TD “Algorithmique 2” - année 2022-2023

- Auteurs: Dorian MAZAURIC, Eric PASCUAL

- Version: 25/03/2023

Présentation du sujet

Le sujet consiste à développer un service indépendant permettant de calculer le mouvement à jouer dans le cadre d’un jeu de Puissance 4 robotisé permettant à un joueur humain d’affronter la machine.

L’architecture système globale est composée de plusieurs services embarqués (contrôleur matériel, calcul de stratégie, vocalisation,...) communiquant via différence moyens (bus de message, HTTP,...) selon les besoins.

Les détails peuvent être trouvés aux adresses suivantes :

- projet global: https://pobot.org/-CyberP4.html
- volet logiciel: https://pobot.org/Le-logiciel.html

Spécifications globales

Spécifications fonctionnelles

Le service attendu est une API REST exposant la route suivante:
`GET /move?b=<board-content>`

La réponse attentue est au format JSON et contient un entier représentant le numéro de la colonne à jouer, compté à partir de 1. Le *status code* HTTP associé est 200.

Le *query argument* `b` fournit le contenu courant du plateau de jeu, en appliquant les conventions suivantes:

- chaîne de 42 caractères
- construite par balayage en colonne, en partant du coin inférieur gauche
- les caractères la composant sont limités à:
  - `0` : (zéro) cellule vide
  - `h` : jeton du joueur humain
  - `m` : jeton de la machine

A titre d’illustration, la chaîne `m00000h00000mm0000hmh000h00000h00000000000` représente (sauf erreur) la configuration de plateau suivante:

```
. . . . . . .
. . . . . . .
. . . . . . .
. . . h . . .
. . m m . . .
m h m h h h .
```

En cas d’impossibilité de calcul (par exemple si le plateau est déjà plein ou bien s’il contient déjà une position gagnante), la réponse en donne la cause au format suivant :

```
{
    "detail": <message>
```

```
}
```

Si la cause de l'impossibilité est liée à la configuration du plateau fournie, le *status code* de la réponse est 422 et le message précise la cause (ex: "board full", "game over",...)

Dans le cas de données d'entrée invalides, le *status code* est 400 et le message décrit la nature de l'anomalie :

- format invalide (longueur, caractère non autorisé,...)
- configuration invalide (ex: jetons au-dessus de cellules vides)
- situation de jeu invalide : les nombres de jetons doivent respecter la condition `N(humain) = N(machine) + 1`

Contraintes d'exploitation

Ce service est un composant d'un système embarqué s'éxécutant sur une cible matérielle contrainte (typiquement une Raspberry Pi modèle 3 [1]).

Il doit par ailleurs respecter le caractère interactif du dispostif, ce qui implique que le temps de réaction global de la machine doit être de l'ordre de la seconde (2 secondes maximum pour fixer les idées) afin que le joueur ne s'impatiente pa. On pourra considérer le temps consommé par le reste du système comme négligeable par rapport au calcul du coup.

Contraintes d'implémentation

Compte tenu de la nature de l'architecture système utilisée par le dispositif, le choix du langage est à priori libre, pour autant qu'il permette d'implémenter les fonctionnalités attendues.

Afin cependant d'homogénéiser les évaluations, il est fortement suggéré d'utiliser Python, supposé être connu par tout le monde à votre niveau de formation. Tout autre choix devra être soumis à approbation préalable par les évaluateurs, ceux-ci n'étant malheureusement pas omniscients et donc en mesure d'évaluer des soumissions proposées dans n'importe quel langage :([2]

Décomposition du problème

Le problème peut être décomposé en 3 grandes parties :

- calcul du coup à jouer
- modélisation du plateau
- API REST

Le cours portant avant tout sur l'algorithmique, l'API REST sera abordée en dernier et peut être considérée comme facultative. A condition que les deux autres parties aient été correctement traitées, le fait de proposer une solution ne pourra qu'améliorer l'évaluation globale du TD.

Le calcul du coup à jouer peut utiliser n'importe quelle méthode, sachant que la plus fréquemment rencontrée dans ce type de problème est le MinMax.

[1] qui est déjà une configuration confortable dans le domaine des systèmes embarqués :)

[2] navrés de vous décevoir :/

Toute autre approche donnant un résultat évaluable sera considérée avec la même attention.

A noter qu'il n'y a pas de notion chronologique dans la liste ci-dessus, hormis pour l'API. En effet, l'implémentation du calcul du coup à jouer est fortement dépendante du modèle de données, lui-même élaboré en fonction des besoins et contraintes du calcul. Ces deux parties sont donc à traiter à priori en parallèle et non en séquence.

Un focus particulier est donné ci-après sur des points plus précis de la démarche, pouvant être considérés pour certains comme des étapes à suivre et des restitutions à en fournir.

Algorithmique

Calcul du coup à jouer Comme déjà mentionné, une des méthodes couramment utilisées pour cette classe de problème est le MinMax, dont il existe un grand nombre de variantes et d'implémentations disponibles en ligne. Diverses optimations sontégalement populaires, notamment le *alpha-beta pruning* permettant de réduire l'espace exploré en éliminant les options ne pouvant pas améliorer le résultat courant.

Si le recours à une base de code disponible en ligne [3] n'est pas interdit (comment pourrions-nous d'ailleurs l'empêcher ?), il est fortement conseillé de s'en approprier réellement les connaissances et non pas de se contenter d'un copier/coller aveugle (qui ne vous apprendra rien). C'est vous qui voyez, mais ce type d'algorithme de recherche est très utilisé dans de nombreux contextes, et force est de constater que les entretiens d'embauches à des postes techniques sont de plus en plus sélectifs et poussés en termes d'évaluation des connaissances réelles des candidats.

Fonction d'évaluation Dans tous les cas de figure, l'exploration exhaustive n'étant pas une option envisageable pour respecter les contraintes exposées plus haut, il est nécessaire de définir une fonction d'évaluation (souvent appelée *utility function* dans le cas du MinMax) des options de jeu, permettant de les hiérarchiser et de sélectionner celle proposée en retour. Ce type de fonction est au coeur de nombreuses méthodes d'exploration de combinatoires explosives, comme par exemple les algorithmes génétiques.

Ce sujet sera donc le premier à traiter, au moins sur le plan conceptuel, sa restitution étant attendue sous la forme d'une fonction prenant comme argument une configuration du plateau et retournant la valeur d'une métrique supportant une relation d'ordre. Le plus simple est un nombre réel, mais d'autres options sont possibles selon la méthode de classement utilisée.

Un des points majeurs ici est l'heuristique de la définition du calcul de cette métrique, sachant qu'il n'en existe pas qui soit absolue à priori. Sa validation conditionne majoritairement la pertinence du résultat retourné par le processus d'exploration de l'espace des possibilités.

Un soin particulier devra être apporté à l'évaluation des performances de la solution, cette fonction étant appelée un très grand nombre de fois au cours

[3] ou à ChatGPT :(

de l'exploration des options de jeu. Un analyse de complexité des algorithmes utilisés est donc à effectuer.

Le format utilisé en interne pour la configuration du plateau est laissé libre mais il est suggéré de travailler sur plusieurs représentations possibles et d'en comparer les efficacités en termes de temps de traitement. Les stratégies à ce niveau sont très liées aux caractéristiques du langages d'implémentation utilisé.

Modélisaton du plateau

Le format d'entrée de la configuration du jeu, tel que fourni par le client de l'API a peu de chances d'être le plus efficace pour les manipulations que les calculs vont faire avec. Une étude des différentes options possibles en fonction de la nature de ces traitements sera à mener afin d'en évaluer les performances, que ce soit en temps de traitement et en consommation de ressources (mémoire notamment), sachant que ce dernier facteur est intrinsèquement moins limitant que le temps.

La remarque faite ci-dessus concernant le lien avec le langage d'implémentation s'applique ici aussi bien évidemment. Ces choix devront donc être argumentés en conséquence.

Implémentation de l'API

Si vous traitez également cette partie, ne pas perdre de vue qu'un tel service n'est pas exposé publiquement sur Internet (en tout cas dans le scénario considéré). Pas besoin donc de se préoccuper des aspects sécuritaires de mise pour des applications en ligne.

La librairie standard Python contient par ailleurs tout ce qui est nécessaire ici. Inutile par conséquent de s'emcombrer d'un *framework* tiers.

Rendu et notation

Outre les codes sources développés, une courte vidéo d'une durée de 2 à 4 minutes est attendue pour présenter les travaux réalisés, incluant les points suivants :

- description et analyse des éléments algorithmiques
- description du programme
- description de l'organisation mise en place au sein de l'équipe projet (décomposition des tâches, répartition entre les membres, méthodologie de travail collaboratif,. . .)

L'originalité du rendu et les qualités pédagogiques de la vidéo feront partie des critères d'évaluation du rendu du projet. Inutile par contre de passer du temps à réaliser des effets spéciaux à la James Cameron ;)