Projet 2021

Tom Redon

May 2021

Contents

1 Introduction

(A remplir)

2 Génération des graphes

(Repréciser les pipelines et les prédicteurs, faire des rappels d'Architecture ✓)

Aujourd'hui, la plupart des processeurs sont pipelinés. Cela signifie que les instructions sont séparées en plusieurs étapes et peut ainsi exécuter plusieurs opérations en même temps, sans avoir à attendre la fin de la précédente pour en commencer une nouvelle.
Le processeur est ensuite capable de remettre dans l'ordre les résultats obtenus lors de l'exécution des différentes opérations et d'obtenir le résultat attendu.
Pour éviter de surcharger le pipeline, le processeur essaye de prédire si le saut sera pris lorsqu'il rencontre une expression conditionnelle (un if). Il exécutera ensuite les opérations qui suivent en fonction de sa prédiction et cela permettra de gagner du temps.
Cependant, si le processeur se trompe lors de sa prédiction, les opérations nécessaires pour rattraper l'erreur auront un certain coût.
Pour pouvoir faire ces prédictions, des prédicteurs de branchements ont été développés.
Il existe donc de nombreux types de prédicteurs de branchements.
Il existe par exemple les prédicteurs statiques ou bien encore les prédicteurs dynamiques.
Les prédicteurs statiques ne se servent pas des informations récoltées pendant l'exécution du code précédent. Il aura donc des réponses pré-enregistrées. Par exemple, répondre à tous les branchements qu'ils sont pris.
De ce fait, leur exécution est rapide, mais ils risquent de se tromper régulièrement et donc de commettre de fausses prédictions qui pourraient avoir un certain coût en terme de temps d'exécution.
Le modèle de prédicteur dynamique a donc été établi, et semble plus réaliste.
Il se sert donc des exécutions précédentes du code pour prendre ses décisions et sera donc plus précis, la plupart du temps.
Il existe différentes techniques pour établir des prédicteurs dynamiques.

(A détailler)

Pour représenter ces différents prédicteurs dynamiques, on souhaite étudier et donc générer les graphes à k-états, $k \in \mathbb{N}$, fortement connexes et non-isomorphes entre eux.

Pour produire ces graphes, plusieurs méthodes ont été appliquées pendant l'avancée du projet. La première n'a pas été sélectionnée car elle était largement moins performante que la seconde. Vous pouvez donc continuer votre lecture après l'exemple.

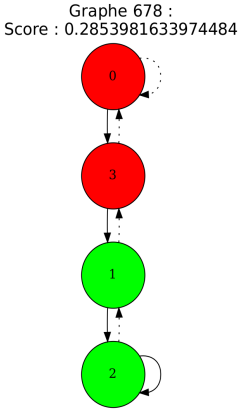
(Repréciser pourquoi une des méthodes est moins bonne que l'autre et bien le préciser en premier lieu ✓)

En premier lieu, tous les graphes à k-états, $k \in \mathbb{N}$, étaient générés à partir de combinaisons établies pour représenter les différentes destinations de chaque arête.
Ainsi, pour un graphe à k-états, on a $2 * k$ arêtes (Une arête "Taken" et une arête "Not Taken" par état).
On doit déterminer, pour chacune, l'état vers lequel elle se dirige.
Pour représenter cela, on générait des combinaisons de taille $2*k$ et dont chaque valeur pouvait appartenir entre 0 et $k-1$.
Ces combinaisons étaient ensuite permutées de toutes les façons possibles pour pouvoir générer tous les graphes à k-états.

Exemple 1 *Pour un graphe à 4-états, on pouvait par exemple avoir la combinaison suivante :*

$$Combinaison = \begin{pmatrix} 0 & 3 & 3 & 2 & 1 & 2 & 0 & 1 \end{pmatrix}$$

On peut donc le lire de la façon suivante :
$\forall n \in [0, k-1], k \in \mathbb{N}, Combinaison[2*n]$ *désigne l'état-destination de la branche "Not Taken".*
$\forall n \in [0, k-1], k \in \mathbb{N}, Combinaison[2*n+1]$ *désigne l'état-destination de la branche "Taken".*
Cette combinaison produira le graphe suivant :



On pourra ensuite permuter la combinaison pour obtenir d'autres graphes.

Cette solution n'étant pas optimale en terme de performance, il a été décidé de générer toutes les combinaisons dont les graphes découlants seraient fortement connexes et n'ayant pas déjà été générés précédemment.
On transforme ensuite ces combinaisons en graphes, puis on vérifie grâce à

l'algorithme de Tarjan que les graphes ne forment bien qu'une seule composante fortement connexe chacun.
(A modifier ✓)
L'algorithme de Tarjan étant un algorithme connu, il ne sera pas détaillé de nouveau ici. Vous pouvez cependant retrouver une description exhaustive de son but et de son implémentation ici : 3.
(A modifier ✓)

(Donner un exemple pour le parcours en profondeur ✓)
(Définir les signatures pour les graphes ✓)
(Préciser les complexités) On effectue par la suite un parcours en profondeur avec étiquetage des chemins, pour vérifier efficacement l'isomorphisme entre les graphes générés.

Exemple 2 *En prenant donc en considération le graphe représenté précédemment, on pourra montrer l'exécution de l'algorithme de la façon suivante :*
On va tout d'abord observer l'état 0.
On créé plusieurs listes qui nous seront utiles pour la suite :
- une liste nommée "marked" qui va représenter les états et s'ils ont déjà été visités ou pas.
- une liste nommée "order" qui est vide pour l'instant, mais qui représentera par la suite l'ordre de rencontre des états en étant parti de l'état choisi au début (ici, 0).
- une liste nommée "étiquette" qui permettra de rédiger l'étiquette marquant le chemin obtenu dans ce graphe au départ de cet état 0. On la comparera aux étiquettes chemins déjà observées dans les autres graphes et on l'intégrera si elle n'y est pas déjà présente. Ensuite on réalise une première exploration du graphe pour marquer l'ordre de rencontre du graphe.
Pour cela, on va tout d'abord marquer l'état comme visité dans la liste "marked", puis on va ajouter ce dernier dans la liste "order". Enfin, on va continuer l'exploration dans les états voisins qui n'ont pas encore été visités. Quand tous les états ont été visités, on arrête l'exploration.
En partant de l'état 0, on obtiendra donc la liste "order" suivante :

$$[0, 3, 1, 2]$$

On effectue donc un second parcours, en utilisant cette liste "order" pour établir l'étiquette.
Pour chaque état dans la liste "order", on va regarder ses voisins dans l'ordre suivant :
En premier lieu, on va observer le voisin auquel on peut accéder par l'arête "Not Taken", puis on observera celui lié par l'arête "Taken". On ajoute ensuite dans la liste "étiquette", la position dans la liste "order" de l'état voisin que l'on observe actuellement.
Ici avec la liste précédemment établie, on obtiendrait la liste "etiquette" suivante :

$$[0, 1, 0, 2, 1, 3, 2, 3]$$

Ce qui représente donc l'étiquette suivante, lorsque la liste est transformée en une chaîne de caractères :

"01021323"

On compare ensuite cette chaîne de caractères avec celles enregistrées précédemment, et si elle n'est pas dans l'ensemble qui nous permet de stocker les chaînes déjà enregistrées, alors on l'ajoute et on passe à l'état suivant dans le même graphe car l'on enregistrera tous les chemins en partant des autres états pour ce graphe. Cependant, si elle est dans l'ensemble, alors on arrête notre parcours ici, car cela signifie que le graphe est isomorphe à un autre graphe précédemment enregistré et on a donc aucun besoin de le réétudier.

Une fois que les graphes sont validés comme étant fortement connexes et non-isomorphes entre eux, on peut les considérer comme des graphes de chaînes de Markov.

3 Chaînes de Markov

Après avoir généré tous les graphes, nous associons des probabilités aux arêtes pour les transformer en chaînes de Markov. Tout d'abord nous devons établir quelques définitions avant de continuer.

Définition 1 *Etant donné une chaîne de Markov défini sur l'ensemble d'états E, alors le nombre* $\mathbb{P}(X_1 = j \mid X_0 = i)$ *est appelé probabilité de transition de l'état i à l'état j en un pas, ou bien probabilité de transition de l'état i à l'état j, s'il n'y a pas d'ambiguïté.*
On note souvent ce nombre $p_{i,j}$ *:* $p_{i,j} = \mathbb{P}(X_1 = j \mid X_0 = i)$.
La famille de nombres $P = (p_{i,j})_{(i,j)\in E^2}$ *est appelée matrice de transition.*

Définition 2 *Le graphe G d'une chaîne de Markov est un graphe orienté défini à partir de l'espace d'états E et de la matrice de transition* $P = (p_{i,j})$ *avec* $(i, j) \in E^2$ *de cette chaîne de Markov :*
- les sommets de G sont les éléments de E,
- les arêtes de G sont les couples $(i, j) \in E^2$ *vérifiant* $p_{i,j} > 0$.

(Faire apparaître p (les probabilités))
(Compléter les définitions pour préciser les matrices spécifiques utilisées)

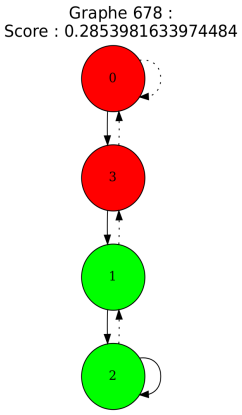
On peut en profiter pour établir la proposition suivante qui nous sera utile lors de l'utilisation de la matrice de transition pour les calculs futurs :

Proposition 1 *La matrice de transition* $P = (p_{i,j})_{(i,j)\in E^2}$ *est stochastique : la somme des termes de n'importe quelle ligne de P donne toujours 1.*

On considérera donc que ces graphes de chaînes de Markov sont représentés par des matrices de transition à l'état initial 0.

(Changer l'ordre avec la proposition)
En considérant la proposition énoncée précédemment et pour être sûr de considérer toutes les valeurs possibles de probabilités $p \in [0, 1]$, les nombres $p_{i,j}$ pourront avoir les valeurs suivantes :
- $p_{i,j} = p$, pour les arêtes "Taken".
- $p_{i,j} = 1 - p$, pour les arêtes "Not Taken".
- $p_{i,j} = 1$, pour les doubles arêtes "Taken" et "Not Taken".
- $p_{i,j} = 0$, pour les arêtes n'existant pas.

En ayant établi ces matrices et les probabilités de transitions possibles, on pourra calculer les probabilités stationnaires π_i pour chaque état $Q \in E$, lié au graphe G.



Exemple 3 *Servons nous donc du graphe ci-dessus, pour donner un exemple de la matrice d'adjacence que l'on obtiendrait :*

$$P = \begin{pmatrix} 1-p & 0 & 0 & p \\ 0 & 0 & p & 1-p \\ 0 & 1-p & p & 0 \\ 1-p & p & 0 & 0 \end{pmatrix}$$

4 Calcul de probabilités stationnaires

Il existe deux moyens de calculer des probabilités stationnaires.
On peut donc soit étudier la probabilité de se retrouver dans chaque état après chaque changement : (Quantifier les X, k et n)
(Revoir la présentation pour la simplifier)

$$X^{(n)} = X^{(n-k)} P^{(k)}$$

Exemple 4 *En reprenant l'exemple ci-dessus, on peut choisir X, telle que :*

$$X^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

On pourra donc calculer les probabilités d'être dans les prochains états à la prochaine prise de branche :

$$\begin{aligned} X^{(1)} &= X^{(0)} P = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1-p & 0 & 0 & p \\ 0 & 0 & p & 1-p \\ 0 & 1-p & p & 0 \\ 1-p & p & 0 & 0 \end{pmatrix} \\ \Leftrightarrow \quad X^{(1)} &= \begin{pmatrix} 1-p & 0 & 0 & p \end{pmatrix} \end{aligned}$$

Ou alors, on peut considérer avec la loi des grands nombres que l'on peut établir tel que :

$$q = \lim_{n \to \infty} X^{(n)}$$

Considérons donc la matrice P (établie dans la section précédente), tel qu'il existe une matrice q qui peut vérifier l'égalité suivante :

$$qP = q$$

Introduisons la matrice identité I, qui nous permettra d'établir l'égalité suivante :

$$\begin{aligned} &\Leftrightarrow \quad qP = qI \\ &\Leftrightarrow \quad q(I - P) = 0 \end{aligned}$$

Exemple 5 *En reprenant l'exemple ci-dessus, on peut établir la matrice q, telle que :*

$$q = \begin{pmatrix} \pi_0 & \pi_1 & \pi_2 & \pi_3 \end{pmatrix}$$

On aura donc :

$$\begin{aligned} q(I-P) &= \begin{pmatrix}\pi_0 & \pi_1 & \pi_2 & \pi_3\end{pmatrix}\left(\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} - \begin{bmatrix}1-p&0&0&p\\0&0&p&1-p\\0&1-p&p&0\\1-p&p&0&0\end{bmatrix}\right) \\ &\Leftrightarrow \begin{pmatrix}\pi_0 & \pi_1 & \pi_2 & \pi_3\end{pmatrix}\begin{pmatrix}p&0&0&-p\\0&1&-p&p-1\\0&p-1&1-p&0\\p-1&-p&0&1\end{pmatrix} \\ &\Leftrightarrow \begin{pmatrix}(-1+p)\pi_3+p\pi_0 & -p\pi_3+\pi_1+(-1+p)\pi_2 & -p\pi_1+(1-p)\pi_2 & \pi_3-p\pi_0+(-1+p)\pi_1\end{pmatrix} \end{aligned}$$

On obtiendra donc le système d'équation suivant :

$$\begin{cases}(-1+p)\pi_3+p\pi_0=0\\-p\pi_3+\pi_1+(-1+p)\pi_2=0\\-p\pi_1+(1-p)\pi_2=0\\\pi_3-p\pi_0+(-1+p)\pi_1=0\\\pi_0+\pi_1+\pi_2+\pi_3=1\end{cases}$$

Ce qui établira les prochaines égalités :

$$\begin{cases}\pi_0=\frac{1-3p+3p^2-p^3}{1-2p+2p^2}\\\pi_1=\frac{p^2-p^3}{1-2p+2p^2}\\\pi_2=\frac{p^3}{1-2p+2p^2}\\\pi_3=\frac{(-1+p)^2p}{1-2p+2p^2}\end{cases}$$

(Faire une transition et repréciser le but des calculs ✓)
On a donc obtenu des valeurs pour chaque état de leur probabilité stationnaire. Cela nous permet donc de savoir la probabilité que j'ai de me retrouver dans l'état pi_i lors des n premiers pas dans la chaîne de Markov.
On pourra donc, en fonction de ces valeurs, facilement déduire si les états doivent être "Taken" ou "Not Taken".
Plus précisément, pour déterminer le statut des états, on appliquera les calculs d'intégrales suivants, pour tous les états π_i du graphe donné :

$$\begin{aligned} x &= \int_0^1 (\pi_i * (1-p))\, dp \\ y &= \int_0^1 (\pi_i * p)\, dp \end{aligned}$$

On choisira la valeur inférieure entre les deux intégrales calculées pour déterminer si l'état est "Taken" ou "Not Taken".

Exemple 6 *En reprenant les calculs effectués précédemment, on obtient :*
Les calculs d'intégrales de π_0 *:*

$$\begin{aligned} x &= \int_0^1 (\frac{1-3p+3p^2-p^3}{1-2p+2p^2} * (1-p))\, dp = 0.27400 \\ y &= \int_0^1 (\frac{1-3p+3p^2-p^3}{1-2p+2p^2} * p)\, dp = 0.08333 \end{aligned}$$

Comme $x \geq y$*, alors l'état est "Not Taken".*
Les calculs d'intégrales de π_1 *:*

$$\begin{aligned} x &= \int_0^1 (\frac{p^2-p^3}{1-2p+2p^2} * (1-p))\, dp = 0.05937 \\ y &= \int_0^1 (\frac{p^2-p^3}{1-2p+2p^2} * p)\, dp = 0.08333 \end{aligned}$$

Comme $x \leq y$*, alors l'état est "Taken".*
Les calculs d'intégrales de π_2 *:*

$$\begin{aligned} x &= \int_0^1 (\frac{p^3}{1-2p+2p^2} * (1-p))\, dp = 0.08333 \\ y &= \int_0^1 (\frac{p^3}{1-2p+2p^2} * p)\, dp = 0.27400 \end{aligned}$$

Comme $x \leq y$*, alors l'état est "Taken".*
Les calculs d'intégrales de π_3 *:*

$$\begin{aligned} x &= \int_0^1 (\frac{(-1+p)^2 p}{1-2p+2p^2} * (1-p))\, dp = 0.08333 \\ y &= \int_0^1 (\frac{(-1+p)^2 p}{1-2p+2p^2} * p)\, dp = 0.05937 \end{aligned}$$

Comme $x \geq y$*, alors l'état est "Not Taken".*

Pour calculer le "score" des graphes, on additionne juste les résultats obtenus.

Exemple 7 *On aura donc, pour ce graphe :*

$$score = 0.08333 + 0.05937 + 0.08333 + 0.05937 = 0.2854$$

Bibliographie 1

https://fr.wikipedia.org/wiki/Cha%C3%AEne_de_Markov, *Visité le 30 Juin 2021*

Bibliographie 2 *https://fr.wikipedia.org/wiki/Graphe_d%27une_cha%C3%AEne_de_Markov_et_classification_des_%C3%A9tats*, *visité le 30 Juin 2021*

Bibliographie 3 *https://fr.wikipedia.org/wiki/Algorithme_de_Tarjan*, *visité le 3 Juillet 2021*