

PyClones : Python CLOsed queuing Networks Exact Sampling.

L'objectif de ce mini TP est de présenter l'utilisation du package Python PyClones. Pour ce faire nous prendrons comme exemples le réseau monoclasse G_1 et le réseau à 2 classes (G_1 et G_2).

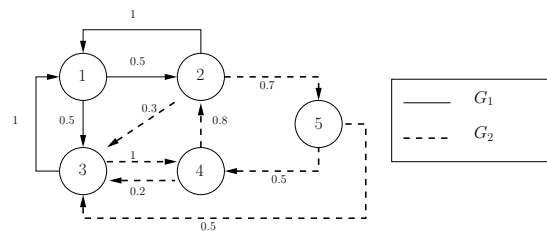


FIGURE 1 – 2 class network.

Le package contient 3 modules : `model`, `state` et `diagram`. Il utilise des packages de la distribution Anaconda (disponible ici : <http://continuum.io/downloads>).

1. Découverte du module `model`

- a) Importer le module `numpy` en le nommant `np`. Importer les classes `Model` et `ClassRouting` du module `model`.

```
>>> import numpy as np
>>> from model import Model, ClassRouting
```

- b) Définir `P` la matrice associée au réseau G_1 dans un objet de type `np.array`. Créer `R` un objet de type `ClassRouting` à partir de `P`.

```
>>> P=np.array([[0,0.5,0.5],[1,0,0],[0,1,0]])
>>> R=ClassRouting(P)
```

- c) Soit `Mu=[1,1,1]` la matrices des taux de services dans chaque file. On considère que le réseau G_1 contient 5 clients. Quelles sont les methodes de la classe `Model`? Utiliser la fonction `help` (taper `q` pour en sortir). Construire `Mod1` un objet de type `Model` et l'afficher.

```
>>> help(Model)
>>> Mu=[1,1,1]
>>> Mod1=Model(5,Mu,[R])
>>> print Mod1
```

- d) Produire un couple aléatoire (i, j) à l'aide de `Mod1`.

```
>>> (i,j)=Mod1.rand_iJ()
>>> print 'i=',i
>>> print 'j=',j
```

2. Découverte du module `state`

- a) Importer la classe `SetStates` à partir du module `state`. Quelles sont ses méthodes ?
- b) Construire `S1` l'ensemble de tous les états associés au modèle `Mod1`. Afficher `S1`.
- c) Effectuer la transition $t_{1,2}$ sur `S1`.
- d) Combien d'états contient maintenant `S1` ?
- e) Remettre tous les états dans `S1`.

3. Découverte du module `diagram`

- a) Importer la classe `Diagram` à partir du module `diagram`. Quelles sont ses méthodes ?
- b) Construire `D1` le diagramme complet associé au modèle `Mod1` et le dessiner.
- c) Combien d'arcs possède `D1` ?
- d) Combien d'états peut-on lire dans `D1` ?
- e) Effectuer la transition $T_{1,2}$ sur `S1`.
- f) Combien d'arcs contient maintenant `D1` ?
- g) Combien d'états peut-on lire dans `D1` ? Les afficher.
- h) Transformer `D1` en un diagramme complet.

4. Utilisation de `PyClones` avec 2 classes de clients

- a) Construire `Mod2` le modèle associé à la Figure 1. Considérer $M=(2 \ 3)$ et $\mu=[1 \ 1 \ 1 \ 1 \ 1]$.
- b) Construire `S2` l'ensemble de tous les états associés au modèle `Mod2`. Effectuer la transition $t_{2,[1,5]}$ sur `S2`. Considérer une politique de type `PRIORITAIRE`. Combien `S2` contient-il d'états maintenant ?
- c) Construire `D2` le diagramme complet associé au modèle `Mod2`. Effectuer la transition $T_{2,[1,5]}$ sur `S2`. Considérer une politique de type `PRIORITAIRE`. Combien d'états représente `D2` maintenant ? Dessiner `D2`.

La fonction `PSstate` utilise la classe `SetStates` pour produire un état aléatoire en utilisant l'algorithme de simulation parfaite.

```

def PS_state(Mod):

    """ Produces one random state with Perfect Sampling algorithm on states """

    S=SetStates(Mod)
    n=1
    U=[Mod.rand_iJ()]
    (i,J)=U[0]
    S.t(i,J)

    while len(S)>1:
        S.reset()
        UU=[Mod.rand_iJ() for nn in xrange(0,n)]
        U=UU+U
        n=n*2

        itU=(r for r in U)
        for (i,J) in itU:
            S.t(i,J)

    return S.get_one_state()

```

5. Simulation parfaite

- a) Ecrire PSdiagram l'analogue de la fonction PSstate en utilisant la classe Diagram.
- b) La tester sur Mod1 et Mod2.