

PRÉSENTATION DU TRAVAIL EFFECTUÉ

Sommaire

1.1 Description du problème

X bergers doivent regrouper Y moutons, initialement aléatoirement répartis dans un monde clos, dans un enclos/bergerie.

1.2 Solutions implémentées

Nous avons implémenté trois approches différentes en NetLogo :
- Une approche naïve
- Un raffinement de l'approche précédente avec une combinaison de différentes forces pondérées
- Autre approche par aggrégation de forces pondérées

L'interface de notre implémentation se trouve à la figure 1.1. Nous aurons l'occasion de décrire plus précisément le rôle des différents boutons de l'interface dans le corps du rapport.

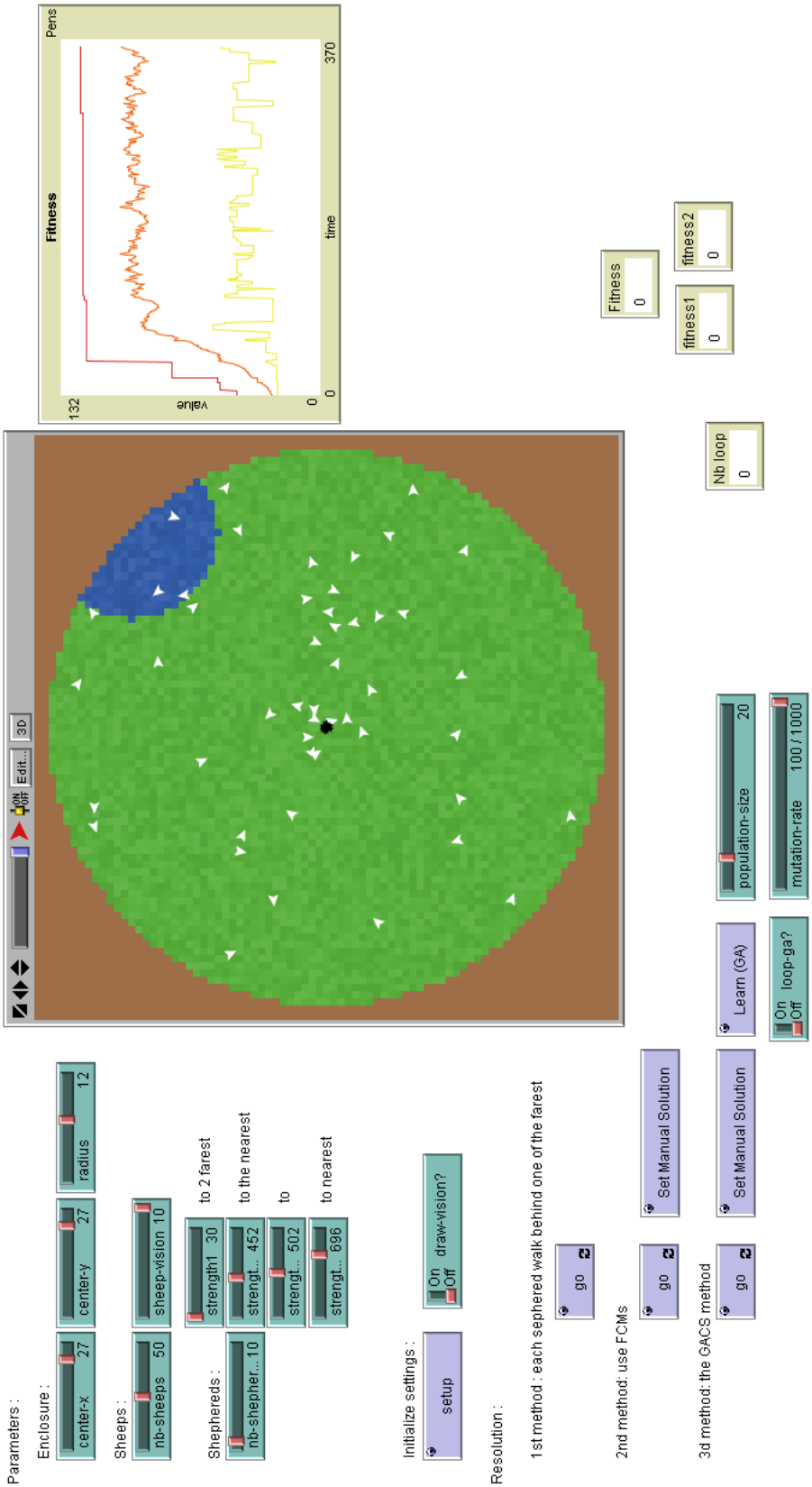

FIG. 1.1: *Interface*

1.2.1 Approche naïve

Modélisation du monde

L'enclos est situé contre un mur. le monde est entouré d'une muraille circulaire, ce qui fait qu'il ne peut y avoir de cas où un mouton se retrouve coincé dans un coin.

Modélisation des moutons

La modélisation des moutons adoptée dans le cadre de cette première approche est une version « statique » (i.e. version implémentée dans un algorithme classique) d'une implémentation des FCMs (décrite plus loin), que nous pouvons représenter par l'algorithme 1.1 :

Algorithme 1.1 Comportement du mouton

```
1:  if au moins un berger est dans mon champs de vision then
2:      if l'un d'entre eux me poursuit then
3:          je le fuis
4:      else
5:          je fuis le berger le plus proche de moi
6:      end if
7:  else
8:      if d'autres moutons sont dans mon champs de vision then
9:          je me rapproche d'eux, i.e. du centre de gravité du groupe
10:     else if je ne suis pas dans l'enclos then
11:         je bouge devant moi avec un angle aléatoire de plus ou moins 50
12:     else
13:         j'avance tout droit afin de sortir de l'enclos
14:     end if
15: end if
```

Notons que tous les mouvements sont pondérés de facons à éviter les murs, i.e. si un mur est présent sur la gauche du mouton, il va effectuer une rotation vers la droite (et inversement), et si le mur est face à lui, il effectuera un évitement aléatoire vers la gauche ou vers la droite.

Avec cette modélisation régie par quelques règles simples, d'importance hierarchique variable, les moutons semblent animés (via simulation) d'un comportement assez naturel.

Du fait que les moutons fuient les bergers, ne restent pas dans l'enclos et contournent les murs ; les bergers devront, pour réaliser une solution stable (i.e. les moutons prisonniers dans l'enclos), élaborer une stratégie coopérative.

Pour cela, ils pourront utiliser le fait que les moutons ont un comportement social qui les poussent à se regrouper en troupeau.

Modélisation des bergers

Première approche Chaque berger suit le mouton le plus loin de l'enclos dans l'espoir que dans sa fuite le mouton ira vers l'enclos. Dès qu'un autre mouton est plus loin, le ou les bergers vont poursuivre cet autre mouton et rechange de mouton cible de la meme manière.

Cette méthode très naïve n'aboutit absolument pas au résultat escompté. En effet tous les bergers poursuivent tout le temps le mouton le plus loin (il n'y a aucun mécanisme de coordination).

Coopération Afin de « coopérer », chaque berger se voit affecter d'un des moutons le plus loin (i.e. un premier berger poursuit le mouton le plus éloigné de la bergerie, le second berger poursuit le second mouton le plus loin,...). Ainsi chaque berger s'occupe d'un mouton.

Il est à noter que plusieurs bergers peuvent poursuivre un même troupeau de moutons (si plusieurs moutons les plus loins sont regroupés).

Cette version fonctionne mieux que la précédente mais souffre encore de diverses limitations dont notamment l'instabilitée de la solution obtenue (i.e. si un mouton est dans la bergerie, le berger qui s'en occupait continue de le poursuivre, ce qui implique une fuite du mouton de la bergerie, la fuite des autres moutons qui y étaient, etc.).

Stabilité (1) Afin de pallier le problème, un mouton n'est « affecté » à un berger, que s'il n'est pas dans l'enclos. Un berger ne se déplace que s'il est affecté à un mouton.

On observe dès lors une « faible stabilité » de la solution. Les moutons les plus loin seront rabattus vers l'enclos (en fuyant les bergers et en contournant les murs), les bergers s'arrêtent avant d'être dans l'enclos (leur mouton y étant).

Il n'y a pas toujours émergence d'une solution stable, en effet si tous les bergers ainsi que tous les moutons arrivent du même côté de l'enclos, alors les moutons fuiront par le côté opposé de l'enclos et les bergers devront continuer à les poursuivre comme le montre la figure 1.2

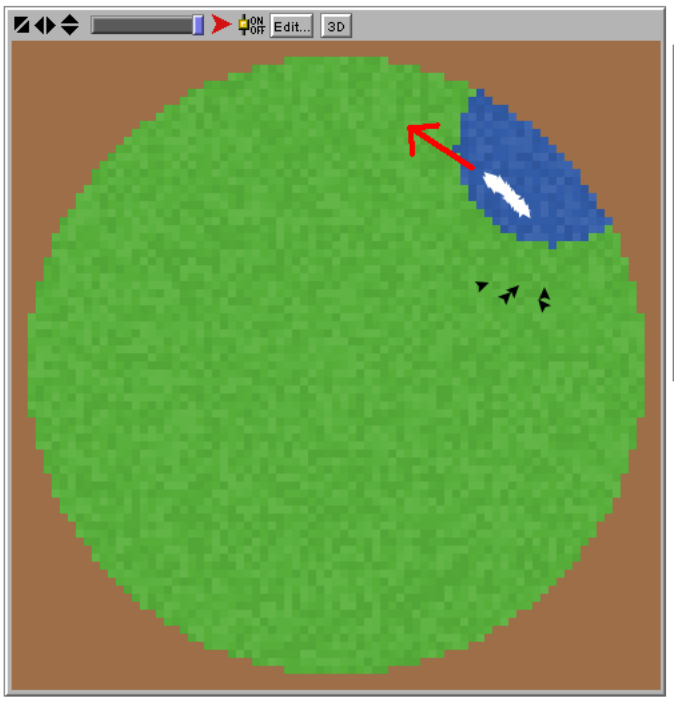

FIG. 1.2: *Solution instable*

Stabilité (2) La dernière amélioration de notre approche naïve consiste à ne plus poursuivre le mouton directement, mais à poursuivre le point placé à une unité derrière le mouton (derrière par rapport à l'enclos).

Cette dernière petite, mais ingénieuse, amélioration, permet de rabattre progressivement les moutons vers la bergerie et de construire une solution stable comme le montre la séquence suivante.

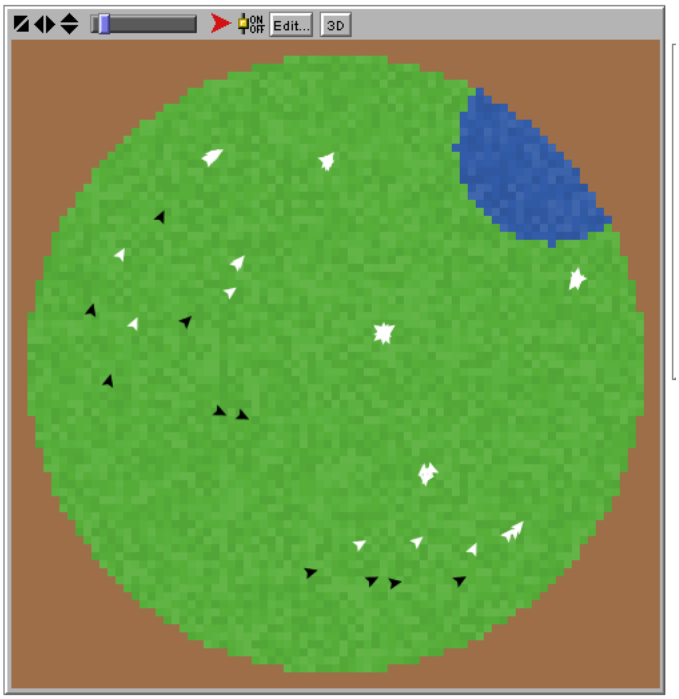

FIG. 1.3: *Etape 1*

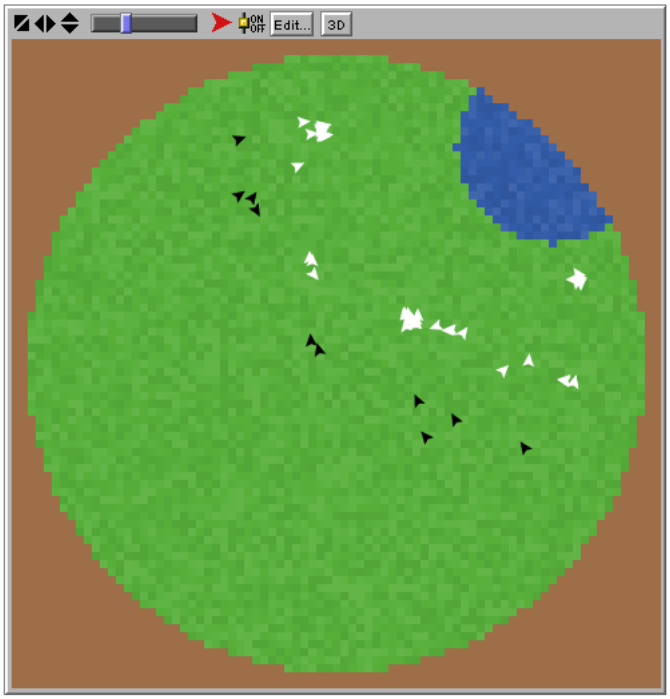

FIG. 1.4: *Etape 2*

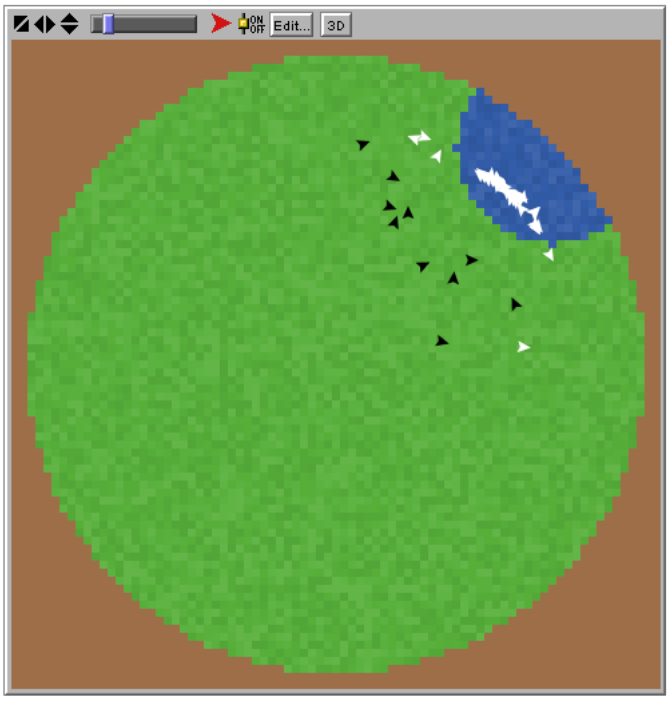

FIG. 1.5: *Etape 3*

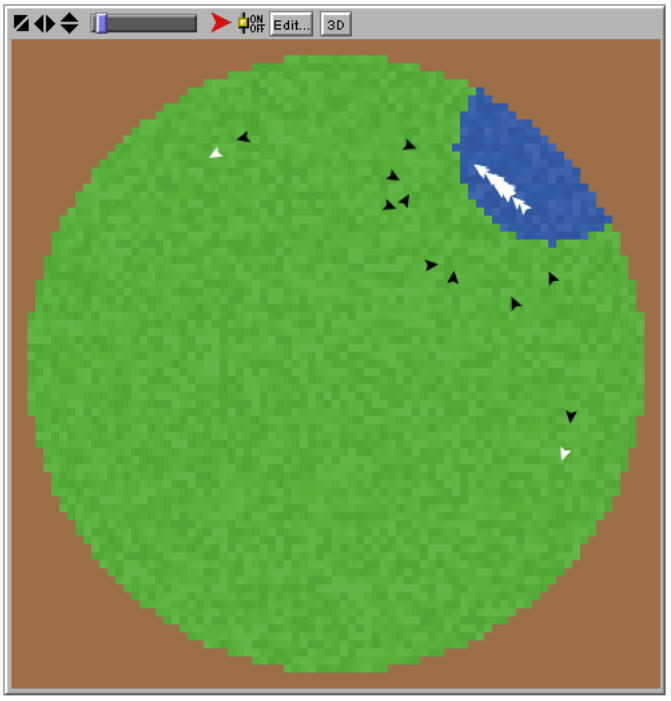

FIG. 1.6: *Etape 4*

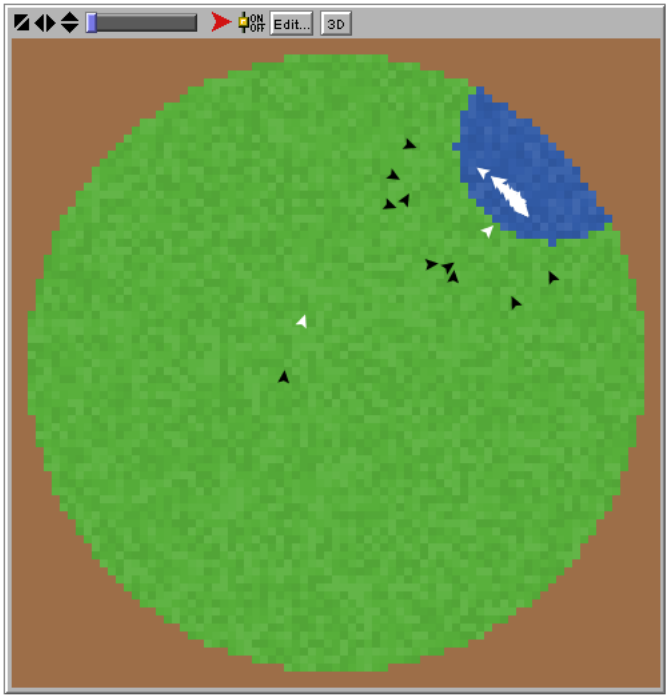

FIG. 1.7: *Etape 5*

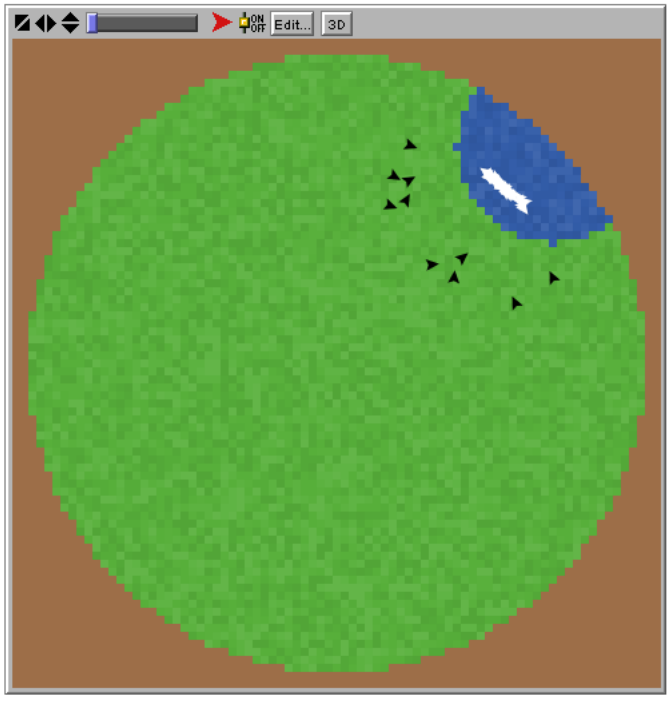

FIG. 1.8: *Etape 6*

Des moutons peuvent encore s'échapper de l'enclos, mais de facon très ponctuelle (un ou deux moutons). Quand cela est le cas, un ou deux bergers poursuivent ces moutons et les rabattent vers la bergerie, les autres restant à leur poste afin d'assurer la stabilité de la

solution précédemment obtenue comme en témoignent les figures 1.6, 1.7 et 1.8.

Stabilité (3) Quand un mouton fuit de la bergerie, il va le faire de façon à fuire le berger qui le poursuivait. Il va donc fuire par l'autre côté de la bergerie. Dans ce cas là le dit berger, va poursuivre le mouton en traversant la bergerie, ce qui va avoir pour effet de disperser les moutons. Pour remédier à ce dernier problème, les bergers ont interdiction de marcher dans la bergerie, et doivent en faire le tour le cas échéant.

Finalement, le comportement des bergers peut être représenté par l'algorithme 1.2 :

Algorithme 1.2 Comportement des bergers

```
1: affecter le mouton le plus loin, qui n'est pas déjà affecté et qui n'est pas dans l'enclos
   à un berger.
2: for all berger do
3:     if un mouton m'est affecté then
4:         poursuivre le mouton 1 unité derriere lui par rapport à l'enclos
5:         if je vais vers la bergerie then
6:             la contourner
7:         end if
8:     else
9:         ne pas bouger
10:    end if
11: end for
```

Cette approche fonctionne parfaitement bien sur cette modélisation du monde. Cependant celle-ci est trop simple. En effet la bergerie est située contre un mur. Si on la déplace, les moutons qui ont tendance à longer les murs en fuyant les bergers, n'ont plus aucune chance de se retrouver dans la bergerie comme présenté sur la figure 1.9.

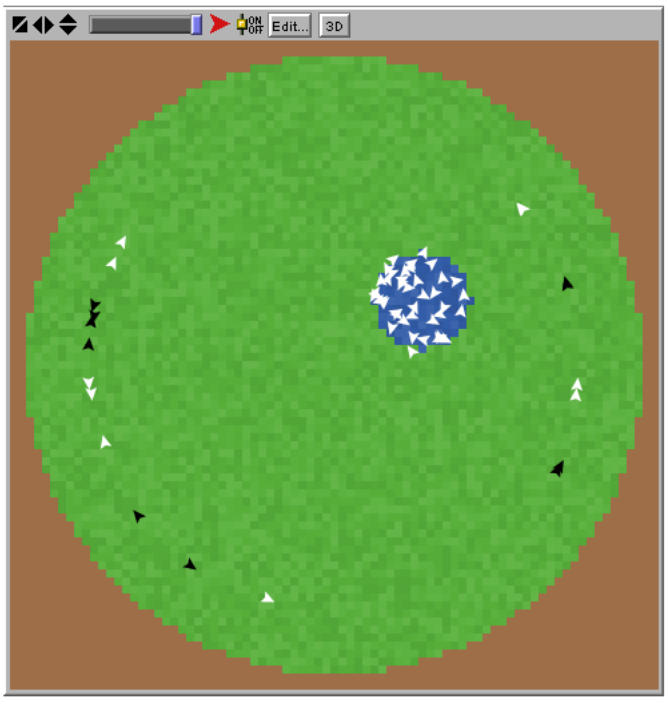

FIG. 1.9: *Problème lorsque l'enclos n'est pas contre un mur*

1.2.2 Approche naïve avec forces

Modélisation du monde

L'enclos peut cette fois-ci être positionné n'importe où dans le monde.

Modélisation des moutons

Identique à la modélisation précédente.

Modélisation des bergers

Cette modélisation se base sur la modélisation précédente, mais essaye de prendre en compte des « forces ».

Les forces en questions sont :

- attirance vers le mouton le plus éloigné (affecté via le processus décrit précédemment, donc vers le point placé une unité derriere le dit mouton) (*strength1* sur l'interface)
- attiré par le mouton le plus près de l'enclos sans affectation particulière à chacun des bergers (toujours une unité derrière)(*strength2* sur l'interface)
- répulsé par la bergerie i.e. le centre de la bergerie (*strength3* sur l'interface)

Le slider *strength4* de l'interface n'est pas pris en compte dans cette modélisation.

Chaque force est divisée par la distance du berger à l'élément qu'elle représente. De plus chaque force est pondérée (choix de l'utilisateur via les sliders).

Cette approche fonctionne pour certaines valeurs des trois forces et ce quelque soit la position de l'enclos dans le monde. Cependant, il est à noter que la stabilité de la solution dépend fortement du nombre de bergers. En effet, il faut que ces derniers puissent être en nombre suffisant afin d'entourer l'enclos et donc d'empêcher la fuite des moutons par l'un des côtés de l'enclos.

Il est à noter aussi l'émergence de l'encerclage de l'enclos (figure 1.10) par les bergers, alors que ce comportement n'a pas été explicitement implémenté (cette émergence n'a pas lieu quand tous les moutons sont toujours du même coté des bergers, ce qui est assez rare, vu l'initialisation aléatoire de la place des moutons dans le monde).

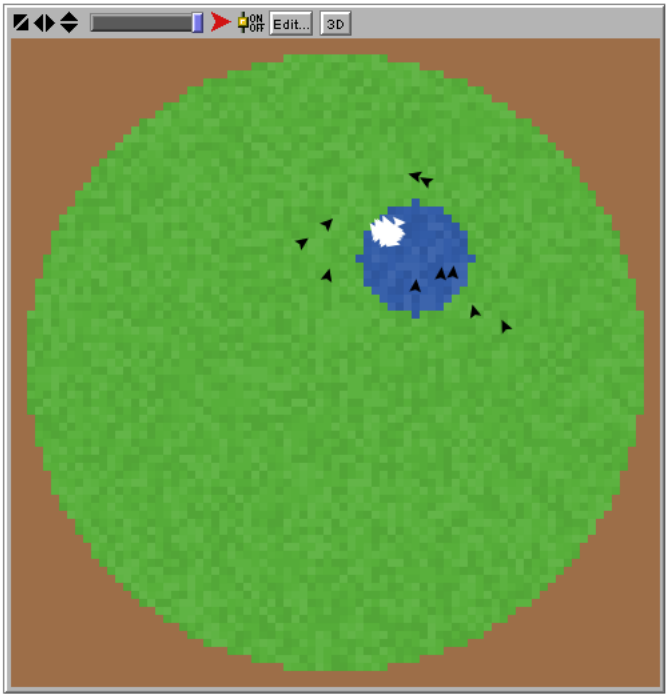

FIG. 1.10: *Encerclage de l'enclos*

La solution (poids des différentes forces) que nous avons trouvée manuellement, peut être positionnée automatiquement en appuyant sur le bouton *Set Manual Solution* à coté du bouton *go* de la deuxième méthode.

Finalement, le comportement des bergers peut être représenté par l'algorithme 1.3 :

Algorithme 1.3 Comportement des bergers

```
1: affecter le mouton le plus loin, qui n'est pas déjà affecté et qui n'est pas dans l'enclos
   à un berger.
2: for all berger do
3:     if si un mouton m'est affecté then
4:         calculer chacune des forces pondérée par la distance de l'objet au berger
5:         calculer la résultante de la somme pondérée des forces
6:         déplacer le berger sans condition particulière pour l'évitement de la bergerie
7:     else
8:         ne pas bouger
9:     end if
10: end for
```

1.2.3 Approche à base de forces

Dernière modélisation, celle-ci est exclusivement basée sur une somme pondérée de force, tant pour les moutons que pour les bergers..

Modélisation du monde

Identique à la précédente.

Modélisation des moutons

Les moutons sont dirigés par une force résultante de l'addition de cinq forces :

1. Attiré par un troupeau (i.e. l'ensemble des moutons dans son angle de vue) (en fait attiré par le centre de gravité de cet ensemble de moutons).
2. Repoussé par le mouton le plus proche. Cette répulsion est proportionnelle à la distance qui les séparent (i.e. plus ils sont proches, plus ils se repoussent).
3. Repoussé par la bergerie proportionnellement à la distance à la bergerie.
4. Repoussé par les murs (et donc attiré par le centre du monde) proportionnellement à la distance au centre.
5. Repoussé par les berger dans son angle de vue (en fait répulser par le centre de gravité de ces bergers).

Ces cinq forces ne sont pas spécifiquement pondérées par des paramètres modulables par l'utilisateur.

Modélisation des bergers

Le mouvement des bergers est régi par quatre forces :

1. Attiré par les 2 moutons les plus loin (en fait par le point situé à une unité derriere eux et donc par le barycentre de ces 2 points) (*strength1* sur l'interface)
2. Attiré par le mouton le plus proche (en fait par le point situé à une unité derriere lui, par rapport à la bergerie toujours). (*strength2* sur l'interface)
3. Repoussé par la bergerie (i.e. par le centre de celle ci) la répulsion est proportionnelle par rapport à la distance du berger à celle-ci (*strength3* sur l'interface)
4. Repoussé par le berger le plus proche proportionnellement à la distance (*strength4* sur l'interface)

Le mouvement du berger est régi par la somme pondérée de chacune de ces forces (choix de l'utilisateur par les sliders de l'interface). La solution obtenue manuellement pour chacune des forces afin d'obtenir la solution escomptée (i.e. les moutons dans l'enclos) peut être initialisée en cliquant sur le bouton *Set Manual Solution* à coté du bouton *go*.

On peut remarquer que le comportement émergeant de nos bergers, satisfaisant intellectuellement, consiste à :

1. Se placer loin de la bergerie, derrière les moutons
2. Attendre que les moutons ne fassent plus qu'un seul troupeau
3. Pousser se troupeau dans l'enclos
4. Attendre près de l'enclos et empêcher les moutons de s'enfuir

Cela est présenté sur les figures 1.11, 1.12, 1.13 et 1.14.

FIG. 1.11: *Etape 1*

FIG. 1.12: *Etape 2*

FIG. 1.13: *Etape 3*

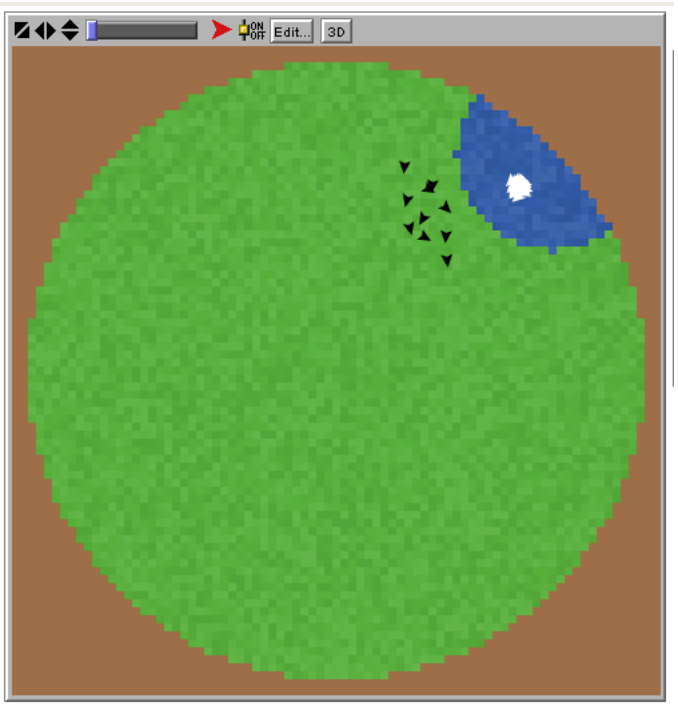

FIG. 1.14: *Etape 4*

On remarque que la stabilité de la solution se fait par un équilibre des force. Les moutons tentent de fuir par l'un des côtés, mais aussitôt, les bergers suivent ce mouvement et empêchent la fuite. Puis les moutons essayent de fuir par l'autre côté, etc.

1.2.4 Apprentissage

Trouver les paramètres pour chacune des 4 forces appliqués aux bergers, afin que ces derniers parviennent à enfermer les moutons dans l'enclos, est un processus long et fastidieux s'il doit être effectué à la main.

La deuxième partie du travail demandé était de mettre en place un algorithme génétique afin de trouver automatiquement les meilleurs paramètres pour chacune de ces forces (ou tout du moins des paramètres proches de l'optimum).

Deux versions des algorithmes génétiques sont communément utilisées : il s'agit des méthodes de *population replacement* et *incremental replacement*.

Pour notre part nous utilisons la méthode incrémentale car elle permet de garder une taille constante de la population et s'avère performante en pratique.

De même la sélection des parents afin de produire un nouveau chromosome peut se faire de plusieurs facons : *roulette wheel*, *tournament selection* ou *elitism*. Nous ne détaillons pas davantage ces méthodes bien connues dans le domaine et renvoyons à l'une des références de la question, qui bien qu'ancienne reste intéressante [Gol89].

Notre implémentation utilise la méthode de la roulette wheel avec rang.

Le remplacement d'un chromosome dans la population par un nouveau chromosome se fait en remplaçant l'un des plus « mauvais » chromosomes actuel (donc pas forcément tout le temps le plus mauvais, sur la meme idée qu'on ne reproduit pas forcément toujours les deux meilleurs individus). En fait on choisit le chromosome à remplacer aléatoirement parmi la deuxieme moitié de la population (la population étant triée par ordre décroissante de fitness).

L'algorithme général se déroule comme suit :

Algorithme 1.4 Apprentissage

1: générer aléatoirement et évaluer une population initiale de N individus.
2: **repeat**
3: trier la population par ordre décroissant des fitness
4: choisir 2 individus (roulette wheel with rank)
5: les croiser (cross over)
6: effectuer une mutation (selon le taux de mutation défini par l'utilisateur, qui est généralement compris entre 0,01 et 0,001)
7: sélectionner un « mauvais » individus
8: le remplacer par le nouveau chromosome
9: **until** condition d'arrêt

Il reste à spécifier trois choses :

- la modélisation de la fitness
- la modélisation des chromosomes, forces
- le principe du cross over

En ce qui concerne la modélisation des forces, une première méthode consiste à les modéliser en "unaire" (nombre de bits à 1 sur un zone de 1000 bits) en les initialisant aléatoirement.

La modélisation finalement retenue est celle qui consiste simplement à stocker quatre nombres (en base 10) (correspondant aux quatre forces) dans chaque chromosome (un chromosome étant donc un tableau de quatre nombres) initialisés aléatoirement entre 0 et 1000.

Le cross over quant à lui consiste à :

- choisir un point de coupe aléatoire (entre 0 et 4)
- copier dans le fils tous les nombres à gauche de ce point de coupe provenant du père.

– copier dans le fils tous les nombres à droite de ce point de coupe provenant de la mère.
– au niveau du point de coupe, on copie dans le fils la moyenne des nombres de la somme du père et de la mère

Exemple avec le point de coupe valant 2 :
père : 15 20 25 30
mère : 7 11 9 5
fils : 15 20 17 5

Enfin le calcul de la fitness se déroule en 2 étapes. Tout d'abord nous évaluons la distance moyenne des moutons par rapport à l'enclos (il faut donc minimiser ce coefficient). Ensuite il faut maximiser le nombre de moutons présents dans l'enclos. Afin de ne pas désavantager les solutions non stables, ce dernier coefficient n'est pas évalué au dernier moment de la simulation mais calculé tout au long de celle ci (i.e. nous retenons le nombre maximum de moutons qu'il y a eu dans l'enclos au cours de la simulation). Et, pour ne pas avantager une répartition de départ aléatoire qui serait favorable alors que la solution codée ne le serait pas (moutons proches de l'enclos au départ), les calculs de la fitness ne sont pris en compte qu'après 100 itérations.

Chaque individu de la population n'est évalué que pendant une durée de 500 itérations.

Au final, entre les itération 100 et 500, il s'agit de calculer :

```
fitness = (distance_max - distance moyenne des moutons à l'enclos)
+ nombre maximum de moutons qu'il y a eu dans l'enclos
```

Notre simulation a consisté en une population initiale de 20 individus, puis à laisser tourner l'apprentissage jusqu'à obtenir une population convenable. Le taux de mutation a été fixé à 1%.

L'évolution de la valeur de la fitness lors d'un apprentissage est présenté à la figure 1.15.

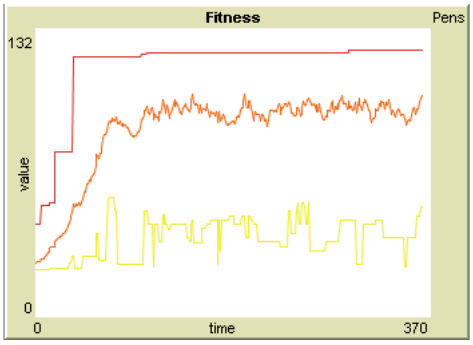

FIG. 1.15: *Evolution de la valeur maximale, moyenne et minimale de la fitness lors de l'apprentissage*

Concrètement, dans la population initiale, aucun individu ne codait de bonnes solutions. Les bergers étaient tous en situation d'interblocage (les bergers étant coincés entre deux troupeaux de moutons, ne sachant pas où donner de la tête, figure 1.16), puis petit à petit les individus se sont améliorés afin de coder un groupe de bergers capables de prendre en chasse un troupeau et de l'amener dans l'enclos, figure 1.17

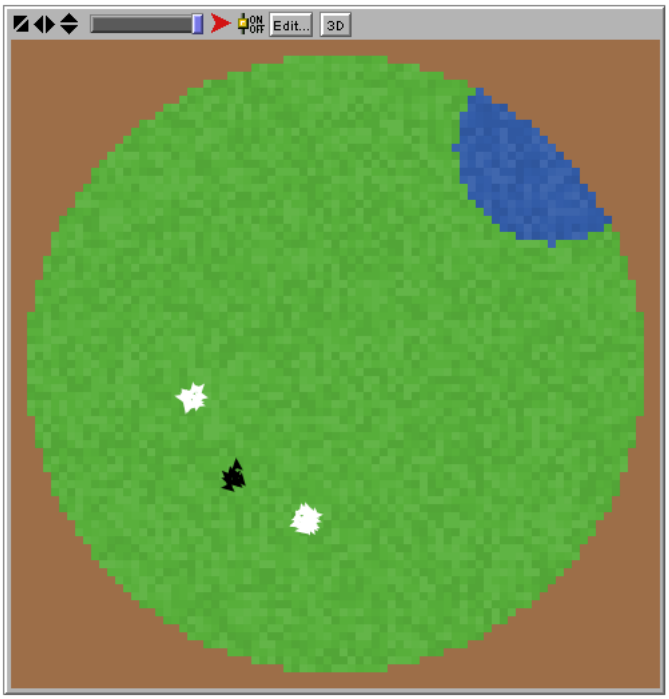

FIG. 1.16: *Début de l'apprentissage*

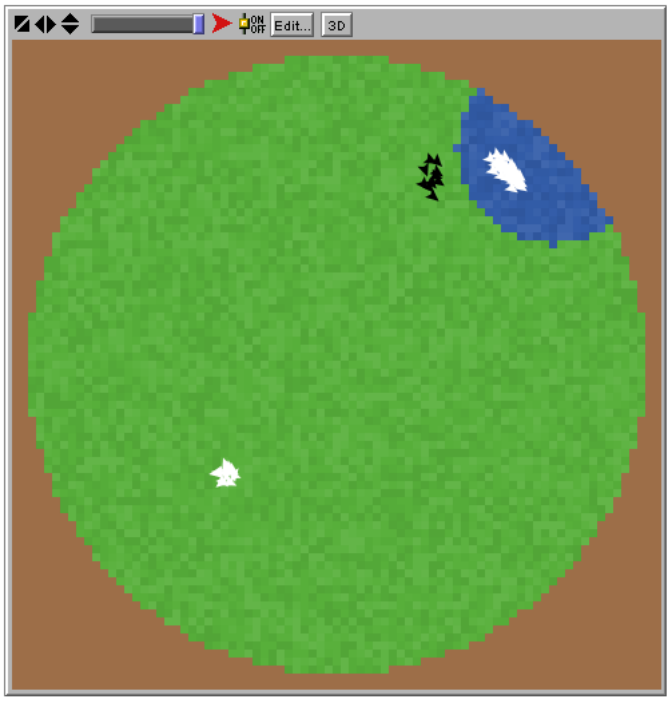

FIG. 1.17: *Phase ultérieure*

Malheureusement, ce résultat n'est pas aussi bon que celui trouvé à la main. Cela est sans doute lié à notre choix de fitness qui reste perfectible pour exprimer au mieux ce que doit être une bonne solution, mais également lié au fait que nous avons arrété la simulation

prématurément au bout d'environ 300 individus remplacés.

Il est à noter que nous aurions également pu mettre en oeuvre cet apprentissage sur notre deuxième solution afin d'en apprendre les coefficients.

1.2.5 Autre approche de l'apprentissage

Comme décrit dans [Par02], nous aurions pu implémenter des approches de type FCMs (Fuzzy Cognitive Map - Cartes Cognitives Floues) pour modéliser les moutons et bergers, et faire de l'apprentissage (via algorithmes génétiques) sur les coefficients de l'influence de chaque interaction sur chacun des centres cognitifs du berger et du mouton. On aurait ainsi pu assister à une adaptation de chacune des deux espèces pour atteindre leur but respectif.

1.3 Conclusion

Bien que nous n'ayons pas eu le temps de mener plus en avant nos investigations, les solutions proposées s'avèrent déjà relativement performantes et les bergers exhibent un comportement cohérent pour regrouper et maintenir les moutons dans l'enclos.

BIBLIOGRAPHIE

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[Par02] Marc Parenthoën. Autonomie et perception proactive pour des acteurs virtuels, Janvier 2002. Journée d'Information sur la Commande et le Diagnostic (JICD'02).