OPTIMISEUR MULTI-OBJECTIFS PAR APPRENTISSAGE PAR RENFORCEMENT

1. Description générale du programme

Ce programme permet la résolution des problèmes d'optimisation mono et multi-objectifs avec ou sans contraintes, en utilisant des algorithmes d'apprentissage par renforcement tels que A2C, DDPG, PPO, SAC et TD3. Le programme propose également un environnement d'optimisation flexible et un mécanisme pour sélectionner les hyperparamètres appropriés pour chaque algorithme.

2. Configurations compatibles

- Python 3.12 et versions compatibles.
- La version de l'OS (Windows 10, Ubuntu 20, Mac...).

3. Installation du module `optimiseur_rl.py`

3.1. Installer toutes les dépendances nécessaires

Pour intaller toutes les dépendances nécessaires pour le module `optimiseur_rl.py`, il faudra exécuter le fichier `requirements.txt` qui contient les dépendances, numpy, gym, Markdown, matplotlib, stable_baselines3, torch, tqdm, PyYAML et setuptools.

Il faut exécuter la commande `pip install -r requirements.txt` dans le dossier où se trouve `requirements.txt` dans votre terminal (pas la console python).

```
pip install -r requirements.txt
```

Cette commande va installer tous les paquets python qui se trouvent dans le fichier `requirements.txt` à partir de Pypi.

3.2. Installer le module `optimiseur_rl.py`

L'étape suivante consiste maintenant à installer le module `optimiseur_rl.py` à l'aide du fichier `setup.py` (qui fait appel au fichier `setup.cfg`). Pour se faire, vérifier que vous êtes bien dans le

dossier où se trouve le `setup.py`, puis exécuter le `setup.py` à l'aide de l'une des deux commandes suivantes:

```
python setup.py install
```

Avec cette commande, le paquet est installé dans votre environnement virtuel tel qu'il est au moment de l'installation.

```
python setup.py develop
```

Avec cette commande, le paquet est installé dans votre environnement virtuel mais toutes modification au code future sera prise en compte et le paquet installé sera automatiquement mis à jour.

3.3. Désinstaller le module `optimiseur_rl.py`

Pour désinstaller le module `optimiseur_rl.py` de votre environnement virtuel, vous pouvez utiliser la commande :

```
pip uninstall optimiseur_rl
```

4. Instructions pour l'exécution

- Assurez-vous d'avoir installé les dépendances nécessaires spécifiées dans le fichier `requirements.txt`.
- Fournissez les données du problème dans un fichier YAML nommé `deck.yaml` en suivant rigoureusement les instructions consignées dans ledit fichier, tout en respectant la syntaxe. Une fois terminé, veuillez enregistrer vos modifications.
- Exécutez le script `main.py` en utilisant la commande suivante :

```
python main.py
```

- Suivez les instructions affichées dans le terminal et saisissez 'GO' en lettres capitales lorsque vos données YAML sont prêtes.
- Après l'exécution, le programme affichera les résultats de l'optimisation ainsi que les graphiques correspondants.

5. Structure du projet

- `main.py` : Le script principal qui contient la logique principale du programme.
- `optimiseur_rl.py` : Le module contenant les classes et fonctions pour la résolution de problèmes d'optimisation en utilisant l'apprentissage par renforcement.
- `LecteurYAML.py` : Le module contenant la classe pour lire les données à partir d'un fichier YAML.
- `environnement_minimisation.py` : Le module contenant la classe pour créer l'environnement de résolution du problème d'optimisation.
- `reglage_hyperparametres.py` : Le module contenant la fonction pour sélectionner les hyperparamètres en fonction de l'algorithme choisi.
- `visualisation.py` : Le module contenant la classe pour l'affichage graphique des résultats de l'optimisation.

6. Description détaillée des modules

Module Environnement (`environnement_minimisation.py`)

Cet environnement fournit une interface pour résoudre des problèmes de minimisation avec des contraintes. Il utilise la bibliothèque Gym pour définir l'espace d'observation et l'espace d'action, et fournit des méthodes pour évaluer les objectifs et vérifier les contraintes.

Module Réglage des hyperparamètres (`reglage_hyperparametres.py`)

Ce module fournit une fonction pour sélectionner les hyperparamètres appropriés pour chaque algorithme de renforcement en fonction de l'algorithme choisi. Il utilise la bibliothèque Stable-Baselines3 pour fournir des hyperparamètres prédéfinis pour les algorithmes A2C, DDPG, PPO, SAC et TD3.

Module LecteurYAML (`LecteurYAML.py`)

Ce module contient une classe pour lire les données à partir d'un fichier YAML. Il utilise la bibliothèque PyYAML pour charger les données YAML et les retourner sous forme de dictionnaire.

Module Visualisation (`visualisation.py`)

Ce module contient une classe pour l'affichage graphique des résultats de l'optimisation. Il utilise la bibliothèque Matplotlib pour créer des graphiques tels que des graphiques simples, des sous-graphiques et des graphiques en nuage de points.

7. Choix de l'algorithme pour l'optimisation

L'algorithme d'optimisation est choisi en fonction des données spécifiées dans le fichier YAML fourni. L'utilisateur peut sélectionner parmi les algorithmes suivants :

- A2C : Asynchronous Advantage Actor Critic
- DDPG : Deep Deterministic Policy Gradient
- PPO : Proximal Policy Optimization
- SAC : Soft Actor Critic
- TD3 : Twin Delayed DDPG

8. Dépendances

- numpy
- gym
- Markdown
- matplotlib
- stable_baselines3
- torch
- tqdm
- PyYAML
- setuptools

9. Auteurs

Ce programme a été développé par Cédric Foffé Ngoufo Email: cedric.foffe-ngoufo.1@ens.etsmtl.ca

10-. Licence

11. Ressources et Références

Arroyo, J., Manna, C., Spiessens, F., & Helsen, L. (2021, September). An Open-AI gym environment for the building optimization testing (BOPTEST) framework. In *Building Simulation 2021* (Vol. 17, pp. 175-182). IBPSA.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Huang, S., Kanervisto, A., Raffin, A., Wang, W., Ontañón, S., & Dossa, R. F. J. (2022). A2C is a special case of PPO. *arXiv preprint arXiv:2205.09123*.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1-8.

Stable Baselines3 Documentation, Release 2.3.0a3 Mar 04, 2024

Welcome to Stable Baselines docs! - RL Baselines Made Easy — Stable Baselines 2.10.3a0 documentation