Vote Itéré de Comité

ElectiSim

Rapport de Projet

Par les étudiants

Loic Urien, Victor Racine et Yasmine Hamdani

pour

M. Nicolas Maudet et M. Paolo Vappiani



Projet-ANDROIDE

Master informatique, parcours ANDROIDE

Faculté d’ingénierie - Sorbonne Université

Lundi 21 Mai 2018

[**Etat de l’art**](#_w8bpq8csgxdq) **3**

[Introduction](#_3ujc8d2gh8sn) 3

[Vote itéré de comité](#_lj01ittbhukj) 4

[Règle de vote](#_yv1x1t5fgp9w) 4

[Stratégies des agents](#_lnb6u9576bq3) 6

[Conclusion](#_pgeve3v9mebp) 6

[**Conception et modélisation**](#_u6be5d5myqe0) **7**

[Introduction](#_6ospnhe4r7n) 7

[Composants logiciels](#_1oclpkr7sjn8) 7

[Interface](#_xybqk1gxk7ec) 8

[**Implémentation et Analyse des résultats**](#_4gm5czooq80a) **9**

[Java](#_6k0jk6ehku8c) 9

[Maven](#_6k0jk6ehku8c) 9

[JavaFx](#_6k0jk6ehku8c) 9

[json](#_6k0jk6ehku8c) 9

[csv](#_6k0jk6ehku8c) 9

[Règles de vote et stratégie](#_6k0jk6ehku8c) 9

[Tests et analyse des résultats](#_6k0jk6ehku8c) 9

[Conclusion](#_6k0jk6ehku8c) 9

[**Bibliographie**](#_87eosb7hyklq) **9**

# Etat de l’art

Dans cette partie nous présentons le fonctionnement des votes itérés. Nous commençons par introduire le sujet en section 1.1, puis nous décrivons le processus en 1.2. Deux caractéristiques importantes des votes itérés sont ensuite exposées en 1.3 et 1.4.

## Introduction

Lors d’un choix collectif, il est intéressant de permettre aux agents de modifier leurs votes en fonction de l’évolution des résultats, et leur offrir ainsi la possibilité d’effectuer des choix stratégiques basés sur des compromis afin de tirer partie de la dynamique du processus d’élection et d’augmenter leur satisfaction finale. Ce type d’élection fait ressortir l’influence de l’opinion générale sur le choix d’un votant, comme c’est le cas lors des élections présidentielles où les citoyens se réfèrent aux sondages pour évaluer les chances de leurs favoris et voter en conséquence. Ce système s’avère également utile dans des plateformes telles que Facebook ou Doodle qui permettent aux utilisateurs de modifier leurs réponses à tout moment pour arriver à un consensus.

Les votes itérés relèvent du domaine du choix social computationnel où sont étudiés les problèmes de décisions collectives d’un point de vue computationnel[1\*]. Cet axe de recherche dérive de la théorie du choix social, dont fait partie la théorie du vote et qui s’intéresse à l’étude de l’agrégation de préférences individuelles pour atteindre un choix collectif. L’introduction de l’informatique à ce domaine de recherche est arrivé dans le début des années 2000s et a permis l’élaboration d’algorithmes déterministes et approchés pour traiter les problèmes de choix sociaux et mieux les analyser. Les applications de cette théorie en informatique se sont avérées multiples, notamment dans les systèmes de décisions dans un cadre multi-agents hétérogène ou encore dans les systèmes de recommandation groupées.

## Vote itéré de comité

Au cours d’un vote itéré[1\*], les agents commencent par voter selon leurs préférences individuelles. Un candidat (ou un comité, dans le cas de l’élection d’un comité) est ensuite élu en appliquant la règle de vote spécifiée et les résultats sont communiqués aux agents (en fonction de leur niveau d’information, cela peut aller d’une simple énumération du comité élu jusqu’aux scores détaillés de tous les candidats/comités). Chaque agent applique alors sa stratégie de vote afin d’évaluer ses votes possibles selon les résultats récents, ses préférences initiales et sa fonction d’utilité. Il choisit ensuite, lorsqu’il existe, le vote qui améliore son utilité. Dans le cas où il ne dispose pas de “coup améliorant”, il conserve le même vote que celui fourni à l’instant *t-1*. De nouveau, la règle de vote élit un vainqueur et les agents sont mis au courant et votent en appliquant leurs stratégies, et ainsi de suite. L’élection s’arrête lorsqu’on atteint un équilibre de Nash, c’est à dire qu’aucun agent n’a intérêt à changer de vote.

Par soucis de convergence, la plupart des études sur les votes itérés portent sur des configurations où seul un agent est autorisé à modifier son vote à un instant *t*. Autrement dit, les cas de votes simultanés sont généralement évités [2][3\*][4\*].

Un mécanisme de vote itéré est donc défini par une électionspécifiant la configuration du vote, une règle de vote permettant d’élire le candidat (ou comité) vainqueur à chaque itération, et la stratégie suivie par les agents pour mettre à jour leurs votes.

Une élection *E* est définie[5\*] comme un couple (*A, R*) où *A* représente l’ensemble des candidats et *R,* le profil, énumèreles préférences des votants sous forme de listes ordinales sur les candidats (généralement un ordre total ≻v de l’agent *v* sur les candidats, mais il peut également être partiel dans certains cas).

## Règle de vote

Dans le cas de l’élection d’un comité, unerègle de voteou fonction de choix social[1][2] est une fonction qui prend en entrée une élection *E* et un entier *k* strictement positif *1⩽ k⩽ |A|*, et qui retourne l’ensemble des comités de taille *k* élus ex-aequo. Elle peut avoir recours à des procédés de tie-breaking afin de les départager et de n’en tirer qu’un seul vainqueur. En général, les procédés choisis sont déterministes (par exemple suivant l’ordre lexicographique) afin de réduire les cas de non-convergence[4\*].

Plusieurs règles de votes existent et diffèrent selon l’objectif de l’élection. On se pose alors la question de savoir si par exemple, l’on cherche à élire les meilleurs candidats ou bien à satisfaire la majorité des votants.

Les règles de votes dépendent de la configuration de celui-ci, notamment de la manière dont les agents votent. Il existe trois principaux types de règles de votes de comités[5\*] : les règles attribuant des scores aux comités, les règles basées sur la notion d’approbation et les règles suivant le principe de Condorcet.

* **Règles basées sur les scores des comités :** Elles associent via une fonction d’utilité *f* à chaque comité (une séquence croissante de *k* candidats élus) un score de telle sorte à ce que si un comité *I* domine un comité *J,* alors *f*(*I*) >= *f*(*J*) (on considèrequ’un comité *I=(i1, i2, …, ik)* domine un comité *J=(j1, j2, …, jk)* si pour tout *t* ∈ *{1, 2,.., k} i*t <= *j*t). La plupart se basent sur des adaptations de fonctions d’attribution de scores appliquées pour des élections à un seul gagnant, notamment *t-approval* qui attribue un point aux *t* candidats préférés d’un agent et *Borda* où le score d’un candidat attribué par un agent correspond au nombre de candidats qui viennent après lui dans les préférences de cet agent.

Quelques exemples de fonctions *f* connues :

* **Bloc :** chaque votant révèle ses *k* candidats favoris et les *k*  candidats cités le plus souvent sont élus.
* ***k*-Borda :** élit les comités avec les *k* candidats ayant les meilleurs scores de Borda.
* **Single Non-Transferable Vote (SNTV) :** un comité reçoit un point d’un votant si il contient le candidat préféré de ce votant.
* **Chamberlin-Courant (ß-CC):** chaque votant attribue à un comité un score correspondant au score de Borda du candidat qu’il préfère parmi ceux du comité.

On remarquera que les règles Bloc et K-Borda cherchent à élire un comité composé des *k* candidats préférés tandis que SNTV et Chamberlin-Courant tentent plutôt de sélectionner les candidats supportés par la majorité, comme c’est le cas lors des élections législatives par exemple.

* **Règles basées sur l’approbation :** sont destinées à des élections où les agents communiquent l’ensemble des candidats qu’ils approuvent. Parmi elles on cite :
  + **Approval Voting (AV) :** retourne les comités formés des *k* candidats cités le plus souvent.
  + **Approval-Based Chamberlin-Courant rule :** un comité reçoit le vote (un seul point) d’un agent s‘il contient au moins un candidat qu’il approuve.
  + **Proportional Approval Voting (PAV) :** cherche à maximiser la somme des satisfactions des agents. La satisfaction d’un agent est calculée par la formule , n étant le nombre de candidats du comité approuvés par l’agent.
    - * **Règles basées sur les vainqueurs de Condorcet :** un comité *C* est un comité au sens du Condorcet si la majorité des votants le préfère à chacun des autres comités *D*. Lorsque les votants ont des préférences sur les candidats, il faut trouver un mécanisme pour se ramener à des préférences sur les comités (par exemple dans le modèle k-approval, on considère qu’un votant préfère le comité *C* s’il contient plus de candidats approuvés que le comité *D*. Dans le modèle ordinal, le comité *C* devrait avoir un score de Borda -correspondant à la somme des scores de Borda de ses membres- supérieur à celui de *D*). Parmi les règles :
        + **Number of External Defeats (NED) :** le score d’un comité *S* correspond au nombre de couples de candidats (*c,d*) tels que pour *c* ∈ *S* et *d* ∈ *A\S*, au moins la moitié des votants préfèrent *c* à *d.* Le comité ayant le plus grand score est élu.
        + **Minimum Size of External Opposition (SEO) :** le score d’un comité *S* correspond au plus petit nombre de votants qui préfèrent un des membres du comité *S* à un candidat qui n’en fait pas parti. Le comité ayant le plus grand score est élu.

## Stratégies des agents

Une stratégie de vote[3\*] est une fonction qui, étant données les préférences d’un agent, les résultats de l’élection à l’instant *t-1* et une règle de vote fournit le vote de cet agent à l’instant *t*. Elle dépend fortement du niveau de connaissance des agents qui spécifie les informations concernant les résultats du vote auxquelles ils ont accès. Les alternatives au vote précédent sont évaluées en appliquant une fonction d’utilité sur le comité élu par la règle de vote en lui passant en paramètre le nouveau vote de l’agent et les votes fournis à l’instant *t-1* des autres agents.

En général, on considère qu’à la première itération, les agents votent “honnêtement”, c’est à dire qu’ils ne cherchent pas à tromper les autres agents en communiquant un vote qui ne correspond pas à leurs préférences réelles. Il existe cependant des variantes où ce n’est pas le cas[...].

De plus, les agents sont souvent considérés comme étant myopes[2];ils votent à chaque itération comme si c’était la dernière et cherchent donc à obtenir un résultat immédiat au lieu de planifier une stratégie qui pourrait donner des résultats intéressants sur le long terme. On suppose également que les agents sont rationnels[2] dans le sens où ils cherchent à maximiser leur fonction d’utilité. Ils ne changeront donc de vote à un instant *t* que si cela permet d’accroître strictement cette utilité.

Les stratégies les plus courantes sont les stratégies dites *Better Response* et *Best Response.* Lorsqu’un agent suit la première stratégie, il vote pour le premier comité considéré améliorant sa fonction d’utilité, tandis que la seconde oblige l’agent à explorer tous les votes possibles et à choisir la meilleure alternative. Ces stratégies se basent sur les préférences de l’agent et requièrent un minimum d’informations sur les derniers résultats (ce peut être le classement des candidats ou des comités, leurs scores…). Il existe d’autres stratégies basées des heuristiques (notamment dans les cas des votes à un seul vainqueur[2]) ou sur l’apprentissage des agents [6\*].

## Conclusion

# Conception et modélisation

Dans cette partie, nous exposons l’architecture logicielle choisie en justifiant nos choix.

## Introduction

Une fois le cahier des charges réalisé, nous nous sommes mis à réfléchir à une manière de structurer notre application qui permettrait de répondre aux attentes de nos encadrants. Nous nous sommes d’abord intéressés au fonctionnement des votes itérés (cf état de l’art) afin d’avoir une idée générale des composantes clés de l’application et de leurs interactions. L’application étant destinée à être enrichie plus tard, son extensibilité s’est présentée comme un critère primordial que nous avons fait en sorte de respecter tout au long de notre travail. Ainsi, l’ajout futur de nouvelles configurations (typiquement de nouvelles règles de votes, stratégies mais aussi des niveaux de connaissances différents) devrait se faire sans difficulté.

## Composants logiciels

L’idée était donc de concevoir une base solide composée de plusieurs modules interagissant les uns avec les autres pour configurer, lancer et récupérer les résultats de simulations.

Ainsi, nous avons obtenu une architecture regroupant les modules suivants :

* les différents composants spécifiant le profil de la simulation, la règle de vote, la stratégie et le niveau de connaissance des agents;
* le buffer circulaire contenant les *n* derniers états de la simulation (pour permettre le retour en arrière), *n* étant un paramètre ajustable par l’utilisateur;
* le détecteur de cycles qui analyse le contenu du buffer pour repérer les cycles (une suite d’états où les agents ont voté de la même manière);
* le simulateur qui est le *coeur* de l’application;
* l’exportation et l’importation de configurations (liste des candidats, préférences et stratégie des agents, règle de vote, taille du comité à élire) décrits sous format json;
* l’exportation des résultats (scores et comité élu) à un instant *t* dans un fichier csv*;*
* l’interface qui permet de configurer une simulation (ou d’en charger une), de la lancer et de représenter les résultats sous forme de graphes.

## Interface

La démarche suivie pour réaliser l’interface est la méthode de conception centrée utilisateur étudiée en cours d’Interface Homme-Machine[8]. Nous avons commencé par interviewer les futurs utilisateurs (en l’occurrence, nos encadrants) pour déterminer leurs besoins et établir un cahier des charges. Nous avons ensuite échangé nos idées et avons créé des prototypes papiers que nous leur avons montrés. Nous avons pris en considération leurs remarques afin de raffiner notre prototype et d’en faire une maquette finale qui leur a été présentée..

Le principal critère à respecter était que l’interface soit simple et intuitive. Les clients étant habitués à l’outil informatique, nous avons fait en sorte de proposer un design classique ressemblant à celui des environnements de développements qui leur sont familiers.

# Implémentation et Analyse des résultats

## Java

## Maven

## Export des résultats

## Règles de vote et stratégie

## Tests et analyse des résultats

## Conclusion

intégrer console à l’interface

stratégies

# Bibliographie

[1\*]F. Brandt, V. Conitzer, U. Endriss, J. Lang, et A. D. Procaccia, *Handbook of Computational Social Choice*. Cambridge University Press, 2016, p. 554.

[2]R.Meir, « Iterative Voting » dans *Trends in Computational Social Choice,*  U. Endriss, AI Access, 2017, pp 69-86.

[3\*]A. Loreggia, « Iterative Voting, Control and Sentiment Analysis », Ph.D. dissertation, Dipartimento di Matematica, University of Padova, 2016.

[4\*]R. Meir, M. Polukarov, J. S. Rosenschein, et N. R. Jennings, « Convergence to Equilibria in Plurality Voting », p. 6.

[5\*]P. Faliszewski, P. Skowron, A. Slinko et N. Talmon, « Multiwinner Voting: A New Challenge for Social Choice Theory » dans *Trends in Computational Social Choice,*  U. Endriss, AI Access, 2017, pp 27-48.

[6\*]S. Airiau, U. Grandi, et F. S. Perotto, « Learning Agents for Iterative Voting », in *Algorithmic Decision Theory*, vol. 10576, J. Rothe, Éd. Cham: Springer International Publishing, 2017, p. 139‑152.

[7] O. Lev et J. S. Rosenschein, « Convergence of Iterative Voting », p. 8.

[8] ajouter cours de J. Eagan