Projet de MOSIMA
Marché du travail simplifié - Courbe de Beveridge

ARBERET Antonin DUPONT-BOUILLARD Alexandre

Résumé

L'objectif de ce projet est de reproduire le modèle de marché du travail présenté dans l'article. [1]

1 Modèle de base

L'implémentation du modèle de base est contenue dans le fichier `LaborMarketBase.nlogo`. Certaines zones d'ombres laissant place à l'interprétation, ce modèle comprend peut-être quelques légères différences avec celui de l'article.

1.1 Etudier le modèle de base proposé

1.1.1 Agents, attributs, comportements

Agents et leurs attributs Il y a dans le modèle trois types d'agents : les travailleurs nommés workers, les entreprises nommées companies et l'agent de matching.

Workers Les travailleurs ont pour attributs :

- skills : liste de 5 booléens. L'élément à l'indice i est vrai si l'agent a la compétence, faux sinon. Ils sont tirés aléatoirement, uniformément et indépendamment.
- salary : entier qui correspond au salaire attendu par l'agent.
- location : entier qui correspond à leur zone sur la carte.
- mean_productivity : nombre décimal tiré aléatoirement entre 0 et 1 qui correspond à une productivité moyenne. A partir de celle ci chaque jour sera tirée une productivité actuelle (current_productivity) qui déterminera la performance journalière de l'employé.
- current_productivity : productivité au tick actuel de l'agent, tiré aléatoirement (uniformément) dans un intervalle centré sur mean_productivity à chaque tick si l'agent a un employeur.
- company : l'entreprise dans laquelle il travaille, nulle si l'employé n'a pas d'entreprise.

— unexpected_motivation : booléen qui sera mis à jour à chaque pas de temps avec une probabilité commune à tous les employés afin de savoir si l'employé a une motivation particulièrement acharnée pour trouver du travail.
— unexpected_worker_motivation : probabilité qu'un employé ait une soudaine motivation pour se faire embaucher lors d'un tick, et qui augmentera son score de matching. Commun à tous les employés.

A chaque tick, un employé, s'il cherche du travail, attend que l'agent de matching l'associe à une entreprise avec laquelle son score dépasse un seuil. Son score avec l'entreprise peut être augmenté s'il a une motivation soudaine.
S'il a déjà une entreprise, il tire aléatoirement dans l'intervalle [productivité moyenne - $\frac{fluctuation}{2}$, productivité moyenne + $\frac{fluctuation}{2}$] sa productivité journalière, et sera renvoyé de l'entreprise si elle est sous un certain seuil, auquel cas il en sera notifié et s'enregistrera en tant que chercheur d'emploi auprès de l'agent de matching.

Companies Les entreprises ont pour attributs :
— skills : idem que les employés
— location : idem que les employés
— salary : idem que les employés
— worker_list : liste d'employés, de taille 1 dans le cas de l'article
— firing_threshold : entre 0 et 1 est commun à toutes les entreprises, c'est le seuil de productivité sous lequel un employé sera viré, commun à toutes les entreprises
— unexpected_motivation : booléen qui indique si l'entreprise a particulièrement besoin de quelqu'un, commun à toutes les entreprises
— unexpected_firing : probabilité de renvoyer un employé à chaque pas de temps, commun à toutes les entreprises
— unexpected_company_motivation : probabilité qu'une entreprise soit en motivation exceptionnelle à chaque tick, commune à toutes les entreprises

A chaque pas de temps une entreprise attend de trouver un employé si elle n'en a pas. Sinon, elle interroge son employé sur sa productivité actuelle. Si celle-ci est inférieure au seuil, elle le renvoie et envoie un message à l'agent de matching pour indiquer qu'elle en cherche un autre.

Matching agent Le matching agent a pour attributs :
— worker_list : liste de travailleurs en recherche d'emploi
— company_hiring_list : liste d'entreprises qui recherchent des employés
— quality_threshold : seuil au dessus duquel un employé et une entreprise sont compatibles.

— exceptional_matching_bonus : bonus accordé au score si un des agents est exceptionnellement motivé. Le bonus est doublé si les deux agents sont motivés.
— matches_by_round : nombre de couples testés par tick, fixé à 1 ici

L'agent de matching, à chaque pas de temps, forme des couples aléatoires parmi toutes les entreprises et les employés potentiels inscrits dans ses listes. Pour savoir si un couple matche, il va utiliser une fonction d'agrégation des caractéristiques de chacun des agents et vérifier que le score renvoyé par cette fonction est bien supérieur à un seuil, si c'est le cas, l'entreprise embauche le chercheur d'emploi et ils sont supprimés des listes du matching agent.

1.1.2 Formules et algorithmes du modèle

Critère de convergence Nous avons implémenté un critère de convergence modifiable. Les valeurs qui nous importent dans ce modèle sont u = $\frac{U}{L}$ et v = $\frac{V}{L}$, avec L le nombre travailleurs, U le nombre de travailleurs en recherche d'emploi et V le nombre de postes vacants. De cette manière nous pourrons essayer de reproduire la courbe de Beveridge.
Notons ϵ le seuil de convergence et n l'itération actuelle. Le critère de convergence le plus logique est :

$|u_n - u_{n+1}| < \epsilon$ et $|v_n - v_{n+1}| < \epsilon$.

Cependant ce critère souffre de la stochasticité du modèle et peut être vérifié bien avant la convergence. Nous l'avons donc généralisé au suivant : soit k un entier pair représentant le nombre de valeurs considérées pour la convergence. Nous utilisons le critère ci-dessous, qui compare les $\frac{k}{2}$ dernières valeurs aux $\frac{k}{2}$ suivantes :

$|\sum_{i=n-k}^{n-\frac{k}{2}} u_i - \sum_{j=1+n-\frac{k}{2}}^{n} u_j| < \epsilon$ et $|\sum_{i=n-k}^{n-\frac{k}{2}} v_i - \sum_{j=1+n-\frac{k}{2}}^{n} v_j| < \epsilon$.

Ici, nous avons fixé k à 50.

Score du matching calculate_score(a,c) est la fonction qui retourne le score de matching entre un employé a et une entreprise c. C'est une moyenne des 3 scores suivants :

— $S_{skills} = \frac{1}{5}\sum_{i=1}^{5} 1 - |a.skills[i] - c.skills[i]|$
Ce score représente le nombre de skills requis par le poste proposé que l'agent possède ainsi que les skills inutiles que l'agent ne possède pas. Cela traduit le fait qu'on ne cherche pas un employé sous-qualifié, mais pas sur-qualifié non plus. Il est à 0 si aucun skill ne correspond, à 1 s'ils correspondent tous.

— $S_{loc} = \begin{cases} 1, & \text{si } a.location = c.location \\ 0, & \text{sinon} \end{cases}$
Ce score représente la proximité géographique de l'employeur et de l'entreprise et se doit d'être binaire dans le modèle de base.

— $S_{sal} = min(1, 1 - \frac{a.salary - c.salary}{max(a.salary, c.salary)})$
Ce score représente la satisfaction des deux agents par rapport aux salaires attendus et proposés. Si le salaire offert par une entreprise est supérieur ou égal au salaire attendu par un employé, le score est bien de 1, sinon il est inférieur à 1. On notera que ce terme est bien entre 0 et 1.

Le score final S est donc :
$S = \frac{1}{3}(S_{skills} + S_{loc} + S_{sal})$
Il varie entre 0 et 1, 0 représente la plus mauvaise compatibilité entre une entreprise et un employé alors que 1 est le matching parfait.

Pseudo-code Voici le pseudo-code décrivant notre simulation. On note U([x,y]) la fonction qui renvoie un tirage uniforme dans l'intervalle [x,y] et U({x,...,y}) celle qui renvoie un tirage équiprobable parmi les éléments de l'ensemble. La boucle principale, itérée a chaque tick est la suivante :

Algorithm 1 Initialisation()

```
the_matching_agent = matching_agent
the_matching_agent.worker_list = [ ]
the_matching_agent.company_hiring_list = [ ]
for i in [1,U_0] do
    w = worker
    w.skills = [U({0,1}),U({0,1}),U({0,1}),U({0,1}),U({0,1})]
    w.location = U[locations]
    w. salary = min(salary_min, U([salary_mean - max_salary_difference / 2, salary_mean +
    max_salary_difference / 2]))
    w.employer = -1
    mathching_agent.woker_list.add(w)
end for
for i in [1,V0] do
    c = company
    c.skills = [U({0,1}),U({0,1}),U({0,1}),U({0,1}),U({0,1})]
    c.location = U[locations]
    c. salary = min(salary_min, U([salary_mean - max_salary_difference / 2, salary_mean +
    max_salary_difference / 2]))
    c.employee = -1
    mathching_agent.company_hiring_list.add(c)
end for
```

Algorithm 2 Boucle principale

```
Initialisation()
while not convergence do
    for w in workers do
        w.work()
    end for
    for c in company do
        c.update()
    end for
    matching_agent.match()
end while
STOP
```

La fonction work décrit l'action quotidienne de chaque travailleur :

Algorithm 3 worker.work()

```
if company != null then
  w.current_productivity = U([ w.mean_productivity - fluctuation/2, w.mean_productivity + fluctuaction/2])
else
  x = U([0,1])
  if x ⩽ unexpected_worker_motivation then
    w.unexpected_motivation = True
  else
    w.unexpected_motivation = False
  end if
end if
```

La fonction update décrit l'action quotidienne de chaque entreprise :

Algorithm 4 c.update()

```
if employe = null then
  unexp_motiv = U([0,1])
  if unexp_motiv ⩽ unexpected_company_motivation then
    c.unexpected_motivation = true
  else
    c.unexpected_motivation = false
  end if
else
  unexp_fire = U([0,1])
  if (unexp_fire ⩽ unexpected_firing) or (employe.current_productivity ⩽ firing_treshold) then
    c.employee.employer = -1
    mathching_agent.woker_list.add(c.employee)
    c.employee = -1
    matching_agent.company_hiring_list.add(c)
  end if
end if
```

La fonction match décrit l'action quotidienne du matching agent :

Algorithm 5 match()

```
a = random_in_woker_list
c = random_in_company_hiring_list
score = calculate_score(a,c)
if a.unexpected_motivation then
  score = score + exepctional_matching_bonus
end if
if c.unexpected_motivation then
  score = score + exepctional_matching_bonus
end if
if score ⩾ quality_treshold then
  c.employee.employer = c
  woker_list.remove(a)
  c.employee = a
  company_hiring_list.remove(c)
end if
```

1.2 Programmer le modèle

1.2.1 Les paramètres de la simulation

Les paramètres de notre modèle comprennent ceux qui étaient explicitement définis dans l'article ainsi que d'autres que nous avons dû ajouter pour NetLogo ou par interprétation du modèle.

Paramètres globaux Paramètres liés à l'environnement et à la simulation globalement :

- U_init et respectivement V_init : nombre de personnes sans emploi et respectivement postes vacants à l'initialisation.
- salary_mean : salaire moyen.
- max_salary_difference : différence maximale entre deux salaires.
- epsilon : seuil du critère de convergence.
- zone_on_each_side : nombre de zones géographiques sur chaque axe de l'environnement, donc racine carrée du nombre de zones. Les agents dans la même zone sont considérés comme ayant la même localisation dans le modèle.
- stop_on_conv : booléen, si vrai : la simulation s'arrête à la convergence.
- display_links : booléen, si vrai : des liens sont affichés décrivant l'appartenance d'un travailleur à son entreprise.

Paramètres du matching agent Paramètres liés au matching agent :

- quality_threshold : score minimum pour accepter un matching entre deux agents.

— exceptional_matching_bonus : bonus ajouté au score du matching en cas de motivation exceptionnelle lorsqu'une entreprise et un employé sont mis en contact par l'agent de matching. Ce bonus est cumulable si les deux agents sont en motivation exceptionnelle.
— matches_by_round : nombre de couples testés par tick, fixé à 1 dans nos expériences.

Paramètres des travailleurs Paramètres liés aux travailleurs :
— unexpected_worker_motivation : probabilité qu'un employé ait une motivation soudaine et augmente son score.
— max_fluctuation : taille de l'intervalle dans lequel la production journalière de chaque employé sera tirée (cet intervalle sera centré sur la valeur de productivité moyenne de l'employé).

Paramètres des entreprises Paramètres liés aux entreprises :
— unexpected_company_motivation : probabilité qu'une entreprise ait une motivation soudaine et augmente son score.
— firing_threshold : seuil minimum de productivité en-dessous duquel un employé est renvoyé.
— unexpected_firing : probabilité qu'un agent soit brutalement licencié.

1.2.2 Visualisation en temps réel

L'environnement est affiché sous la forme d'un damier, chaque case correspondant à une zone géographique.
Les entreprises sont représentées par des maisons de couleurs.
Les travailleurs sont les personnages, blancs lorsqu'ils sont en recherche d'emploi, de la couleur de l'entreprise sinon.
Si le paramètre display_links est sur "On" alors un lien coloré est créé entre un employé et son entreprise.
L'état de la simulation est visible dans un petit moniteur au-dessus de l'affichage de l'environnement.

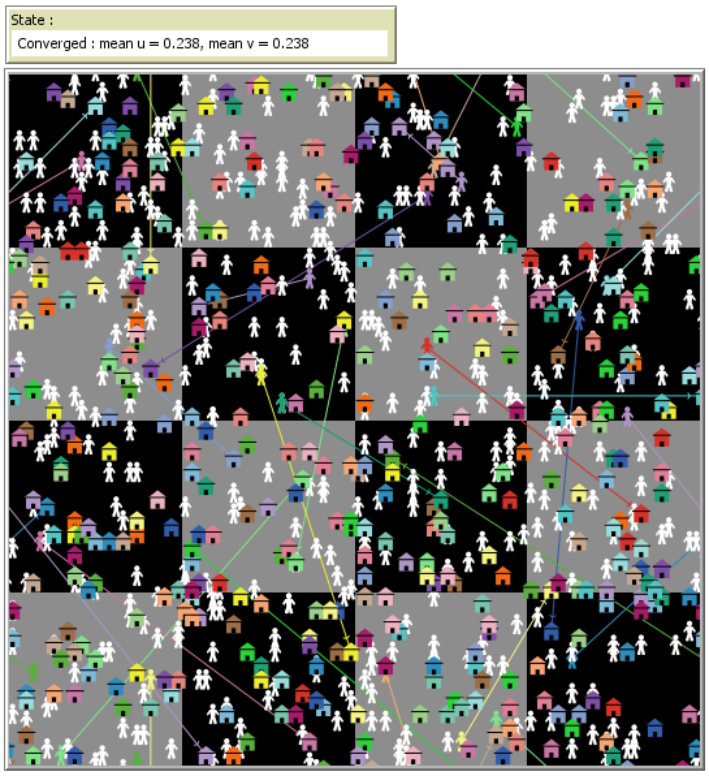


FIGURE 1 – Affichage de l'environnement

Les ratios u et v sont tracés en temps réel sur un graphique, ainsi que les taux d'embauche et de licenciement. Lorsqu'on utilise la simulation pour reproduire la courbe de Beveridge, à la fin de chaque simulation les valeurs de u et v à convergence sont utilisées pour placer un nouveau point sur un graphique dédié.

FIGURE 2 – Affichage des variables en temps réel

1.2.3 Courbe de Beveridge

Pour reproduire la courbe de Beveridge en appuyant sur le bouton"get_beveridge_curve", nous appelons une macro qui exécute successivement plusieurs simulations jusqu'à convergence, en essayant toutes les combinaisons des valeurs de U_0 et V_0 dans $\{100, 200, 300, 400\}$. Les autres paramètres prennent les valeurs données par l'utilisateur via l'interface. Les valeurs de convergence de u et v sont sauvegardées dans une liste et affichées au fur et à mesure des simulations sur le graphique. De plus, lors du setup de chaque simulation, avant de réinitialiser les variables, nous stockons les listes (contenant u et v) dans une variable locale ce qui permet de conserver les valeurs des précédentes simulations qui seraient perdues sinon.

1.3 Reproduction de la courbe de Beveridge

Avec la méthode décrite en 1.2.3, nous obtenons la courbe suivante :

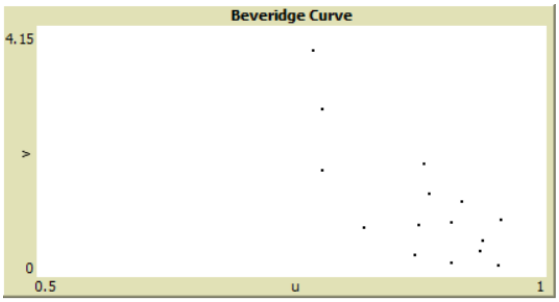


FIGURE 3 – Courbe de Beveridge reproduite

Cette courbe a une asymptote à u = 0.75 environ, ce qui se rapproche des résultats décrits dans l'article. Pour obtenir ces courbes, nous avons utilisé les mêmes valeurs de paramètres que dans l'article, ce qui nous donne les paramètres d'initialisation suivants :

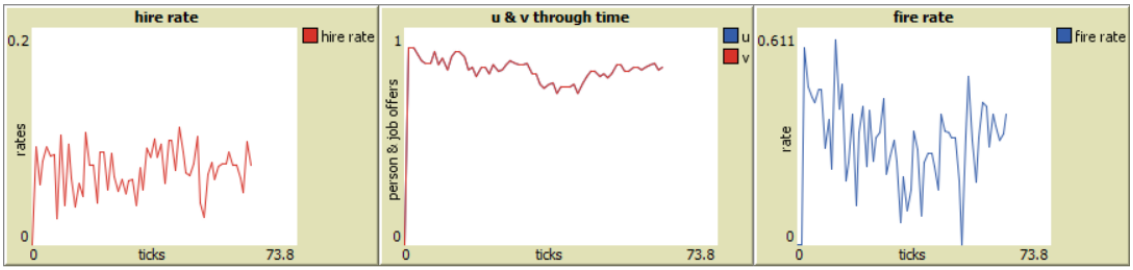
- epsilon = 0.001
- salary_mean = 2000
- max_salary_difference = 400
- zone_on_each_side = 2 (donc 4 zones)
- quality_threshold = 0.5
- exceptional_matching_bonus = 0.1
- matches_by_round = 1
- unexpected_worker_motivation = 0.1
- max_product_fluctuation = 0.3
- unexpected_company_motivation = 0.1
- firing_threshold = 0.5
- unexpected_firing = 0.1

Ces paramètres sont donc les mêmes pour chaque simulation, tandis que U_0 et V_0 ont été choisis dans $\{100, 200, 300, 400\}$. Nous avons donc exécuté 16 simulations, une pour chaque couple possible.

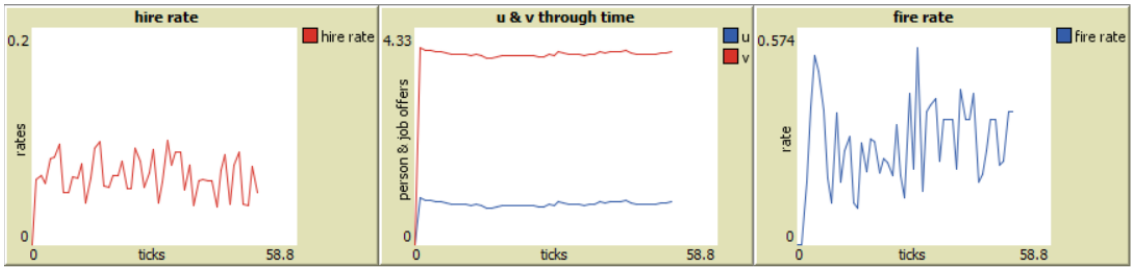
1.4 Étude du modèle avec 4 configurations type

Les simulations avec les paramètres ci-dessous nous ont donné les résultats suivants :

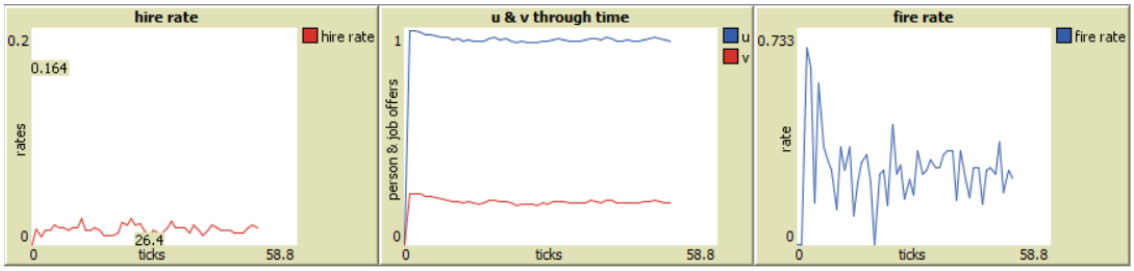
U = 100 et V = 100



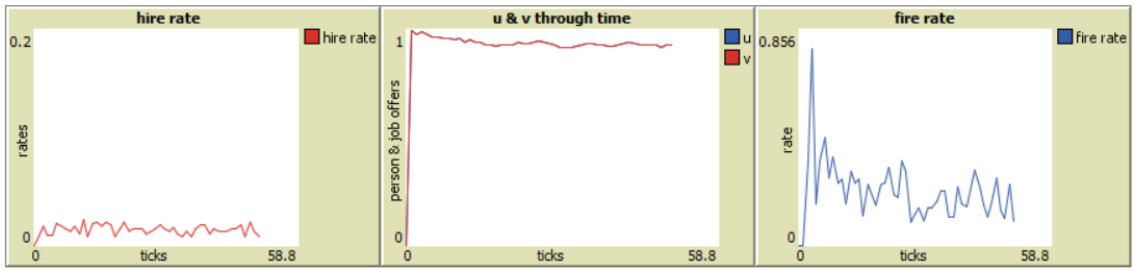
U = 100 et V = 400



U = 400 et V = 100



U = 400 et V = 400



On remarque que u et v varient logiquement de la même manière puisque les nombres de postes et de travailleurs sont fixes.
Dans toutes les configurations types, on retrouve un pic sur u et v correspondant à l'embauche de masse qui suit l'initialisation à la suite de laquelle tous les travailleurs sont sans emploi. Ce pic est suivi d'une convergence, signe que l'environnement tend vers un état dans lequel tous les employés qui pouvaient trouver un emploi stable l'ont trouvé et que le reste de la masse laborieuse oscille entre chômage et emploi de courte durée sans véritablement impacter u et v.
Les fire et hire rates ne convergent pas véritablement puisqu'ils sont calculés indépendamment à chaque itération et subissent directement la stochasticité du modèle.

1.5 Étude de la sensibilité aux paramètres

L'étude de la sensibilité d'un paramètre sur le modèle a été effectuée en ne modifiant qu'un seul paramètre à la fois avant de revenir à la configuration initiale.

- **le nombre de localisations** a un impact important sur le modèle, lors du matching, le score est soit augmenté de $\frac{1}{3}$ soit de 0. On a donc, lorsque le nombre de localisations est faible, beaucoup d'agents aux mêmes endroits ce qui donne de meilleurs scores au matching en moyenne et plus d'embauche.
- **quality_threshold** influe sur l'abscisse de l'asymptote de la courbe de Beveridge, ce qui parait cohérent car une valeur faible augmente la probabilité d'embauche, moins cette valeur est élevée moins le score nécessite d'être élevé pour concrétiser une embauche.
- **exceptional_matching_bonus** parait être un paramètre important du modèle puisque qu'avec des valeurs élevées, autour de 0.5 la valeur de l'asymptote de la courbe de Beveridge semble assez intéressante : elle se trouve à $u < 0.75$, valeur que nous n'avions pas encore obtenue en modifiant les autres paramètres. En effet, il est possible que les motivations exceptionnelles compensent les différences de skills et de localisations de la fonction de scoring.
- **max_product_fluctuation** ne semble pas influer énormément sur la courbe de Beveridge. L valeurs faibles ne laissent aucune chance aux employés non productifs (ils se font licencier dès le premier jour) alors que les autres resteront en poste en permanence. En revanche, des valeurs fortes nuisent aux employés productifs (qui se font licencier a cause de la forte probabilité bien que leur productivité journalière soit faible) tandis que les employés en moyenne peu productifs ont des chances d'être tout de même suffisamment productifs. Il y a probablement une forme d'équilibre qui se crée entre ces deux mécanismes, rendant assez constant l'instabilité de l'emploi.
- **salaire** : le salaire moyen n'a aucune influence contrairement à la différence de salaire maximum, qui, lui, influe sur le score de matching indirectement en permettant (ou pas) que des employés aient des exigences de salaire trop supérieures à ce que leur proposent les entreprises.
- **unexpected_firing** a un impact sur la stabilité du modèle, en particulier lorsqu'il est nul, les agents ne perdent pas leur emploi tant qu'ils ne passent pas sous le seuil de productivité, ce

qui permet à un certain nombre d'entre eux de ne presque jamais changer d'emploi.

— **matches_by_round** n'a pas d'impact si ce n'est de changer la vitesse à laquelle le système évolue.

— **unexpected_company_motivation et unexpected_worker_motivation** influent sur le modèle, puisque des grandes valeurs impliquent une augmentation du taux d'embauche. Cependant ce paramètre doit être combiné avec exceptional_matching_bonus, car si ce-dernier est à 0, les deux premiers n'ont aucun impact.

— **quality_threshold par rapport à firing_threshold** : avec une valeur de quality_threshold inférieure à 50 on note que si firing_threshold est inférieur à quality_threshold, la qualité de notre asymptote est augmentée, on peut traduire cela par le fait qu'un salarié a moins de chance de se faire licencier s'il n'est pas assez productif (il faut donc une valeur de quality_threshold raisonnable sinon il n'y a personne dans les entreprises et donc personne à renvoyer).

2 Extensions du modèle

2.1 Démission des employés

L'implémentation de cette extension est contenue dans le fichier `LaborMarketExtendedQuit.nlogo`.

Nous avons modélisé les démissions symétriquement aux licenciements. Nous avons choisi d'ajouter aux entreprises un paramètre mean_atmosphere entre 0 et 1 qui représente l'ambiance moyenne dans une entreprise. Autour de cette moyenne une ambiance actuelle sera tirée chaque jour (dans un intervalle borné par une fluctuation maximum et centré autour de mean_atmosphere) et si la valeur de l'atmosphère actuelle n'est pas assez élevée (par rapport à une valeur quitting_threshold commune à tout le monde), l'employé quitte l'entreprise.
Cette extension permet aux employés de quitter leur entreprise de la même manière que les entreprises peuvent licencier leurs employés, la productivité moyenne d'un employé correspondant à l'ambiance moyenne au sein d'une entreprise et le seuil minimum de productivité, au seuil minimum de la valeur d'ambiance. Nous avons aussi ajouté une probabilité que les agents démissionnent de manière inattendue, gérée par le paramètre unexpected_quitting.
Le pseudo-code de la fonction work modifiée est donné ci-dessous, ajoutant un test similaire à celui vérifiant la productivité des employés dans la fonction update appelée pour les entreprises. Bien sûr l'initialisation et la fonction update des entreprises sont modifiées aussi avec la même idée.

Algorithm 6 worker.work()

```
if company != null then
  w.current_productivity = U([ w.mean_productivity - fluctuation/2, w.mean_productivity + fluctuaction/2])
  if company.atmosphere ⩽ quitting_treshold then
    quit()
  end if
else
  x = U([0,1])
  if  x ⩽ unexpected_worker_motivation  then
    w.unexpected_motivation = True
  else
    w.unexpected_motivation = False
  end if
end if
```

L'ajout de la démission n'impacte pas la manière dont se déroule la simulation, mais rend l'emploi encore moins stable, augmentant logiquement le chômage. Cela a pour effet de repousser l'asymptote de la courbe de Beveridge sur les abscisses jusqu'à 0.9 avec les mêmes paramètres qu'en 1.3 auxquels s'ajoutent les nouveaux :

- max_atmosphere_fluctuation = 0.3
- quitting_threshold = 0.5
- unexpected_quitting = 0.1

2.2 Quelles sont les limites du modèle

Ce modèle simplifie beaucoup la réalité, il y a donc un certain nombre de choses à préciser. Nous proposons les limites qui suivent :

Les entreprises et les employés ne cherchent pas de meilleures options tant qu'ils sont engagés ensemble. Cela a pour effet que des employés non adaptés à un poste peuvent le garder indéfiniment tant qu'ils produisent suffisamment. La productivité d'un employé n'a d'ailleurs aucun lien avec le fait que ses compétences correspondent au poste.

Le modèle ne représente que des entreprises ayant un seul employé, on pourrait vouloir étendre le modèle en permettant à chaque entreprise de recruter plusieurs personnes avec des compétences communes car l'entreprise est spécialisée dans un domaine, voire même tirer les compétences nécessaires selon une distribution représentative de la diversité des postes dans l'entreprise.

On pourrait envisager que les salaires proposés par une entreprise qui ne trouve pas d'employé augmentent, et aussi que le salaire minimum d'un employé qui ne trouve pas de travail diminue de la même manière. Cela mettrait en avant la compétition sur le marché de l'emploi.

Le critère localisation influe sur la fonction de matching de manière binaire, ce qui n'est pas représentatif, en effet on préfère travailler dans une entreprise dans une ville différente mais proche, que dans une ville très éloignée. On pourrait rendre son influence sur la fonction de matching continue en utilisant une distance réelle.
Le modèle ne permet pas non plus de représenter l'évolution des compétences des individus, un individu qui ne trouve pas de travail peut chercher à se former pour en trouver un qui nécessite un autre domaine d'expertise. Les employés ayant un travail peuvent aussi obtenir des compétences nouvelles au fur et à mesure de leur évolution dans leur entreprise. Il faudrait alors probablement représenter les compétences de façon plus complexe, au risque de se retrouver rapidement avec des agents surqualifiés.
Enfin, dans la mesure où le modèle tient compte de l'aspect géographique, on pourrait simuler la mobilité des individus et des entreprises qui pourraient être amenés à déménager pour diverses raisons, notamment si le marché du travail ne leur correspond pas dans leur région actuelle.

2.3 Nos extensions du modèle

Pour étendre le modèle, nous avons proposé deux extensions en espérant voir émerger un comportement intéressant de la simulation en les utilisant simultanément : la spécialisation des agents et leur mobilité.
L'implémentation de nos extensions est contenue dans le fichier
`LaborMarketExtendedSpecializationAndMoving.nlogo`.

2.3.1 Première extension : spécialisation

La première extension permet aux agents et aux entreprises de se spécialiser autour d'un domaine, en particulier parmi 3 domaines différents. De plus, les entreprises peuvent employer plusieurs personnes, mais toujours avec le même profil.

Compétences La spécialisation se traduit par une modification de la création des compétences, la liste de booléens skills est à présent utilisée comme deux sous-listes :

- Les 3 premiers bits décrivent le domaine de l'agent (entreprise ou employé). L'un d'entre eux est à 1 et les deux autres à 0. Ce bit est tiré aléatoirement.
- Les 2 suivants correspondent à des compétences transversales des agents qui peuvent donc être présentes, ou pas, qu'importe le domaine de l'agent. Les valeurs de ces deux bits sont tirées aléatoirement et indépendamment.

Chaque agent a donc un domaine et entre 0 et 2 compétences transversales. Nous avons donc modifié de manière arbitraire la composante liée aux compétences de la fonction de score pour prendre en compte cette modification :

$$S_{skills} = \sum_{i=3}^{5} \frac{1}{10} 1 - |a.skills[i] - c.skills[i]| + \begin{cases} 0.8, & \text{si } a.domaine = c.domaine \\ 0, & \text{sinon} \end{cases}$$

La composante des compétences est donc calculée à 80% sur le domaine et à 20% sur les compétences transversales.

Distribution des offres d'emplois Nos entreprises peuvent donc embaucher plusieurs personnes en même temps, et chaque entreprise possède un nombre constant de postes qu'on appellera capacité. Cependant, nous voulions des entreprises de taille variable, nous ne voulions donc pas tirer une entreprise pour chaque poste, ce qui aurait donné des capacités pratiquement égales en général.
Nous avons donc ajouté un paramètre nb_company permettant de fixer le nombre d'entreprises. Au moment de l'initialisation on tire nb_company - 1 valeurs dans $]0, V_0[$ en prenant soin de ne pas accepter de doublons. On construit ensuite une liste triée avec ces valeurs auxquelles on ajoute V_0 et 0. Chaque écart entre deux valeurs successives est ensuite affecté à la capacité d'une entreprise, ce qui permet une certaine diversité de tailles en général.

Rendu graphique de l'extension Pour représenter graphiquement cette extension nous avons associé à chaque domaine une couleur : le bleu pour les entreprises du domaine 1 qui sera l'informatique, le rouge pour le domaine 2 qui sera le commerce et le jaune pour les entreprises du domaine 3, le domaine agricole.
Chaque entreprise a ensuite une taille en fonction de sa capacité, et sa couleur varie en intensité : plus la couleur est vive, plus l'entreprise a d'employés.

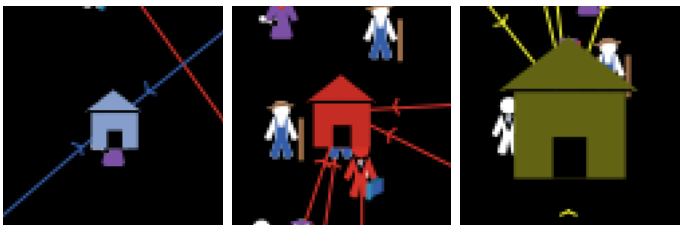


FIGURE 4 – 3 entreprises différentes

Pour les travailleurs, nous avons conservé le même code couleur : un chômeur est blanc, un salarié est de la couleur de son entreprise (sans considérer l'intensité variable). Cependant, un employé peut travailler en dehors de son domaine. Nous avons donc représenté les domaines de spécialité des employés par leurs formes : les jeunes diplômés sont spécialisés en informatique, les porteurs de mallettes dans le commerce et les fermiers dans le secteur agricole.
La figure ci-dessous représente donc un brillant informaticien égaré dans une entreprise de commerce, un commercial tout à fait à sa place ainsi qu'un fermier travaillant pour une entreprise d'informatique :

FIGURE 5 – 3 employés différents

Valeurs observables Pour apporter un retour à l'utilisateur, nous affichons en temps réel le taux de personnes travaillant dans un secteur pour lequel ils ne sont pas spécialisés :

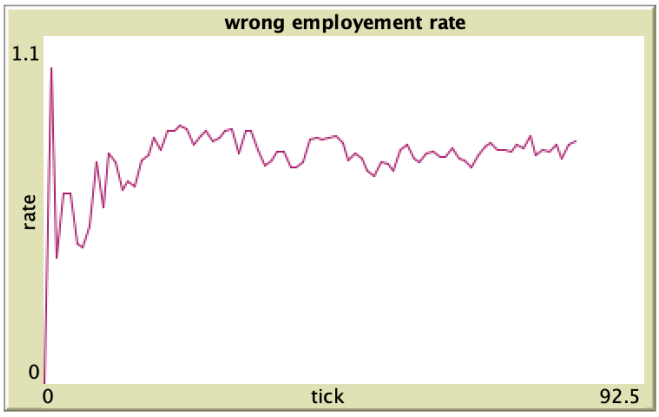


FIGURE 6 – Wrong employement rate

2.3.2 Deuxième extension : mobilité

La seconde extension permet aux agents de déménager notamment si le marché de l'emploi semble inadéquat là où ils sont.

Distance réelle Pour améliorer l'impact de la distance sur la recherche d'emploi, nous avons décidé de modifier la composante liée à la distance de la fonction de score. Au lieu de la fixer à 1 quand la localisation est la même et à 0 sinon, on considère la distance euclidienne entre les agents, et la composante est calculée comme ceci :

$$S_{loc} = \frac{distance\_max - distance\_actuelle}{distance\_max}$$

où distance_max est un paramètre global précisant la distance maximale à laquelle un agent accepte de travailler. Au-delà de cette distance, la fonction de score renvoie 0, quelles que soient toutes les autres composantes. Ce paramètre est introduit pour tenter de forcer la création de clusters au besoin, mais prend en valeur maximale la plus grande distance possible de l'environnement si on veut considérer toutes les combinaisons possibles.

Déménagement Les employés et les entreprises peuvent déménager s'ils ne sont pas satisfaits après un certain nombre de matching dans leur position actuelle. Dans ce cas, ils changent de position pour n'importe quel point tiré uniformément sur la carte.
Les travailleurs déménagent s'ils subissent worker_max_fail échecs consécutifs, qui est un paramètre du modèle et commun à tous les employés.
Les entreprises sont satisfaites à partir d'une proportion min_capacity_prop de leur capacité. Après company_max_fail en-dessous de la proportion désirée l'entreprise déménage. Lorsque qu'elle déménage, elle recalcule les scores avec ses employés. Tous ceux dont le score est au-dessus du seuil d'embauche habituel continuent de travailler pour l'entreprise. Les autres sont licenciés.

Valeurs observables Avec cette extension, l'utilisateur peut visualiser le nombre total de déménagements depuis le début de l'exécution. L'idée étant que la courbe devrait s'adoucir si les agents se stabilisent.

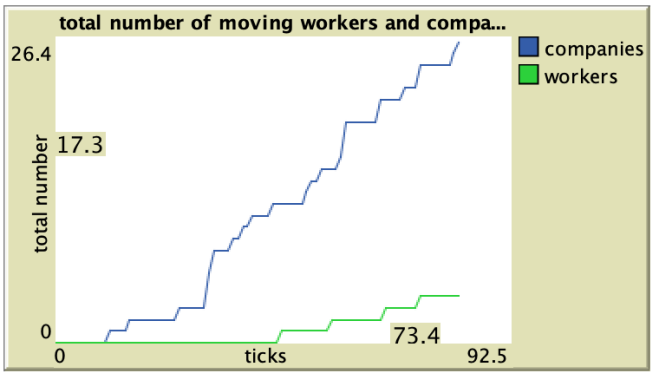


FIGURE 7 – Number of moving

2.3.3 Résultats et analyses

Ces deux extensions ont été pensées pour être exécutées conjointement en espérant l'émergence de cluster géographique autour d'un domaine particulier.
Nous avons notamment obtenu les résultats de la figure 8.

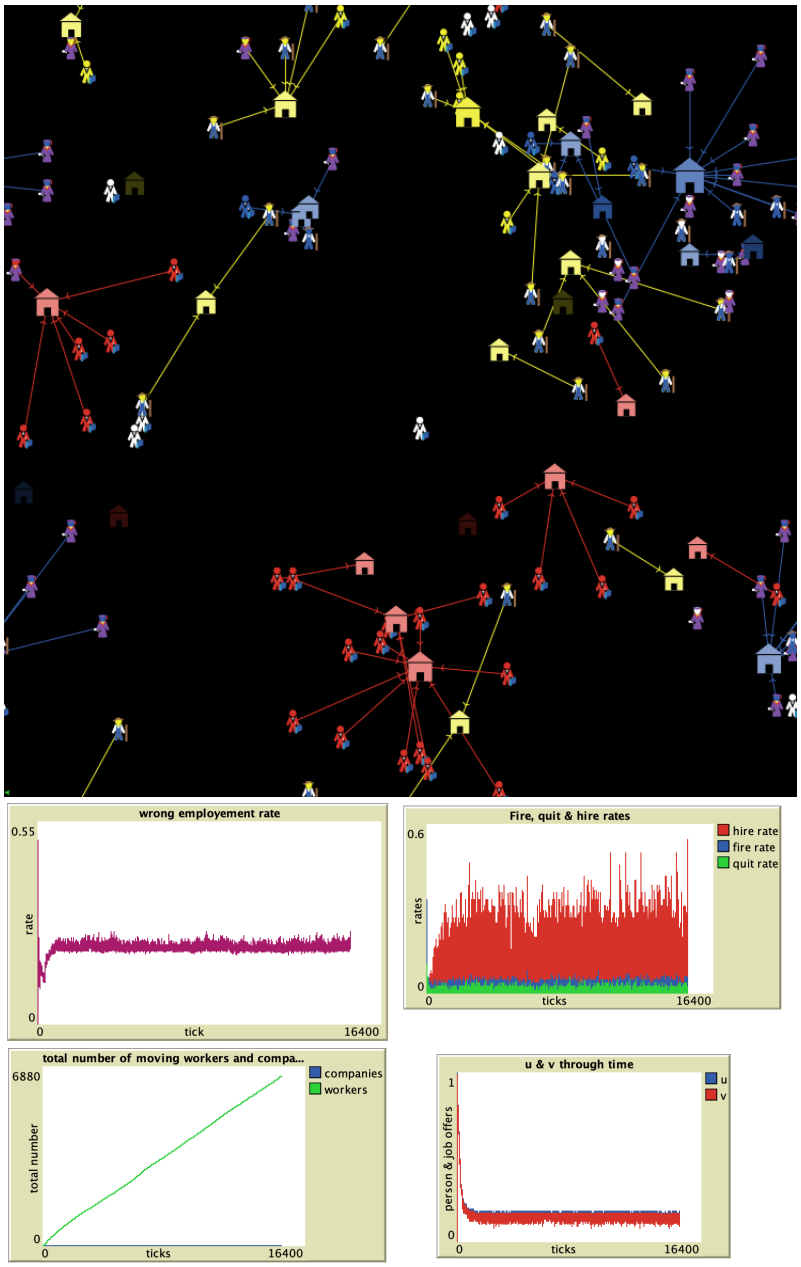


FIGURE 8 – Number of moving

Pour avoir ces résultats, nous avons créé une instance avec une embauche élitiste mais une certaine stabilité de l'emploi pour permettre aux clusters de s'installer. Nous avons aussi utilisé une distance maximum assez faible, et donné un nombre d'échecs acceptable bien plus élevé pour les entreprises que pour les travailleurs (ce qui fausse l'affichage des valeurs sur le graphique qui semble stagner à 0 déménagement pour les entreprises). L'idée étant que les travailleurs déménagent jusqu'à être assez proche d'une entreprise qui leur convient. Certaines entreprises seront assez rapidement satisfaites et resteront en place. Les autres déménageront pour arriver dans une meilleur zone, qui pourrait être

proche d'une entreprise similaire puisque beaucoup de travailleurs du domaine adéquat y sont déjà, et peuvent être licenciés. Voici le jeu de paramètres utilisé :

— U_init = 100
— V_init = 100
— Stop_on_conv = False (notre critère de convergence est inadapté à la lenteur du modèle)
— salary_mean = 2000
— max_salary_difference = 400
— quality_threshold = 0.6
— exceptional_matching_bonus = 0.1
— matches_by_round = 10
— unexpected_worker_motivation = 0.1
— max_product_fluctuation = 0.2
— unexpected_company_motivation = 0.1
— firing_threshold = 0.1
— unexpected_firing = 0
— max_atmosphere_fluctuation = 0.2
— quitting_threshold = 0.1
— unexpected_quitting = 0
— max_dist = 20
— company_max_fail = 2500
— min_capacity_prop = 0.5
— worker_max_fail = 10

On voit assez bien des zones denses se dessiner, centrées autour d'entreprises d'un même domaine. En contraste, certaines parties de la carte sont délaissées. Le taux d'embauche dans un secteur différent de celui de prédilection de l'employé est autour de 25% et on observe un nombre de déménagements qui croit linéairement après une période plus instable dans les premiers ticks.

Cependant, ces résultats ne sont pas obtenus à chaque simulation. Le nombre de paramètres élevé complexifie la recherche d'instances intéressantes.

2.4 Conclusion

Les extensions ont un impact intéressant sur la simulation, mais pas aussi nettement que nous l'espérions. Bien que cela n'apporte pas beaucoup d'explications sur le marché du travail réel, nous pourrions peut-être faire émerger clairement la formation de clusters en automatisant la recherche dans l'espace des paramètres possibles avec un algorithme génétique par exemple, visant à minimiser un critère comme la distance moyenne entre entreprises similaires sur la carte.
Ces extensions peuvent malgré tout être intéressantes, par exemple en modifiant la distribution entre les domaines pour mettre en avant les différences d'offre et de demande entre les domaines d'activité et leur impact sur le marché de l'emploi.

Références

[1] Lewkovicz Zach, Stefanovitch Nicolas et Sommer Christian. "Emergence of the Matching Function in Multi Agent Based Simulations of the Labor Market". In : *LIP6 Workshop* (nov. 2007).