

TER 2015 - Rapport de modélisation

Bruno Y., Lionel F., François S., Alexandre V.

22 mai 2015

1 Partie Théorique

1.1 Introduction

Pour débiter, ce document explique le fonctionnement d'une modélisation ayant pour but de mettre en valeur l'émergence d'une civilisation. Cette implémentation se base sur MetaCiv, framework de modélisation de société humaine utilisant la technique des SMA. Toutes les constantes mentionnées dans ce document sont données dans la section "**Constantes**".

1.2 L'environnement

En premier lieu, l'environnement dans lequel nos agents se déplacent est un espace fermé de \mathbb{R}^2 . Il existe un total de quatre types de patches présents : Mer, Prairie, Terre Sterile et Forêt. Le modèle étant relativement basique, il n'existe que deux types de ressources disponibles : Les baies et le bois. Le tableau suivant donne le récapitulatif des différents patches et ressources disponibles.





Patch	Ressources		Valeur initiale		Croissance des ressources		Passabilité
Mer 	Aucunes		0		0		Non
Terre Sterile 	Aucunes		0		0		Oui
Prairie 	Baies	Bois	10	1	0.2	0	Oui
Forêts 	Baies	Bois	10	2	0.1	0.2	Oui

FIGURE 1 – Tableau récapitulatif des patches et des ressources.

Un patch de terre stérile ne produit pas de ressources alors que les patches Prairie et Forêts produisent en permanence des baies. On pourra remarquer que seules les forêts produisent du bois même si les prairies possèdent une valeur en bois initiale. Un patch peut être récolté dès lors que les ressources en bois ou en baies sont supérieures ou égales à 1. Une ressource "baies" est alors transformée en un objet baie (de même pour le bois).

Le point d'apparition des agents se situe au milieu de l'environnement et plusieurs prairies sont présentes aux alentours. On peut aussi distinguer une grande forêt dans la partie droite de l'environnement.

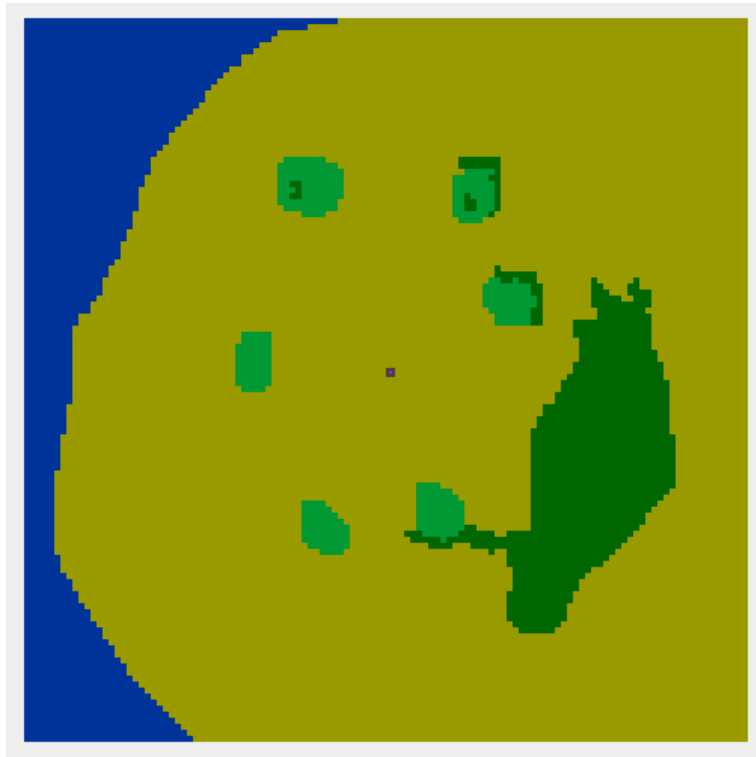


FIGURE 2 – Environnement de la modélisation

Il convient de remarquer que lorsque les agents apparaissent dans l'environnement, ils sont éparpillés autour du point d'apparition et non dessus.

1.3 Les agents

Un agent possède un ou plusieurs "Cognitons" qui permettent d'associer à chaque "plan", une importance. En effet, les cognitons peuvent influencer les plans de manière positive ou négative. De plus, chaque agent possède un certain nombre d'attributs représentant son état physique. Ils peuvent interagir avec leur environnement, manipuler des objets et construire des aménagements. Les agents sont initialement au nombre de 81 lors du lancement de la modélisation.

1.3.1 Les cognitons

Afin de décider quel plan choisir parmi l'ensemble des plans possibles, un système de pondération des plans a été mis en place : les cognitons. Chaque agent

possède un ensemble de cognitons qui représentent ce que l’agent aime, voit ou pense. Il existe deux interactions entre un cogniton et un plan. La première rend un plan ”accessible” par l’agent et la deuxième influence l’envie (ou la répulsion) d’effectuer un plan.

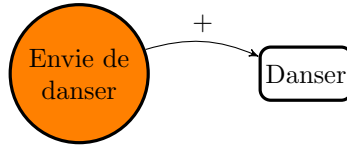


FIGURE 3 – Exemple d’un cogniton renforçant un plan (flèche positive).

Dans l’exemple précédent, on peut remarquer que le cogniton ”Envie de danser” influence positivement l’envie de l’agent à effectuer le plan ”Danser”, mais ce dernier n’étant pas accessible à l’agent, il ne peut l’effectuer. On rajoute alors un lien ”conditionnel” (qui peut éventuellement venir du même cogniton) permettant d’activer le plan.

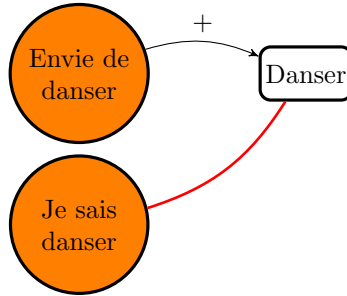


FIGURE 4 – Exemple d’un cogniton renforçant un plan (flèche positive) et d’un deuxième cogniton qui rend le plan accessible (trait rouge).

Les cognitons peuvent avoir un ”trigger”. Il s’agit d’un dispositif permettant d’apparaître ou de disparaître le cogniton en question selon la valeur d’un attribut. Dans le cas contraire, les cognitons sont soit des ”Starting cognitons”, c’est à dire des cognitons présent dès le début de la simulation, soit des cognitons qui n’apparaissent qu’en réponse d’un plan.

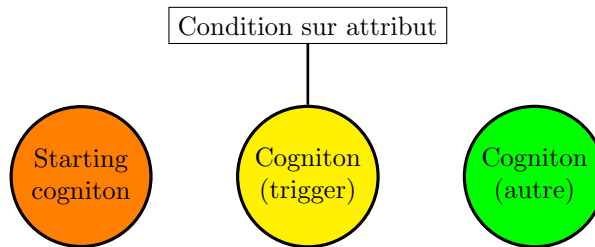


FIGURE 5 – Exemple des différents types de cognitons : Starting cogniton (en orange), cogniton activé par trigger (en jaune) et les cognitons ajoutés par des plans (en vert).

1.3.2 Les plans et les actions

Un plan est un ensemble de plusieurs actions génériques qui peuvent être partagées par un ou plusieurs plans. Une action peut prendre en entrée des paramètres (groupe, rôle, constante, attribut, nombre, aménagement, objet, etc...). Il existe deux types d'actions :

- Les actions qui modifient les paramètres d'un agent (attributs, position, inventaire, cognitons) ou l'environnement (récolte, construction de routes, etc...). Le nom d'une telle action est de la forme : `A_nomDeLAction`.
- Les actions conditionnelles sont généralement composées de deux actions internes. Elles exécutent la première action si la condition passée en paramètre est satisfaite et la seconde action sinon. Le nom d'une telle action est de la forme : `L_nomDeLAction`.

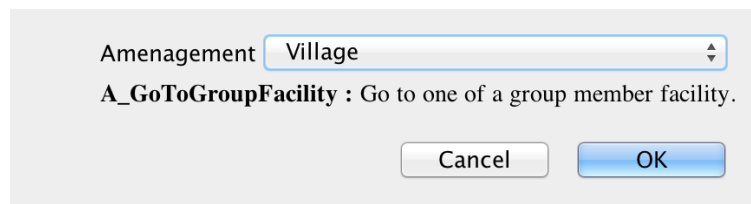


FIGURE 6 – Exemple de l'interface d'édition d'une A_action dans MetaCiv

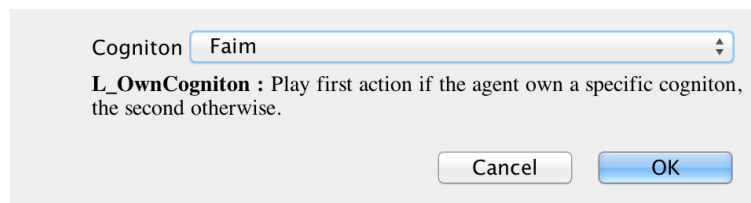


FIGURE 7 – Exemple de l'interface d'édition d'une L_action dans MetaCiv

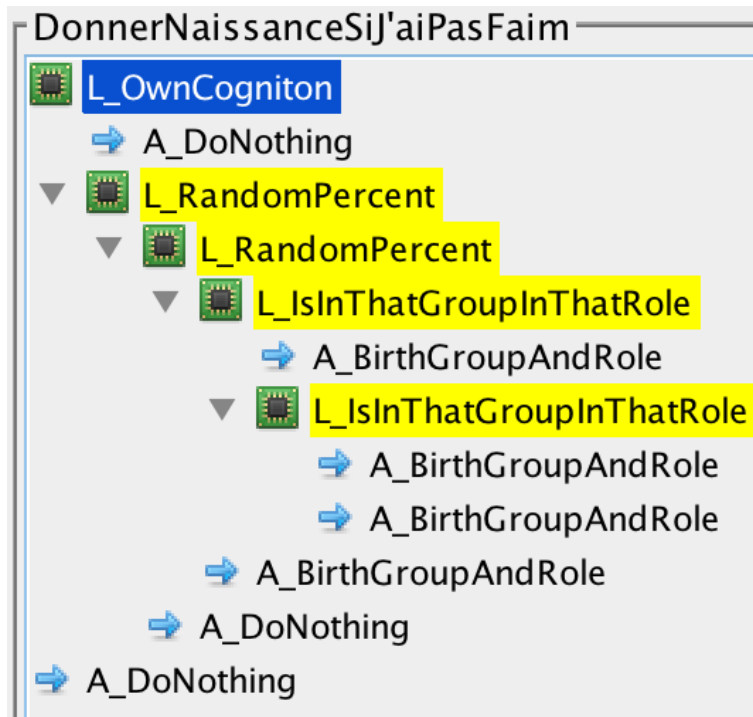


FIGURE 8 – Exemple de l’interface du plan ”DonnerNaissanceSiJ’aiPasFaim” dans MetaCiv. Les actions conditionnelles sont repérées par une icône en forme de processeur et les autres actions par une icône en forme de flèche.

1.3.3 Les attributs

Il existe six attributs dans ce modèle : Vie, Age, CompetenceArtisanat, Energie, CompetenceCueilleur et EsperanceDeVie.

- L’attribut Energie est baissée chaque tour d’une certaine quantité (*BaisseEnergieParTick*), et n’est remontée qu’en mangeant de la nourriture (Baies).
- L’âge d’un individu est augmenté à chaque tick (*AgeParTick*).
- La compétence en artisanat reflète la dextérité d’un individu pour l’artisanat et n’est augmentée qu’après avoir fabriqué divers objets.
- L’attribut ”CompetenceCueilleur” reflète l’aptitude d’un individu pour la recherche et la récolte de baies et n’est augmentée qu’après avoir ramené des baies au village.
- La vie d’un individu est utilisée lors des combats entre deux agents. (Non implémentée pour l’instant).
- L’attribut EsperanceDeVie est initialisée dans le ”BirthPlan”. Ce plan est lancé automatiquement à la création d’un agent et permet l’initialisation de plusieurs paramètres. Cet attribut fixe l’âge à partir duquel l’agent est susceptible de mourir. Afin de donner une espérance de vie à chacun

des agents, nous générons la fonction de masse de la loi de Poisson de paramètre $k = 20$ et $\lambda = 10$.

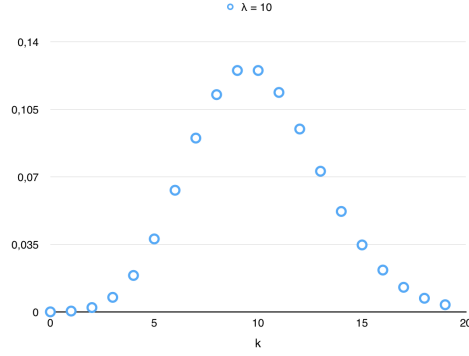


FIGURE 9 – Représentation de la fonction de masse de formule $P(k) = \frac{\lambda^k}{k!}e^{-\lambda}$

On calcule ensuite sa fonction de répartition et on génère une variable suivant cette loi de Poisson en deux étapes :

- On tire un nombre r entre 0 et 1.
- On trouve l'entier i tel que $P(X \leq i - 1) \leq r \leq P(X \leq i)$.

Pour avoir une valeur légèrement plus précise, on trouve le nombre approché i' tel que :

$$i' = \frac{r}{\alpha} - \frac{P(X \leq i)}{\alpha} + i \text{ avec } \alpha = P(X \leq i) - P(X \leq i - 1).$$

Enfin, pour avoir une espérance de vie plus acceptable, nous posons :

$$EsperanceDeVie = 5 * i'$$

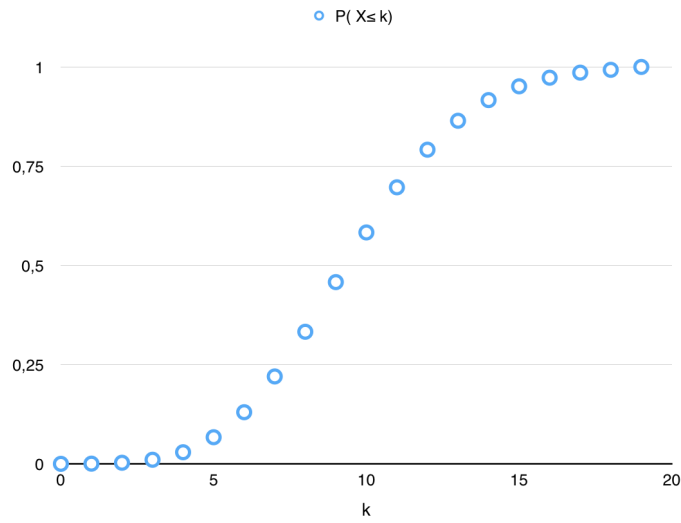


FIGURE 10 – Représentation de la fonction de répartition de la loi de Poisson de paramètre $\lambda = 10$ et $k = 20$.

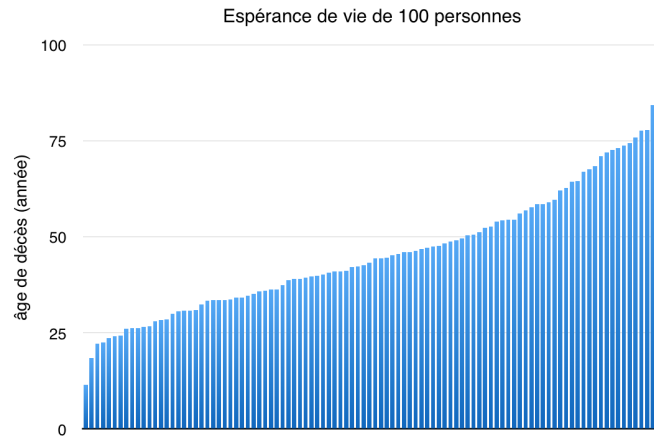


FIGURE 11 – Représentation triée de la valeur de l'attribut "EspéranceDeVie" de 100 agents.

Ainsi, sur les cents agents, trois ont un attribut "EspéranceDeVie" inférieur ou égal à 20, vingt-trois agents ont un attribut "EspéranceDeVie" supérieur ou égal à 60 et le reste des soixante-quatorze agents ont une "EspéranceDeVie" comprise entre 20 et 60.

Attribut	Valeur par défaut
Vie	50
Age	0
CompétenceArtisanat	0
CompétenceCueilleur	0
Energie	100
EsperanceDeVie	Valeur initialisé par le "BirthPlan".

FIGURE 12 – Tableau récapitulatif des attributs d'un agent et de leurs valeurs par défauts.

1.3.4 Les plans par défaut

Lorsque les agents arrivent dans l'environnement, ils possèdent plusieurs plans par défauts. Il en existe quatre : Birthplan, Standard, Ne rien faire et DonnerNaissanceSiJ'aiPasFaim.

- Le plan "Standard" est effectué par chaque agent à chaque tick et gère tout ce qui relève de la physiologie de l'agent. Elle modifie donc les attributs : âge et Energie. Elle est également responsable des tests pour détruire les agents. On peut également remarquer la présence de la ligne "Créer Un Groupe" dans le plan standard, cette ligne crée un groupe et donne à l'agent courant le rôle de "Cueilleur".
- Le plan "Birthplan" qui est lancé à chaque fois qu'un agent est crée ne sert qu'à initialiser l'attribut "EsperanceDeVie".
- Le plan "Ne rien faire" permet aux agents ayant du bois d'augmenter leurs attributs CompétenceEnArtisanat (avec une certaine probabilité).
- Le plan "DonnerNaissanceSiJ'aiPasFaim" permet à un agent de créer un autre agent (avec une certaine probabilité) s'il n'a pas le cogniton "Faim".

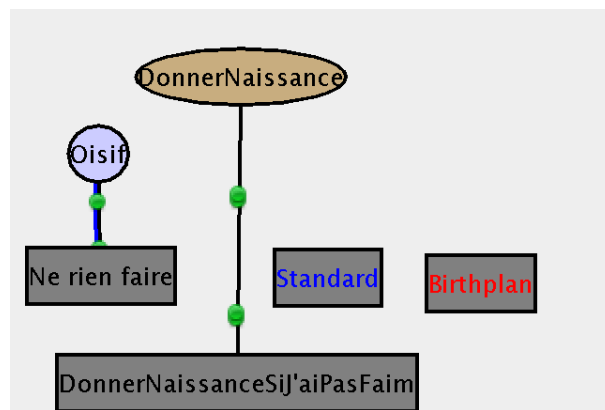


FIGURE 13 – Affichage des plans et cognitons associés dans MetaCiv (les couleurs des cognitons n'indiquent pas le type de cogniton).

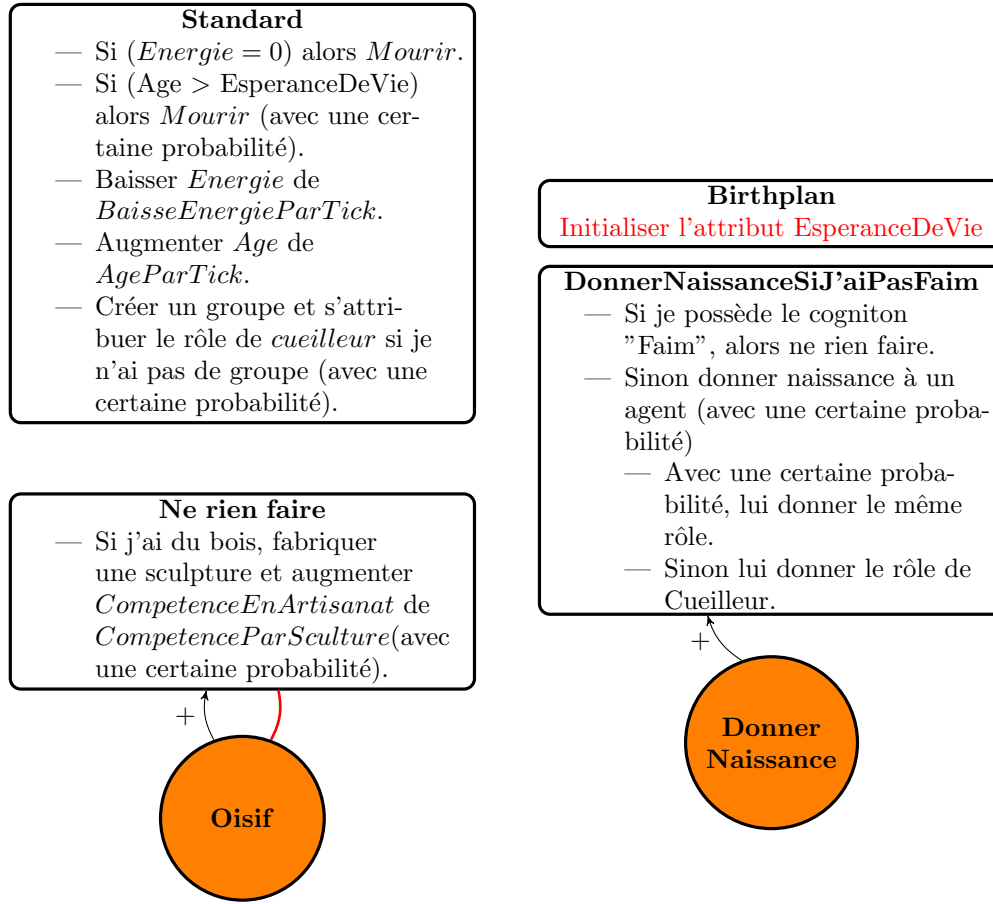


FIGURE 14 – Figure des plans "Ne Rien Faire", "Standard", "DonnerNaissanceSiJ'aiPasFaim" et "Birthplan".

On remarquera que les plans "Birthplan" et "Standard" n'ont pas besoin de cognitons pour être effectués. Le plan "Ne rien faire" est accessible dès le lancement de la simulation puisqu'il est activé par le cogniton "Oisif". Même si le plan "DonnerNaissanceSiJ'aiPasFaim" est influencé, il n'est pas activé.

En plus de ces deux plans, nous avons aussi plusieurs plans qui permettent de gérer la *Faim* mais aussi le rapatriement des *Baies* au village : "Consommer", "AllerChercherAuVillage" et "RamenerBaiesAuVillage".

- Le plan "Consommer" utilise une baie de l'inventaire de l'agent pour remonter ses vies s'il en possède ou augmente le poids du cogniton "FaimEtRienDansMonSac" dans le cas contraire.
- Le plan "AllerChercherAuVillage" ramène l'agent au village du groupe pour récupérer des baies. Si des baies sont présentes dans l'inventaire du village, il baisse le poids du cogniton "FaimEtRienDansMonSac".

- Le plan "RamenerBaiesAuVillage" ramène l'agent au village du groupe afin qu'il y dépose son surplus de baies. Une fois cette opération effectuée, son cogniton "TropDansMonSac" est baissé.

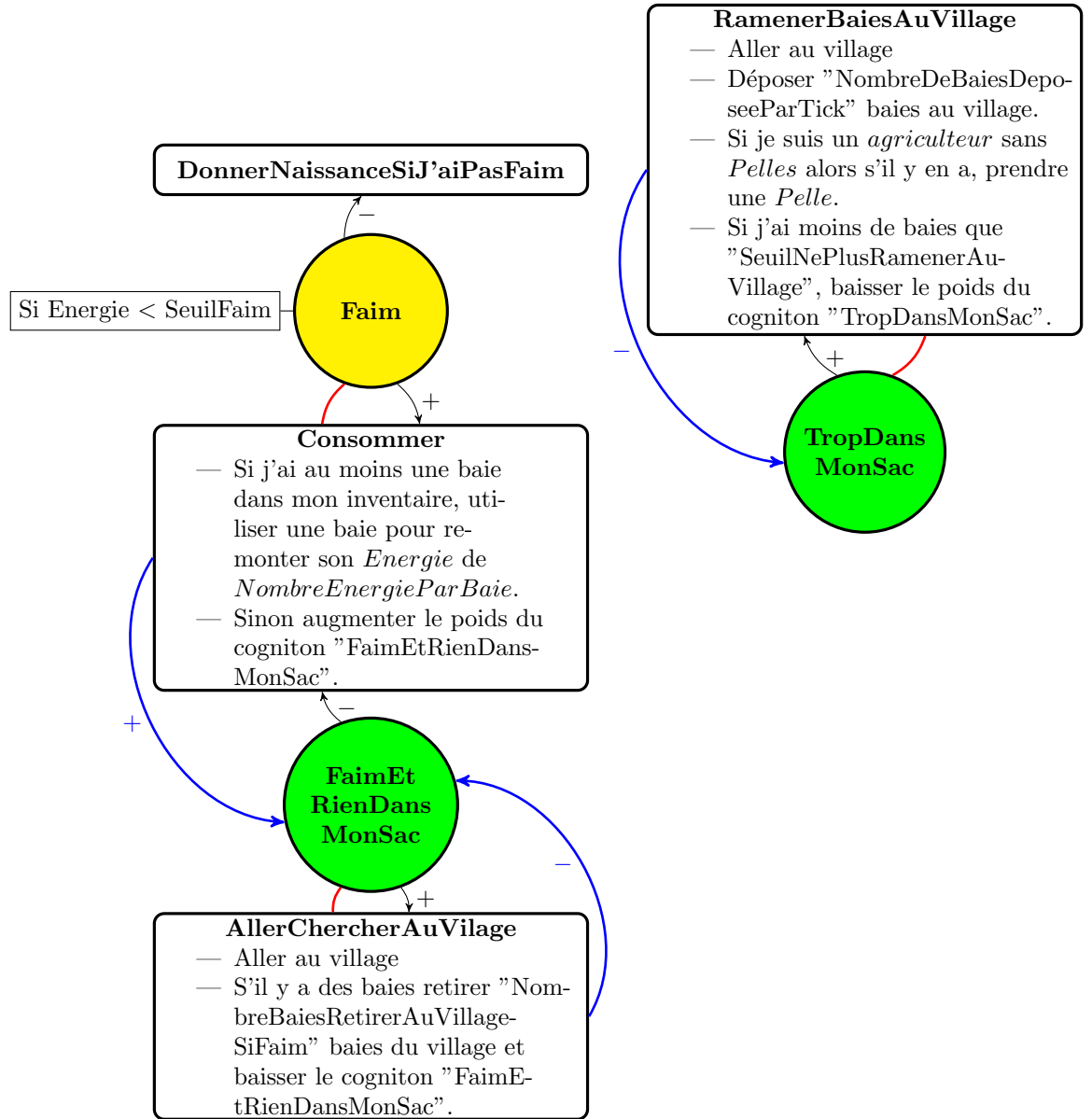


FIGURE 15 – Figure des plans "DonnerNaissanceSiJ'aiPasFaim", "Consommer", "AllerChercherAuVillage" et "RamenerBaiesAuVillage".

On remarquera que le plan "RamenerBaiesAuVillage" permet aux agriculteurs de récupérer des pelles au village (s'il y en a) lorsque ceux-ci viennent déposer des baies.

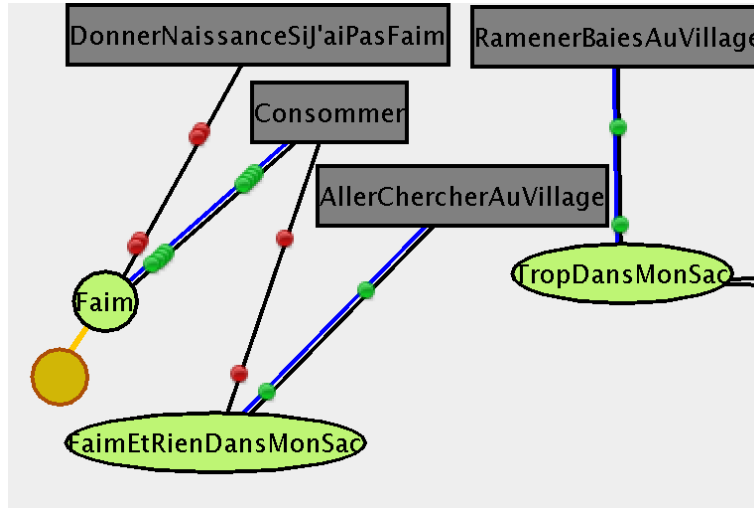


FIGURE 16 – Représentation des cognitons et plans associés dans MetaCiv.

1.4 Les rôles

Lorsque plusieurs agents se rassemblent pour former un groupe, chaque agent de ce groupe possède un rôle. Dans notre modélisation, nous pouvons distinguer trois rôles : *Cueilleur*, *Agriculteur* et *Artisan*. Chacun de ces rôles ajoute des plans supplémentaires au moyen de nouveaux cognitons dits "culturons" et permettent ainsi la spécialisation de l'agent.



FIGURE 17 – Exemple de culturon (en marron).

1.4.1 Le cueilleur

Lorsqu'un groupe est créé, le premier individu de ce groupe est un *cueilleur*. La fonction principale de cet agent est bien sûr de cueillir mais aussi de recruter d'autres agents n'ayant pas de groupe et de construire le village du groupe (s'il

n'y en a pas déjà un). Il possède également la capacité de devenir un *Agriculteur* ou un *Artisan*.

Le cueilleur possède donc cinq plans supplémentaires :

- Le plan "Cueillir" permet aux cueilleurs de partir à la recherche des baies et influence le cogniton "TropDansMonSac" pour le rapatriement des baies. Il est activé par le culturon "AllerChercherDesBaies".
- Le plan "recruter" permet aux agents de former des groupes. Il est activé par le culturon "Recruter".
- Le plan "Construire" permet aux agents de construire leur village, s'ils n'en possèdent pas. Il est activé par le culturon "ConstruireVillage". Une fois le village construit, le cogniton "PasBesoinDeVillage" est ajouté et inhibe le plan "Construire".
- Le plan "DevenirAgriculteur" est activé par le cogniton "DevenirAgriculteur?" mais n'est influencé que par le cogniton "BonCueilleur", lui-même activé que lorsque l'agent a son attribut "CompétenceCueilleur" supérieure ou égale à la constante "SeuilBonCueilleur". Ainsi, un cueilleur n'est susceptible de devenir un agriculteur que lorsque les conditions sont satisfaites.
- Le plan "DevenirArtisan" reprend le principe du plan "DevenirAgriculteur".

On remarquera que le plan "DonnerNaissanceSiJ'aiPasFaim" qui était influencé par le cogniton "DonnerNaissance" n'est activé que maintenant par le culturon "ReproduireCueilleur".

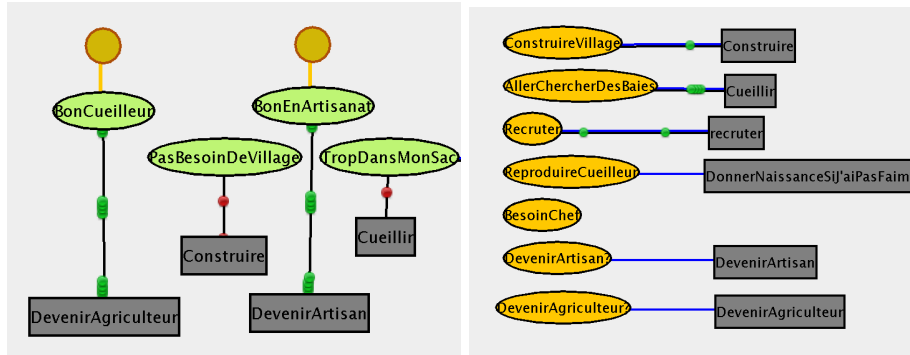


FIGURE 18 – Représentation des plans, culturons et cognitons du rôle cueilleur dans MetaCiv.

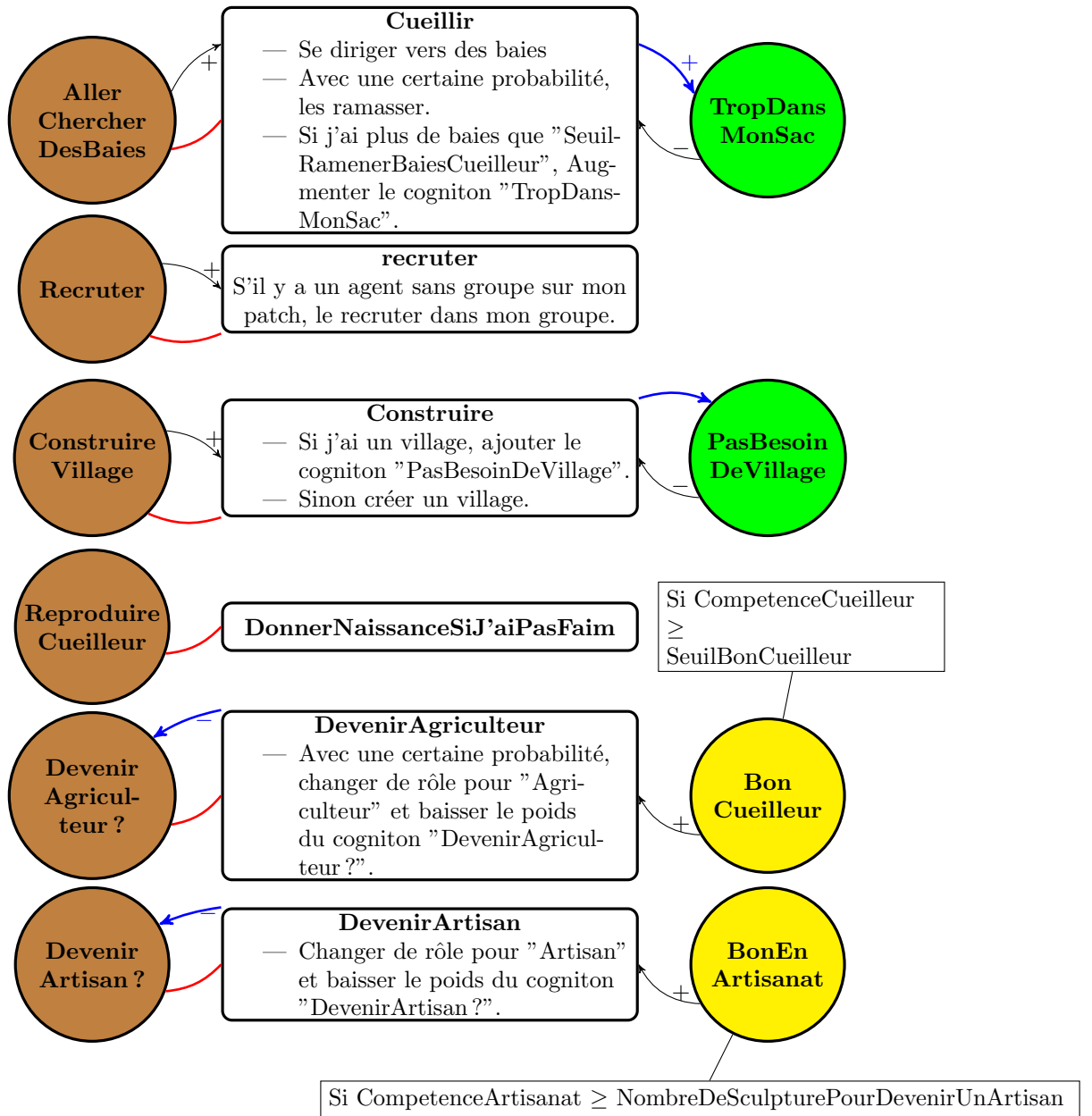


FIGURE 19 – Figure des plans d'un cueilleur et des interactions avec les cognitons associés.

1.4.2 Les artisans

Les artisans sont des agents qui construisent leurs huttes et partent à la recherche de bois pour construire des outils (ici des pelles) pour augmenter la production des agriculteurs. En contre-partie, ceux-ci reçoivent des baies supplémentaires du village. Ils possèdent quatre plans qui leurs sont associés : ChercherBois, RamenerPelleAuVillage, ConstruirePelle et ConstruireHutte.

- Le plan "ChercherBois" consiste à se déplacer vers les patches les plus riches en bois et à transformer cette ressource en objet bois.
- Le plan "RamenerPelleAuVillage" ramène l'agent au village pour y déposer la ou les pelles que l'agent possède. Il retire ensuite un bonus de baies en contre-parti et baisse le cogniton "JaiUnePelle".
- Le plan "ConstruirePelle" fait construire une pelle à l'agent s'il possède au moins NombreDeBoisPourFaireUnePelle unités de bois dans son inventaire. S'il dispose d'une pelle, le cogniton "JaiUnePelle" est augmenté.
- Si l'agent possède une hutte, le cogniton "PasBesoinHutte" est ajouté sinon une hutte est construite sur un emplacement libre.

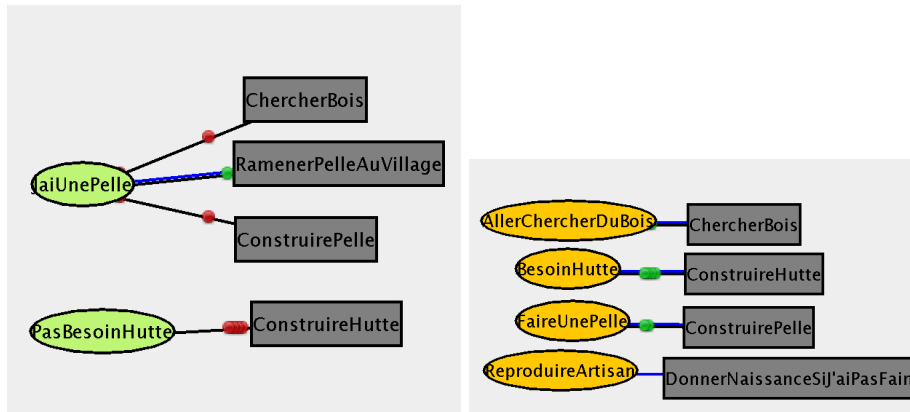


FIGURE 20 – Représentation des plans, culturons et cognitons du rôle artisan dans MetaCiv.

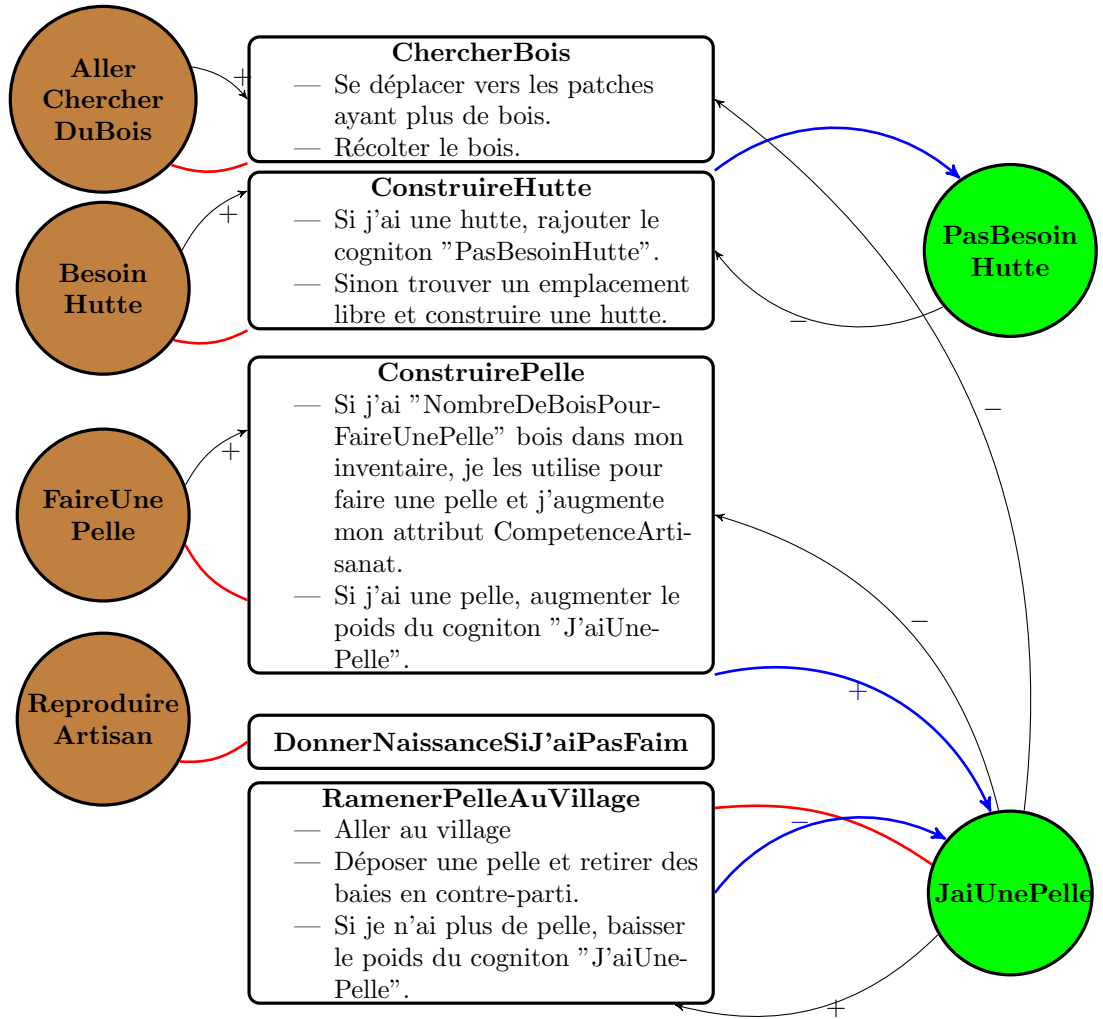


FIGURE 21 – Figure des plans d'un artisan et des interactions avec les cognitons associés.

1.4.3 Les agriculteurs

La seule fonction des agriculteurs est de récolter des baies dans leurs champs ou de construire un champs s'ils n'en possèdent pas. Ils ont donc un seul plan : *Agriculturer*. Si l'agent n'a pas d'aménagement "Champs", il choisit un emplacement libre pour en construire un. Sinon, il se positionne sur son champ pour récolter "NombreDeBaiesRecolteesParTick" baies s'il n'a pas de pelles et "NombreDeBaiesSupplementairesPelles" baies s'il en a une. Lorsque l'agent possède plus de "SeuilRamenerBaiesAgriculteur" baies dans son inventaire, le poids du cogniton "TropDansMonSac" est augmenté.

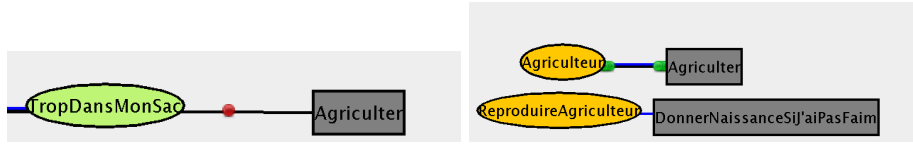


FIGURE 22 – Représentation des plan, culturons et cognitons du rôle agriculteur dans MetaCiv.

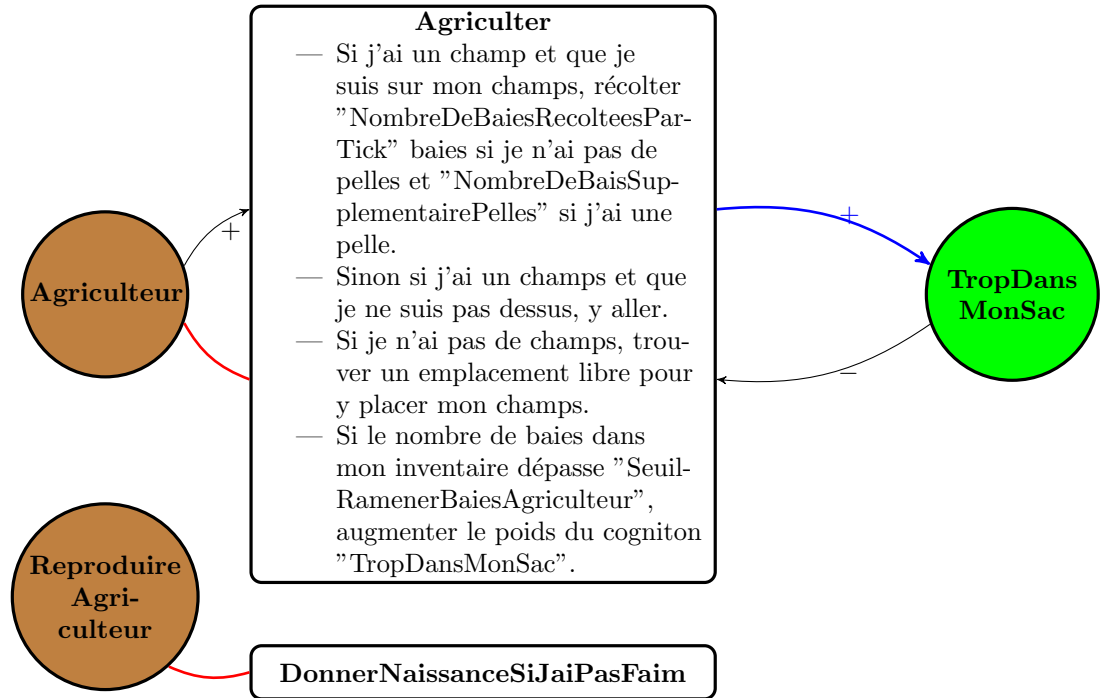


FIGURE 23 – Figure des plans d'un agriculteur et des interactions avec les cognitons associés.

1.5 Les aménagement

Les aménagements sont des marques déposées sur des patches et sont repérés par leurs positions. Ils ont un inventaire qui permet aux agents de déposer ou retirer des objets à l'intérieur. Le tableau suivant récapitule les aménagements de la modélisation.




Aménagement	Apparence	Types d'agents qui utilisent l'aménagement	Utilisation de l'inventaire
Village		Tout types d'agents	Oui
Hutte		Artisans	Non
Champs		Agriculteurs	Non

FIGURE 24 – Tableau récapitulatif des aménagements.

1.6 Les étapes de la modélisation

Au début de la modélisation, les agents sont dispersés autour du point d'apparition. Ils sont immobiles puisqu'ils ne possèdent aucun plan permettant le mouvement (on peut facilement l'ajouter).



FIGURE 25 – Situation initiale

1.6.1 La création des groupes et des villages

Puis, grâce à leurs plans "Standard", certains agents fondent leur groupe en partant à la recherche d'autres agents. Bien sûr, lorsque les agents détectent qu'ils n'ont pas de village, ils en construisent un.

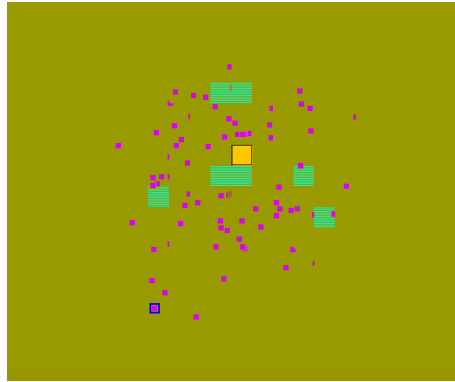


FIGURE 26 – Fondement des groupes et création des villages

1.6.2 La cueillette

Les cueilleurs ayant construit leur village, ils partent alors à la recherche de ressources dans les zones avoisinantes pour remplir le village en ressources. Ceux n’ayant pas réussi à en trouver survivent en se fournissant au village.



FIGURE 27 – La recherche de nourriture

1.6.3 L’apparition de l’agriculture

Lorsque des cueilleurs ramènent des ressources au village, leurs attributs ”CompétenceCueilleur” augmente et à partir d’un certain seuil, ils deviennent des agriculteurs. Leur rôle se réduit alors à récolter des baies dans leurs champs, puis les ramener au village.

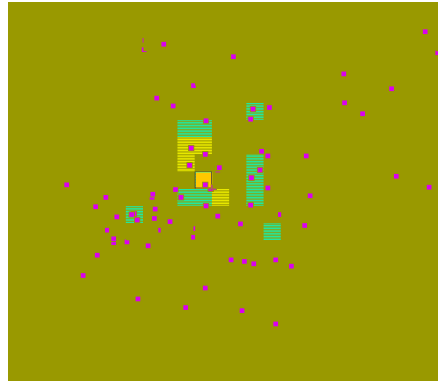


FIGURE 28 – Apparition des premiers champs et agriculteurs

1.6.4 L'apparition des artisans

Certains *Cueilleurs* ont la possibilité de se convertir en *Artisan*. Leur rôle est de récolter du bois et de revenir à leurs huttes pour produire des outils, ici, des pelles. Ils ramènent ensuite ces pelles au villages pour que les agriculteurs puissent s'en servir.

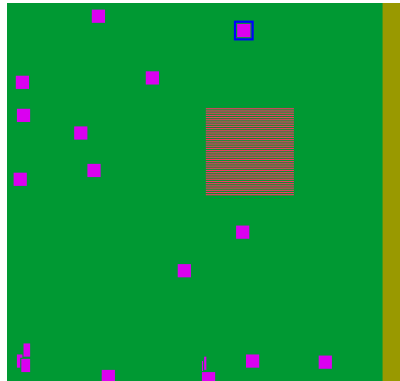


FIGURE 29 – Un artisan (entouré en bleu) proche de sa hutte.

ID	57	Inventaire	
Agent name	Humain-64	Pelle	1
Y	60.0034863108...	Baies	45
Civilization ID	0		
N-GroupRole	1		
Project	Agriculteur		
Total plan weight	7.0		
(Attribute) age	51.5200000000...		
(Attribute) Comp...	0.0		
(Attribute) Energie	50.4000000000...		
(Attribute) Insatisf...	-13.0		
(Attribute) Vie	50.0		
		Patch X	56
agriculteur		Patch Y	60
		Terrain	Sterile

FIGURE 30 – Exemple d'un agriculteur ayant récupéré une pelle au village.

1.7 L'évolution de la population selon les âges.

1.7.1 L'évolution de la population avec la constante "ChancesAvoir-DesEnfants" égale à 0.5

Dans cette partie, la reproduction des agents ne produit que des cueilleurs. On effectue une première simulation qui ne contient que des cueilleurs.

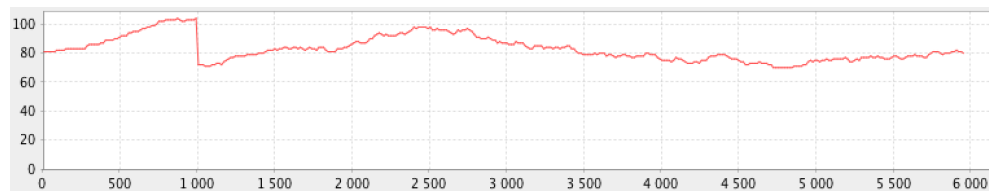


FIGURE 31 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation ne contenant que des cueilleurs.

On rajoute ensuite la possibilité que certains cueilleurs deviennent des artisans.

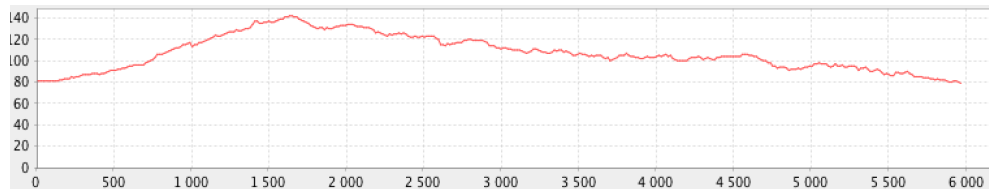


FIGURE 32 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation ne contenant que des cueilleurs et des artisans.

Puis on rajoute la possibilité qu'un cueilleur devienne un agriculteur.

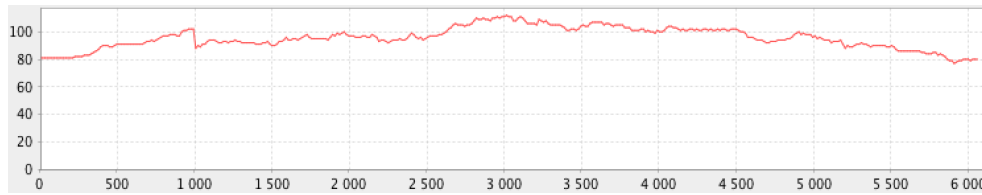


FIGURE 33 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation contenant des cueilleurs, artisans et agriculteurs.

Après observation de plusieurs graphes, on conclut que les graphes précédents ne donnent pas de résultats significatifs car la population est trop peu nombreuse et le taux de reproduction trop faible. Une modification de la constante "ChancesAvoirDesEnfants" pourrait potentiellement résoudre ce problème.

1.7.2 L'évolution de la population avec la constante "ChancesAvoirDesEnfants" égale à 1.

Dans cette partie, la reproduction des agents ne produit que des cueilleurs et la constante "ChancesAvoirDesEnfants" a été fixée à 1. On remarque après plusieurs lancements que cette modification de constante permet d'avoir des graphes plus significatifs et plus homogènes entre chaque simulation. On effectue une première simulation qui ne contient que des cueilleurs.

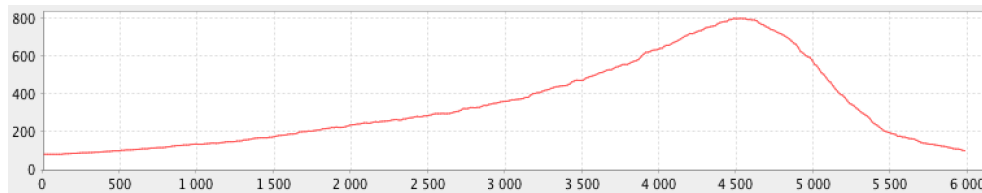


FIGURE 34 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation ne contenant que des cueilleurs.

La population des cueilleurs augmente considérablement par rapport aux simulations précédentes. Elle croît jusqu'au tick 4500 puis décroît. En effet, la population augmentant dans un environnement pauvre en ressource génère l'apparition du cogniton "Faim" dans l'esprit des agents, les empêchant ainsi de se reproduire ce qui explique la baisse de population.

On rajoute ensuite la possibilité que certains cueilleurs deviennent des artisans.

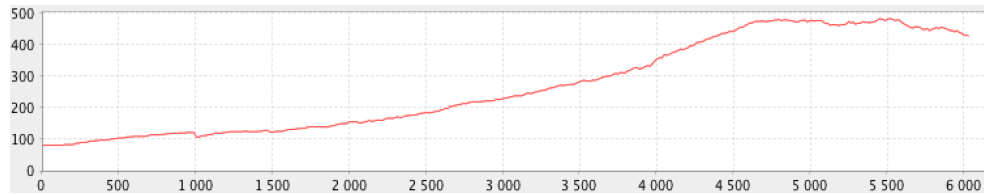


FIGURE 35 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation ne contenant que des cueilleurs et des artisans.

On remarque qu'au lieu de diminuer à partir du tick 4500, la courbe du nombre d'agents par rapport au tick se stabilise. En fait, les artisans ayant ramenés des pelles au villages constituent une réserve de baies qui leurs permettent de survivre même lorsque la nourriture vient à manquer. Même si les autres agents meurent de famine, les artisans ayant encore des baies peuvent se reproduire et créer d'autres cueilleurs puisqu'ils ne possèdent pas du cogniton "Faim". La mortalité des cueilleurs et l'apparition de nouveaux agents explique la stabilisation de la courbe vers le tick 4500.

Puis on rajoute la possibilité qu'un cueilleur devienne un agriculteur.

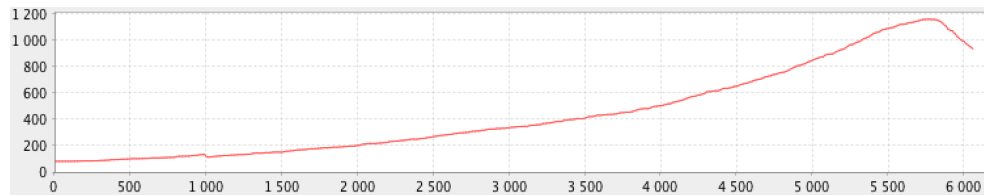


FIGURE 36 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation contenant des cueilleurs, artisans et agriculteurs.

On remarque que la diminution de la population due à la famine a été déplacée vers le tick 5700. Les agriculteurs apportant de la nourriture même lorsque l'environnement est vide permet à la population de survivre plus longtemps. Cependant, une fois que ces agriculteurs meurent, le nombre d'agriculteur n'augmente plus. En effet, la condition pour qu'un cueilleur devienne un agriculteur est qu'il augmente son attribut "CompétenceCueilleur" et cela est difficile lorsque l'environnement est vide.

Une solution serait alors d'introduire le fait qu'un agent ait une probabilité que ses enfants aient le même rôle que lui, cela réglerait hypothétiquement le problème du manque d'agriculteurs.

1.7.3 L'évolution de la population avec la constante "ChancesAvoir-DesEnfants" égale à 1 et l'héritage des rôles.

Dans cette partie, les agents ont une probabilité (ChancesEnfantMemeRole) d'avoir un enfant ayant le même rôle. On effectue une première simulation qui

ne comporte que des cueilleurs.

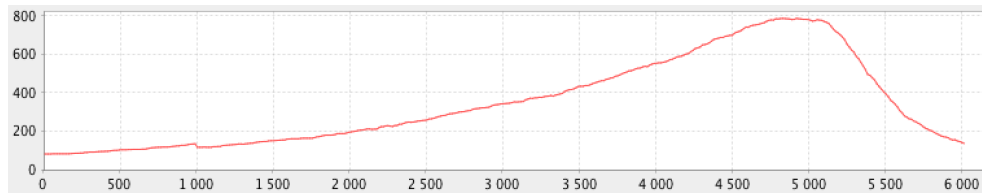


FIGURE 37 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation ne contenant que des cueilleurs.

On ne remarque aucune différence avec la simulation précédente. On rajoute ensuite la possibilité que certains cueilleurs deviennent des artisans.

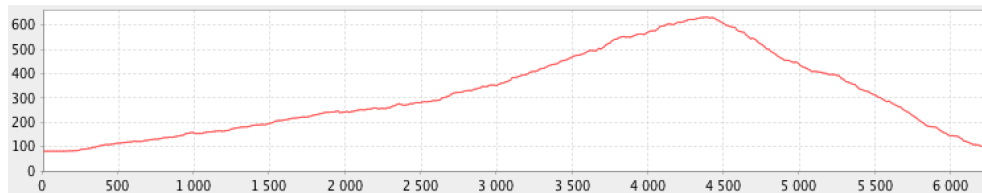


FIGURE 38 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation ne contenant que des cueilleurs et des artisans.

On remarque que ce qui semblait être un équilibre à partir du tick 4500 n'existe plus. Les artisans étant en trop grand nombre, ces derniers n'ont pas de quoi composer leurs réserves de baies. Toute la population est touchée par la famine. Il convient donc de rajouter la possibilité qu'un cueilleur devienne un agriculteur.

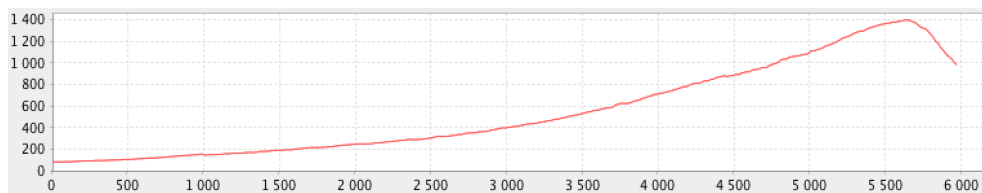


FIGURE 39 – Graphe représentant l'évolution de la population en fonction du temps dans une civilisation contenant des cueilleurs, artisans et agriculteurs

Ici encore, on ne remarque pas de changements majeurs par rapport à la modélisation précédente mise à part une augmentation de la population (1400 au lieu de 1200) mais finalement, le manque d'agriculteur se fait toujours ressentir.

1.8 Impressions sur l'utilisation de MetaCiv

MetaCiv permet de modéliser un grand nombre de situation (interactions entres molécules, sociétés, etc...) si l'on est capable de coder les actions nécessaires et la gestion des plans par des cognitons est une alternative originale à la modélisation des modèles par la subsomption. Voici quelques remarques que nous avons eu tout au long de notre modélisation :

- L'ajout des constantes est un atout majeur pour l'utilisateur facilitant ainsi la modification des paramètres de la simulation. On peut ainsi voir en temps réel l'impact d'un paramètre sur la simulation.
- La sauvegarde automatique des dernières versions fonctionnelles d'un modèle permet d'économiser énormément de temps lorsque l'on travaille sur une modélisation. En effet, il n'est pas rare de changer un paramètre, sauvegarder et de ne plus être capable de lancer son modèle, le rendant alors inutilisable.
- L'interface de gestion des agents est un outil agréable à utiliser, on voit directement les paramètres importants de chaque agent : cognitons, plans, attributs, groupe. Il n'est cependant pas assez efficace si l'on veut observer un type d'agent particulier dans une grande population.
- Il serait intéressant de pouvoir sauvegarder la modélisation à un instant donné et ainsi pouvoir la relancer avec différents paramètres. En effet, lorsque l'on modélise une société avec plusieurs niveaux technologiques, on perd énormément de temps à repasser le début du modèle pour observer l'impact d'un changement.
- La construction d'un plan est très intuitive, on imbrique des actions les unes dans les autres. Les descriptions des actions sont claires et précises. Il n'en reste que construire un plan ou le modifier prend beaucoup de temps puisqu'il faut à chaque fois sélectionner l'action dans un menu déroulant. Une autre interface serait à envisager.

MetaCiv possède de grande possibilités d'évolution, nous sommes conscient que l'ajout de nouvelle fonctionnalité ne doit pas rendre le logiciel plus complexe mais le rendre plus simple et facile d'utilisation pour le grand public.

1.9 Les constantes de la modélisation

SeuilRamenerBaiesCueilleur	50
SeuilRamenerBaiesAgriculteur	50
SeuilNePlusRamenerDePellesAuVillage	0
SeuilNePlusRamenerAuVillage	30
SeuilEnergiePourMourir	0
SeuilConsommationDeBaies	1
SeuilBonCueilleur	5
SeuilFaim	50
NombreEnergieParBaies	8
NombreDeSculpturesPourDevenirUnArtisan	5
NombreDeSculpturesAjoutée	1
NombreDePelleAvantDeRamenerAuVillage	1
NombreDePelleAjoute	1
NombreDeChefParVillage	1
NombreDeBoisParSculpture	1
NombreDeBoisPourFaireUnePelle	5
NombreDeBaiesRecolteesParTick	5
NombreDeBaiesRecolteesAvecPelle	10
NombreDeBaiesDeposeesParTick	10
NombreBaiesRetireesAuVillageSiFaim	6
CompetenceParSculpture	1
CompetenceParPelle	1
ChancesEnfantMemeRole	30
ChancesAvoirDesEnfants	1
BaisseEnergieParTick	-0.1
AgeParTick	0.02

FIGURE 40 – Tableau récapitulatif des constantes de la modélisation.