Graphab 2.6

Manuel de référence
Ligne de commande

Céline Clauzel, Jean-Christophe Foltête, Xavier Girardet, Gilles Vuidel

26/01/2021

Contents

Chapter 1

Prérequis

Le programme Graphab est un outil de modélisation des réseaux écologiques fondé sur les graphes paysagers.

Il peut être utilisé en ligne de commande (CLI : Command Line Interface) à partir de la version 1.2. Ce mode est utile pour exécuter Graphab sur un ordinateur distant sans interface graphique, ou bien lancer automatiquement plusieurs traitements à la suite.

Attention, les projets ne sont pas compatibles entre les version 1.x et 2.x de Graphab !

1.1 A propos de Graphab

1.1.1 Auteurs

Le programme Graphab a été développé par Gilles Vuidel et Jean-Christophe Foltête au laboratoire ThéMA (Université de Franche-Comté – CNRS). Le développement du logiciel a été financé par le ministère de l'écologie, du développement durable, des transports et du logement dans le cadre du programme ITTECOP. Le logo de Graphab a été conçu par Gachwell.

1.1.2 Conditions d'utilisation

Le programme Graphab est disponible librement sous licence GPL. Les utilisateurs de Graphab sont invités à citer la référence [Foltête et al.(2012a)] dans leurs travaux :

Foltête J.C., Clauzel C., Vuidel G., 2012. A software tool dedicated to the modelling of landscape networks. Environmental Modelling & Software, 38: 316-327.

1.2 Configuration requise

Graphab fonctionne sur n'importe quel ordinateur supportant Java 8 ou supérieur (PC sous Linux, Windows, Mac...). Toutefois, lorsqu'il s'agit de données très volumineuses, la quantité de mémoire vive (RAM) de l'ordinateur peut limiter le nombre maximum de nœuds et de liens qui peuvent être traités en une seule fois avec Graphab. De plus, pour certaines métriques complexes, la puissance de votre processeur va déterminer la vitesse de leur calcul. Vous trouverez plus de détails à ce sujet dans la section 4 et dans [Foltête et al.(2012a)].

1.3 Installation et lancement du programme

Graphab est téléchargeable à cette adresse : `https://sourcesup.renater.fr/graphab`.

- Télécharger et installer Java 8 ou + (java.com ou adoptopenjdk.net). Installer de préférence la version 64 bits de Java.
- Télécharger graphab-2.6.jar
- Lancer graphab-2.6.jar

1.4 Lancer Graphab en ligne de commande

Il faut tout d'abord ouvrir une fenêtre de terminal, puis aller dans le répertoire contenant Graphab avec la commande *cd*. Enfin, vous pouvez saisir la commande suivante pour afficher l'écran d'aide de Graphab :

```
java -jar graphab-2.6.jar --help
```

Résultat

```
Usage :
java -jar graphab-2.6.jar --project prjfile.xml
...
...
```

Vous êtes prêt à utiliser Graphab en ligne de commande.

1.5 Syntaxe

1.5.1 Définition

Les commandes commencent par un double tiret : `--project, --linkset, ...`
Les options globales commencent par un tiret simple : `-proc, -nosave, ...`
Les paramètres n'ont pas de tiret : `name, complete, maxcost, ...`

1.5.2 Séparateur

Les espaces sont utilisés pour séparer les différents éléments d'une ligne de commande, par conséquent, vous ne pouvez pas avoir un nom qui contient des espaces.
Evitez de mettre des espaces dans les noms des éléments d'un projet.

1.5.3 Paramètre optionnel

Les paramètres entourés de crochets sont optionnels, et donc les paramètres qui ne sont pas entourés par des crochets sont obligatoires.

1.5.4 Intervalle et liste de valeurs

Définir un intervalle pour un paramètre (au lieu d'une valeur simple) exécutera la commande plusieurs fois pour chaque valeur définie dans l'intervalle. Un intervalle est défini par un minimum, un incrément et un maximum :

```
min:inc:max
```

Un intervalle de 0 à 10 avec un incrément de 2 créera 6 valeurs : 0, 2, 4, 6, 8, 10 :

```
0:2:10
```

La valeur minimale est toujours inclue, mais la valeur maximale est inclue seulement si la valeur incrémentée tombe exactement sur le maximum, sinon la dernière valeur sera la valeur incrémentée maximale inférieure au maximum. Un intervalle de 0 à 9 incrémenté de 2 donnera 5 valeurs : 0, 2, 4, 6, 8 :

```
0:2:9
```

Des valeurs décimales peuvent être utilisées, avec le point comme séparateur décimal :

```
1.5:0.5:4
```

Pour utiliser un ensemble de valeurs qui ne suit pas une progression arithmétique, vous pouvez utiliser une liste de valeurs séparées par des virgules :

```
0,1,4,8,10
```

La liste de valeurs ne doit pas contenir d'espace.

Une liste de valeurs ou une valeur simple peuvent toujours être utilisées à la place d'un intervalle. Si une commande contient plusieurs intervalles pour différents paramètres, la commande sera exécutée pour toutes les combinaisons possibles.

1.5.5 Séquençage des commandes

Graphab peut être lancé avec plusieurs commandes sur la même ligne, sauf pour les commandes *–help* et *–metrics*. Les commandes *–create* et *–project* peuvent être utilisées une seule fois et doivent être la première commande. Après une de ces deux commandes, toutes les autres commandes seront exécutées séquentiellement dans le même ordre que sur la ligne.

```
java -jar graphab-2.6.jar --project prj.xml --graph --gmetric NC
```

Cette ligne de commande charge le projet *prj.xml*, puis exécute la commande *–graph*, puis la commande *–gmetric*.

Chapter 2

Références des commandes

2.1 Commandes générales

2.1.1 –help : affichage de l'aide

Commande :

```
java -jar graphab-2.6.jar --help
```

Résultat :

```
Usage :
java -jar graphab.jar --metrics
java -jar graphab.jar [-proc n] --create prjname landrasterfile habitat=code1,...,coden ...
java -jar graphab.jar [-mpi | -proc n] [-nosave] --project prjfile.xml command1 [command2 ...]
Commands list :
--show
--dem rasterfile
--linkset distance=euclid|cost [name=linkname] [complete] [maxcost=valcost] [slope=coef] [remcrosspa
--uselinkset linkset1,...,linksetn
--corridor maxcost=[{]valcost[}] [format=raster|vector] [beta=exp|var=name]
--graph [name=graphname] [nointra] [threshold=[{]min:inc:max[}]]
--usegraph graph1,...,graphn
--cluster d=val p=val [beta=val] [nb=val]
--pointset pointset.shp id=fieldname [name=pointname] [random_absence=value [inpatch|outpatch[=dist]]
--usepointset pointset1,...,pointsetn
--pointdistance type=space|graph distance=leastcost|circuit|flow|circuitflow [dist=val proba=val]
--capa [area [exp=value] [code1,..,coden=weight ...]] | [maxcost=[{]valcost[}] codes=code1,code2,...
--gmetric global_metric_name [resfile=file.txt] [maxcost=valcost] [param1=[{]min:inc:max[}] [param2=
--cmetric comp_metric_name [maxcost=valcost] [param1=[{]min:inc:max[}] [param2=[{]min:inc:max[}] ...]
--lmetric local_metric_name [maxcost=valcost] [param1=[{]min:inc:max[}] [param2=[{]min:inc:max[}] ..
--interp name resolution var=patch_var_name d=val p=val [multi=dist_max [sum]]
--model variable distW=min:inc:max [vars=var1,...,varn] [raster=r1,...,rn]
--delta global_metric_name [maxcost=valcost] [param1=[{]val[}] ...] obj=patch|link [sel=id1,id2, ...
--addpatch npatch global_metric_name [param1=val ...] [gridres=min:inc:max [capa=capa_file] ...
--remelem nstep global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link [sel=id1,id2, .
--gtest nstep global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link [sel=id1,id2, ...
--gremove global_metric_name [maxcost=valcost] [param1=val ...] [patch=id1,id2,...,idn|fpatch= ...
--metapatch [mincapa=value]
--landmod zone=filezones.shp id=fieldname code=fieldname [sel=id1,id2,...,idn ] [novoronoi]

min:inc:max -> val1,val2,val3...
```

2.1.2 –metrics : affichage des métriques

Cette commande affiche l'ensemble des métriques disponibles avec leur abréviation, leur description et leurs paramètres.

Commande :

```
java -jar graphab-2.6.jar --metrics
```

Résultat :

```
===== Global metrics =====
S#F - Sum Flux (F)
        params : [d, p, beta]
EC - Connectivité équivalente  (EC)
        params : [d, p]
PC - Probabilité de connectivité (PC)
        params : [d, p]
IIC - Indice intégral de connectivité (IIC)
CCP - Probabilité de coïncidence (CCP)
MSC - Taille moyenne des composantes (MSC)
SLC - Taille de la plus grande composante (SLC)
ECS - Taille de cluster attendue (ECS)
GD - Diamètre (GD)
H - Harary indice (H)
NC - Nombre de composantes (NC)
dPC - Delta PC décomposé (dPC)
        params : [d, p]
W - Indice de Wilks (W)
        params : [attrs, npatch, warea]

===== Local metrics =====
F : Flux (F)
        params : [d, p, beta]
BC : Centralité intermédiaire (BC)
        params : [d, p, beta]
IF : Flux d'interaction (IF)
        params : [d, p, beta]
Dg : Degré du noeud (Dg)
CC : Coefficient d'agrégation (CC)
CCe : Centralité de proximité (CCe)
CCor : Corrélation de Connectivité (CCor)
Ec : Excentricité (Ec)
CF : Flux circuit (CF)
        params : [beta]
```

2.2 Gestion du projet

2.2.1 –create : création d'un projet

`java -jar graphab-2.6.jar --create` *`prjname landrasterfile`* `habitat=`*`code1,...,coden`* `[nodata=`*`val`*`] [minarea=`*`val`*`] [maxsize=`*`val`*`] [con8] [simp] [dir=`*`path`*`]`

Paramètres obligatoires

- `prjname` : nom du nouveau projet

- `landrasterfile` : nom du fichier image contenant l'occupation du sol au format tiff, ascii grid ou rst
- `habitat=code1,...,coden` : le ou les codes d'occupation du sol représentant les taches d'habitat.

Paramètres optionnels

- `nodata=val` : valeur représentant l'absence de données dans l'image raster
- `minarea=val` : taille minimale d'une tache d'habitat en hectare
- `maxsize=val` : découpe les taches dont la longueur ou largeur dépasse `maxsize` sur une grille de taille `maxsize`
- `con8` : connexité aux 8 pixels voisins pour la définition des taches, par défaut la connexité est seulement de 4
- `simp` : simplifie la géométrie des taches pour accélérer le calcul de la topologie planaire (voronoï)
- `dir=path` : chemin où enregistrer le projet, par défaut le projet est enregistré dans le répertoire courant.

Description

Cette commande permet de créer un nouveau projet et de le charger.

La commande `--create` ne peut être utilisée qu'une seule fois et doit être la première commande de la ligne. Les commandes suivantes seront exécutées sur le projet nouvellement créé.

2.2.2 –project : chargement d'un projet

Cette commande définit le chemin vers le fichier xml du projet à charger.

```
java -jar graphab-2.6.jar --project path2myproject/myproject.xml
```

Charge le projet *myproject* contenu dans le répertoire *path2myproject*.

La commande `--project` ne peut être utilisée qu'une seule fois et doit être la première commande de la ligne. Toutes les commandes qui suivent dans ce manuel ont besoin d'un projet chargé.

2.2.3 –show : affichage des éléments d'un projet

Affiche l'ensemble des jeux de liens, graphes et jeux de points contenu dans le projet chargé. Cette commande est utile pour récupérer les noms exacts des éléments pour les utiliser par la suite dans une ligne de commande.

Commande :

```
java -jar graphab-2.6.jar --project path2myproject/myproject.xml --show
```

Résultat :

```
===== Link sets =====
Complete_Euclid
Complete_cost
Planar_Euclid
Planar_cost

===== Graphs =====
2000m-3500cost
2000m-3500cost_comp
```

```
2000m_euclid
2000m_euclid_comp

===== Point sets =====
Presence_absence
```

Attention aux noms des éléments : la ligne de commande est sensible à la casse (différentiation minuscule/majuscule) et ne gère pas les espaces dans les noms.

2.2.4 –capa : capacité des taches

```
--capa [area [exp=value] [code1,..,coden=weight ...]]
       | [file=capacity.csv id=fieldname capa=fieldname]
       | [maxcost=[{]valcost[}] codes=code1,code2,...,coden [weight]]
```

Paramètres optionnels

- `area`: défini la capacité comme la superficie de la taches (défaut)
- `exp=value`: modifie la surface des taches par un exposant.
- `code1,..,coden=weight ...`: pondération de la superficie de la tache par les codes d'occupation du sol la composant. Ce paramètre est utile seulement quand les taches sont définies par plusieurs codes d'occupation du sol.
- `maxcost=[{]valcost[}]` : coût cumulé maximum pour le voisinage
- `codes=code1,code2,...,coden` : codes d'occupation du sol à prendre en compte dans la surface de voisinage
- `weight` : introduit une pondération en fonction de l'éloignement à la tache, par le biais d'une fonction exponentielle négative. De cette façon, les surfaces sélectionnées compteront plus si elles sont proches de la tache et inversement.

Description

La commande `--capa` calcule la capacité des taches et enregistre le projet sauf si l'option `-nosave` a été spécifiée. Sans paramètre ou avec le paramètre `area`, la capacité est définie comme la surface des taches en m^2. Avec le paramètre `maxcost`, la capacité est calculée comme une surface d'éléments autour de la tache, jusqu'à la distance *valcost*. Les distances sont calculées à partir des coûts définis dans le jeu de lien sélectionné. Si c'est un jeu de lien en euclidien, les coûts sont mis à 1 pour l'ensemble des classes d'occupation du sol.

Cette commande supporte la sélection d'un seul jeu de lien (cf. –uselinkset : sélection des jeux de liens).

Exemples

```
--capa
```

```
--capa area
```

Les 2 exemples précédents calculent la capacité comme la surface de la tache (en m^2).

```
--capa area 1=0.5 5=2
```

Défini la capacité de la tache comme la surface de la tache (en m^2) pondérée par 0.5 pour le code d'occupation du sol 1 et pondérée par 2 pour le code d'occupation du sol 5.

```
--capa maxcost=100 codes=1,3
```

Définit la capacité de chaque tache comme la surface (en m^2) d'éléments de code d'occupation du sol 1 et 3 présent autour de la tache jusqu'à une distance de 100 (en coût).

2.2.5 –metapatch : création d'un projet méta-tache

`--metapatch [mincapa=`*`value`*`]`

Paramètre optionnel

- `mincapa=value` : capacité minimale d'une méta-tache pour être conservée dans le nouveau projet

Description

La commande `--metapatch` crée un nouveau projet de méta taches basé sur le graphe sélectionné. Chaque composante du graphe sélectionné correspondra à une méta-tache. Celle-ci est donc constituée de l'ensemble des taches d'habitat connectées entre elles. La capacité des méta-taches est définie comme la somme de la capacité des taches composant la méta-tache. Le nouveau projet est enregistré dans un sous répertoire du projet actuel.

Ce nouveau projet devient le projet courant pour les commandes suivantes.

Cette commande supporte la sélection d'un seul graphe (cf. –usegraph : sélection des graphes).

Exemple

```
--usegraph 2000m_euclid --metapatch mincapa=1000
```

Cette commande créera un projet méta-tache basé sur le graphe *2000m_euclid* et supprimera toutes les méta-taches dont la capacité est inférieure à 1000.

Références

[Clauzel et al.(2015b)], [Foltête et al.(2016)]

2.2.6 –dem : chargement d'un MNT

Cette commande définit le chemin vers un MNT qui pourra être utilisé dans la création des liens pour tenir compte de la pente.

`--dem `*`rasterfile`*

Paramètre obligatoire

- `rasterfile` : fichier raster au format Tiff ou AsciiGrid représentant les altitudes. La géométrie du raster doit correspondre exactement à la carte du paysage.

2.3 Jeu de liens et graphes

2.3.1 –linkset : création d'un jeu de liens

`--linkset distance=euclid|cost [name=`*`linkname`*`] [complete] [maxcost=`*`valcost`*`] [slope=`*`coef`*`]`
`[remcrosspath|nopathsaved] [[`*`code1,..,coden=cost1 ...`*`] `*`codei,..,codej=min:inc:max`*
`| extcost=`*`rasterfile`*`]`

Paramètre obligatoire

- `distance=euclid|cost` : type distance pour les liens, euclidien (`distance=euclid`) ou coût cumulé (`distance=cost`). Si `distance=cost`, il faut définir les coûts par classe d'occupation du sol ou un raster contenant les coûts.

Paramètres optionnels généraux

- `name=linkname` : nom du jeu de lien qui sera créé
- `complete` : passe en topologie complète au lieu de planaire
- `maxcost=valcost` : le paramètre *maxcost* permet de limiter le calcul des liens à une distance maximale.

Paramètres spécifiques au cas distance=cost

- `slope=coef` : pondère les couts par la pente. Pour utiliser cette option, il faut que le projet contienne un MNT.
- `remcrosspath` : supprime les liens traversant une tache
- `nopathsaved` : n'enregistre pas la géométrie des chemins. Utile pour réduire l'utilisation mémoire en topologie complète
- `[code1,..,coden=cost1 ...] codei,..,codej=min:inc:max` : définition des coûts à partir des codes de l'occupation du sol
- `extcost=rasterfile` : définition des coûts à partir d'un raster externe au format tiff, ascii grid ou rst.

Description

Créé un nouveau jeu de liens dans le projet chargé et enregistre le projet sauf si l'option globale `-nosave` est utilisée.

Si le paramètre `name` n'est pas défini, le nom du jeu de lien sera déterminé par la définition des coûts.

La commande `--linkset` ne supporte pas plusieurs intervalles ou liste de valeurs.

Après la commande `--linkset`, les jeux de liens sélectionnés correspondent aux jeux de liens créés.

Exemples

Cette commande créera un jeu de lien en topologie planaire nommé *cost_1_2_3_4_5_6_7-1.0* avec tous les coûts à 1 :

```
--linkset distance=cost 1,2,3,4,5,6,7=1
```

Cette commande créera un jeu de lien *cost_1_2_3-1.0* avec un coût égal à 1 pour les valeurs d'occupation du sol 1, 2 et 3 et un coût égal à 2 pour les valeurs d'occupation du sol 4, 5, 6 et 7 :

```
--linkset distance=cost 1,2,3=1 4,5,6,7=2
```

Par défaut la topologie est planaire ; pour créer une topologie complète, utilisez l'option `complete` :

```
--linkset distance=cost complete 1,2,3,4,5,6,7=1
```

Avec une topologie complète, on peut préciser un seuil de distance pour éviter de créer trop de liens. Exemple avec un seuil max à 100 :

```
--linkset distance=cost complete=100 1,2,3,4,5,6,7=1
```

On peut créer plusieurs jeux de liens avec des coûts différents en utilisant un intervalle ou une liste de valeurs :

```
        --linkset distance=cost 4,5,6,7=10 1,2,3=100:50:200
```

ou

```
        --linkset distance=cost 4,5,6,7=10 1,2,3=100,150,200
```

Résultat : la commande ci-dessus créera 3 jeux de liens *cost_1_2_3-100.0 cost_1_2_3-150.0 cost_1_2_3-200.0* avec les valeurs d'occupation du sol 1, 2 et 3 égales à 100, 150 ou 200.

2.3.2 –uselinkset : sélection des jeux de liens

`--uselinkset` *`linkset1,..,linksetn`*

Sélectionne les jeux de liens qui seront utilisés dans les commandes suivantes. Les noms de jeu de liens sont sensibles à la casse et ne peuvent pas contenir d'espace.

Par défaut, tous les jeux de liens sont sélectionnés.

2.3.3 –removelinkset : suppression de jeux de liens

`--removelinkset [`*`linkset1,..,linksetn`*`]`

Supprime les jeux de liens sélectionnés. Tous les graphes dépendant de ces jeux de liens sont aussi supprimés.
L'option `-nosave` est sans effet sur cette commande.

2.3.4 –graph : création d'un graphe

`--graph [name=graphname] [nointra] [threshold=[{]min:inc:max[}]]`

Paramètres optionnels

- `name=graphname` : nom du graphe qui sera créé. Ce paramètre peut être utilisé uniquement si la commande ne crée qu'un seul graphe.
- `nointra` : désactive les distances intra-tache pour le calcul des métriques
- `threshold=[{]min:inc:max[}]` : défini la distance maximale des liens inclus dans chaque graphe. Sans ce paramètre, le graphe contient tous les liens du jeu de lien. Entouré d'accolades, les valeurs de distance sont converties en valeur de coût automatiquement.

Description

Crée un graphe pour chaque jeu de lien sélectionné et enregistre le projet sauf si l'option globale `-nosave` est utilisée. Pour l'instant, la commande ne supporte pas la création de graphe MST (Minimum Spanning Tree).

Après la commande `--graph`, les graphes sélectionnés correspondent aux graphes venant d'être créés.

Exemples

Sans paramètre, la commande créera un graphe sans seuil pour chaque jeu de lien sélectionné :

```
        --graph
```

Le nom du graphe sera la concaténation de *comp_* et du nom du jeu de lien.

Le paramètre *threshold* permet de spécifier un seuil (100 dans l'exemple) pour le graphe à créer :

```
--graph threshold=100
```

Le nom du graphe sera la concaténation de *thresh_100.0_* et du nom du jeu de lien.

Si le paramètre `threshold` est défini avec un intervalle ou une liste de valeurs, un graphe élagué pour chaque valeur de seuil et chaque jeu de liens sera créé :

```
--graph threshold=1000:100:1500
```

ou

```
--graph threshold=1000,1100,1200,1300,1400,1500
```

Résultat : cette commande créera 6 graphes élagués (1000, 1100, 1200, 1300, 1400, 1500) pour chaque jeu de liens sélectionné.

2.3.5 –usegraph : sélection des graphes

`--usegraph` *graph1,..,graphn*

Sélectionne les graphes qui seront utilisés dans les commandes suivantes. Les noms de graphe sont sensibles à la casse et ne peuvent pas contenir d'espace.
Par défaut, tous les graphes sont sélectionnés.

2.3.6 –removegraph : suppression de graphes

`--removegraph [`*graph1,..,graphn*`]`

Supprime les graphes sélectionnés.
L'option `-nosave` est sans effet sur cette commande.

2.3.7 –corridor : calcul de corridors

`--corridor maxcost=[{]`*min:inc:max*`[}] [format=raster|vector] [beta=`*exp*`|var=`*name*`]`

Paramètre obligatoire

- `maxcost=[{]min:inc:max[}]` : coût cumulé maximum des chemins formant un corridor. Entouré d'accolades, la distance donnée en mètre est convertie en unité de coût automatiquement.

Paramètre optionnel

- `format=raster|vector` : format en sortie raster (.tif) ou vectoriel (.shp). Le format par défaut est vectoriel
- `beta=exp` :
- `var=name` :

Description

La commande `--corridor` calcule le corridor associé à chaque lien d'un jeu de lien. Le résultat est stocké dans un shapefile (ou un raster au format tif) différent pour chaque jeu de lien sélectionné. Chaque shapefile contient, pour chaque lien du jeu de lien, un polygone représentant l'ensemble des chemins possibles reliant les deux taches et ayant une distance en coût cumulé inférieure ou égale à *valcost*. Au format raster, chaque pixel dénombre le nombre de corridors passant par ce pixel.

Cette commande ne fonctionne pas avec les jeux de lien euclidiens, ils sont ignorés.

Exemple

```
--corridor maxcost=500
```

2.3.8 –cluster : partitionnement de graphe

`--cluster d=`*`val`* `p=`*`val`* `[beta=`*`val`*`] [nb=`*`val`*`]`

Paramètres obligatoires

- `d=val` : distance permettant de définir l'exposant α.
- `p=val` : probabilité permettant de définir l'exposant α. Pour définir α à 0 et ne pas tenir compte de l'impédance des liens, mettre `p` à 1.

Paramètres optionnels

- `beta=val` : exposant des capacités, par défaut à 1. Pour ne pas tenir compte de la capacité des taches, mettre `beta` à 0.
- `nb=val` : nombre de compartiments, par défaut le nombre de compartiments sélectionné correspond à la modularité maximale.

Description

Crée un graphe basé sur son partitionnement qui maximise la modularité [Newman(2006)] pour chaque graphe sélectionné et enregistre le projet sauf si l'option globale `-nosave` est utilisée. Les nouveaux graphes conservent uniquement les liens intra-groupes.

La modularité est calculée à partir d'un poids (w_{ij}) défini pour chaque lien du graphe :

$$w_{ij} = (a_i a_j)^{\beta} e^{-\alpha d_{ij}}$$

Les 2 paramètres β et α permettent de définir respectivement l'importance de la capacité des taches $(a_i a_j)$ et l'importance de la distance (d_{ij}) pour le poids du lien w_{ij}. Si $\alpha = \beta = 0$ alors les poids sont tous identiques : $w_{ij} = 1$.

Après la commande `--cluster`, les graphes sélectionnés correspondent aux graphes venant d'être créés.

Référence

[Foltête and Vuidel(2017)]

2.4 Métriques

2.4.1 –gmetric : calcul d'une métrique globale

`--gmetric` *`global_metric_name`* `[resfile=`*`file.txt`*`] [maxcost=`*`valcost`*`] [`*`param1=min:inc:max`*
`[`*`param2=min:inc:max ...`*`]]`

Paramètre obligatoire

- `global_metric_name` : acronyme de la métrique globale à calculer (PC, IIC, ...)

Paramètres optionnels

- `resfile=file.txt` : fichier contenant le résultat du calcul de la métrique
- `maxcost=valcost` : limite le calcul de chemin à *valcost* (ie. les chemins supérieurs à *valcost* ne seront pas calculés). Ce paramètre peut réduire significativement le temps de calcul d'une métrique basée sur le calcul de chemins, mais le résultat sera moins précis.
- `param1=[{]min:inc:max[}]` : paramètre(s) de la métrique, si elle en a. Entouré d'accolades, les valeurs de distance sont converties en valeur de coût automatiquement.
- ...

Description

Calcule la métrique globale données sur chaque graphe sélectionné. Le nom de la métrique est son nom court comme affiché par la commande `--metrics`. Si la métrique a besoin de paramètres, ils peuvent être spécifiés dans n'importe quel ordre. Les résultats sont enregistrés dans un fichier texte dans le répertoire du projet. Par défaut, le nom du fichier correspond au nom court de la métrique. Un autre nom peut être donné par le paramètre `resfile`.

Exemples

Calcule la métrique NC sur tous les graphes sélectionnés :

```
--gmetric NC
```

Le résultat est enregistré dans le fichier NC.txt dans le répertoire du projet :

```
Graph                NC
2000m-3500cost       25.0
2000m-3500cost_comp  24.0
2000m_euclid         9.0
2000m_euclid_comp    9.0
```

Pour les métriques ayant des paramètres, on peut tester plusieurs valeurs de paramètre en une seule commande. La commande suivante calcule la métrique PC avec 6 jeux de paramètres différents pour chaque graphe sélectionné :

```
--gmetric PC d=1000:500:2000 p=0.05 beta=0,1
```

Le résultat est stocké dans le fichier PC.txt :

```
Graph               d            p           beta         PC
2000m-3500cost      1000.0       0.05        0.0         2.108166945899072E-15
2000m-3500cost      1500.0       0.05        0.0         2.4839095790785042E-15
2000m-3500cost      2000.0       0.05        0.0         2.866220685546806E-15
2000m-3500cost      1000.0       0.05        1.0         1.317091007462398E-6
2000m-3500cost      1500.0       0.05        1.0         1.4758311225154786E-6
2000m-3500cost      2000.0       0.05        1.0         1.5884005111333579E-6
2000m-3500cost_comp 1000.0       0.05        0.0         2.1106833149811817E-15
2000m-3500cost_comp 1500.0       0.05        0.0         2.493027588606987E-15
2000m-3500cost_comp 2000.0       0.05        0.0         2.887027144684246E-15
2000m-3500cost_comp 1000.0       0.05        1.0         1.3171878007185563E-6
2000m-3500cost_comp 1500.0       0.05        1.0         1.476224502635306E-6
2000m-3500cost_comp 2000.0       0.05        1.0         1.5892024206564504E-6
2000m_euclid        1000.0       0.05        0.0         2.8238137481213476E-15
2000m_euclid        1500.0       0.05        0.0         3.516516320195261E-15
2000m_euclid        2000.0       0.05        0.0         4.285009196943927E-15
2000m_euclid        1000.0       0.05        1.0         1.7079340030649911E-6
2000m_euclid        1500.0       0.05        1.0         1.8176869551880345E-6
```

```
2000m_euclid        2000.0        0.05        1.0        1.8976284261240914E-6
2000m_euclid_comp   1000.0        0.05        0.0        2.867215798466552E-15
2000m_euclid_comp   1500.0        0.05        0.0        3.581685718161269E-15
2000m_euclid_comp   2000.0        0.05        0.0        4.374805768414666E-15
2000m_euclid_comp   1000.0        0.05        1.0        1.7172345329380555E-6
2000m_euclid_comp   1500.0        0.05        1.0        1.8277346595840518E-6
2000m_euclid_comp   2000.0        0.05        1.0        1.9080569002930505E-6
```

2.4.2 –cmetric : calcul d'une métrique par composante

`--cmetric` *`global_metric_name`* [`maxcost=`*`valcost`*`]` [*`param1=min:inc:max`* [*`param2=min:inc:max`* `...`]]

Paramètre obligatoire

- `global_metric_name` : acronyme de la métrique globale à calculer (PC, IIC, ...)

Paramètres optionnels

- `maxcost=valcost` : limite le calcul de chemin à *valcost* (ie. les chemins supérieurs à *valcost* ne seront pas calculés). Ce paramètre peut réduire significativement le temps de calcul d'une métrique basée sur le calcul de chemins, mais le résultat sera moins précis.
- `param1=[{]min:inc:max[}]` : paramètre(s) de la métrique, si elle en a. Entouré d'accolades, les valeurs de distance sont converties en valeur de coût automatiquement.
- ...

Description

Calcule la métrique globale donnée sur chaque composante de chaque graphe sélectionné. Le nom de la métrique est son nom court comme affiché par la commande `--metrics`. Le résultat est enregistré dans un attribut des composantes de chaque graphe. Le projet est enregistré sauf si l'option globale `-nosave` est utilisée.

Exemple

Avec une métrique ayant des paramètres, on peut tester plusieurs jeux de paramètres en une commande :

```
--cmetric PC d=1000:500:2000 p=0.05 beta=0,1
```

6 PC seront calculés pour chaque composante de chaque graphe sélectionné :

- PC_d1000_p0.05_beta0
- PC_d1500_p0.05_beta0
- PC_d2000_p0.05_beta0
- PC_d1000_p0.05_beta1
- PC_d1500_p0.05_beta1
- PC_d2000_p0.05_beta1

2.4.3 –lmetric : calcul d'une métrique locale

`--lmetric` *`local_metric_name`* [`maxcost=`*`valcost`*`]` [*`param1=min:inc:max`* [*`param2=min:inc:max`* `...`]]

Paramètre obligatoire

- `local_metric_name` : acronyme de la métrique locale à calculer (F, CF, ...)

Paramètres optionnels

- `maxcost=valcost` : limite le calcul de chemin à *valcost* (ie. les chemins supérieurs à *valcost* ne seront pas calculés). Ce paramètre peut réduire significativement le temps de calcul d'une métrique basée sur le calcul de chemins, mais le résultat sera moins précis.
- `param1=[{]min:inc:max[}]` : paramètre(s) de la métrique, si elle en a. Entouré d'accolades, les valeurs de distance sont converties en valeur de coût automatiquement.
- ...

Description

Calcule la métrique locale donnée sur chaque élément (noeud et/ou lien) de chaque graphe sélectionné. Le nom de la métrique est son nom court comme affiché par la commande `--metrics`. Le résultat est enregistré dans un attribut des taches et/ou des liens. Le projet est enregistré sauf si l'option globale `-nosave` est utilisée. L'option `-nosave` est utile, utilisée conjointement avec la commande `--model` ou `--interp`.

Exemple

Avec une métrique ayant des paramètres, on peut tester plusieurs jeux de paramètres en une commande :

```
--lmetric F d=1000:500:2000 p=0.05 beta=0,1
```

La métrique F sera calculée avec 6 jeux de paramètres différents pour chaque élément de chaque graphe sélectionné :

- F_d1000_p0.05_beta0
- F_d1500_p0.05_beta0
- F_d2000_p0.05_beta0
- F_d1000_p0.05_beta1
- F_d1500_p0.05_beta1
- F_d2000_p0.05_beta1

2.4.4 –interp : interpolation de métriques

`--interp` *name resolution* `var=`*patch_var_name* `d=`*val* `p=`*val* `[multi=`*dist_max* `[sum]]`

Paramètres obligatoires

- `name` : nom du fichier raster stockant le résultat de l'interpolation
- `resolution` : résolution de l'interpolation. Dans certains cas, ce paramètre est ignoré et la résolution de la carte d'occupation du sol est utilisée à la place.
- `var=patch_var_name` : nom de la variable à interpoler. Ce nom doit correspondre à un attribut des taches ; habituellement le résultat du calcul d'une métrique locale.
- `d=val` : distance permettant, avec le paramètre `p`, de définir le coefficient α

- `p=val` : probabilité permettant, avec le paramètre `d`, de définir le coefficient α

Paramètres optionnels

- `multi=dist_max` : interpolation à partir de toutes les taches à une distance inférieure ou égale à *dist_max*, au lieu de la tache la plus proche
- `sum` : agrégation par une somme au lieu d'une moyenne pondérée. Utilisé uniquement avec le paramètre `multi`.

Description

La commande `--interp` permet d'interpoler sur l'ensemble de la zone une donnée disponible au niveau des taches, habituellement une métrique locale. L'interpolation est basée sur une fonction décroissante de la distance. La valeur interpolée à un point p_j de l'espace sera :

$$p_j = var_i * e^{-\alpha d_{ij}}$$

var_i représente la valeur de la variable pour la tache la plus proche du point p_j. Le coefficient α est défini par les 2 paramètres `d` et `p`, tel que $p = e^{-\alpha d}$. Le calcul de la distance entre le point et la tache la plus proche d_{ij} dépend du jeu de liens en cours (euclidien ou coût cumulé).

Si le paramètre `multi` est activé, l'interpolation ne se calcule plus uniquement avec la tache la plus proche, mais aussi avec toutes les taches dont la distance est inférieure ou égale à *dist_max*. Ces différentes valeurs seront agrégées, par défaut, par une moyenne pondérée. Si le paramètre `sum` est ajouté, l'agrégation utilisée sera une somme. Dans ce cas la formule d'interpolation d'un point p_j devient :

$$p_j = \sum_i var_i * e^{-\alpha d_{ij}}$$

Si plusieurs jeux de liens sont sélectionnés, une interpolation différente sera calculée pour chaque jeu de liens.

Exemple

```
--interp test 10 var=F_d1000_p0.5_beta1_2000m_euclid d=1000 p=0.5 multi=1000 sum
```

2.5 Jeu de points et SDM

2.5.1 –pointset : import de jeu de points

`--pointset` *`pointset.shp`* `id=`*`fieldname`* `[name=`*`pointname`*`] [random_absence=`*`value`* `[inpatch|outpatch[=`*`dist`*

Importe un nouveau jeu de points à partir du shapefile donné pour chaque jeu de liens sélectionné et enregistre le projet sauf si l'option globale `-nosave` est utilisée.

Paramètres obligatoires

- `pointset.shp` : shapefile des points
- `id=fieldname` : nom du champ identifiant chaque point

Paramètres optionnels

- `name=pointname` : définit le nom du nouveau jeu de points
- `random_absence=value` : génère des points de pseudo absence. `value` entre 0 et 1 représente une estimation de la part de points à générer
- `inpatch` : génère les points de pseudo absence uniquement dans les taches
- `outpatch=dist` : génère les points de pseudo absence à au moins une distance `dist` des taches

Le nom par défaut du jeu de point est la concaténation du nom du fichier importé et du nom du jeu de lien. Après la commande `--pointset`, les jeux de points sélectionnés correspondent à ceux nouvellement créés.

2.5.2 –usepointset : sélection de jeux de points

`--usepointset` *`ps1,..,psn`*

Sélectionne les jeux de points à utiliser par la commande `--model`. Par défaut, tous les jeux de points sont sélectionnés.

2.5.3 –removepointset : suppression de jeux de points

`--removepointset [`*`pointset1,..,pointsetn`*`]`

Supprime les jeux de points sélectionnés.
L'option `-nosave` est sans effet sur cette commande.

2.5.4 –pointdistance : matrice de distances d'un jeu de point

`--pointdistance type=space|graph distance=leastcost|circuit|flow|circuitflow [dist=`*`val`* `proba=`*`val`*`]`

Paramètres obligatoires

- `type`: le calcul de distance est basé sur l'espace continu `space` (euclidien ou raster) ou sur le graphe `graph`
- `distance`: à partir d'un raster, la distance peut être moindre coût (`leastcost`) ou `circuit`, à partir d'un graphe, la distance peut être `leastcost`, `circuit`, `flow` or `circuitflow`

Paramètres optionnels

Pour les distances `flow` et `circuitflow` les paramètres suivants doivent être renseignés :

- `dist` : la distance entre 2 taches
- `proba` : la probabilité de mouvement entre 2 taches pour la distance `dist`

Description

Cette commande calcule la matrice de distances pour chaque jeu de points sélectionnés et enregistre chaque matrice dans un fichier texte commençant par `distance_`. Pour un calcul basé sur un raster de coût, une matrice de distance sera calculé pour chaque jeu de points et chaque jeu de liens sélectionnés, pour un calcul basé sur un graphe, une matrice de distance sera calculé pour chaque jeu de points et chaque graphe sélectionnés. Dans ce second cas, les graphes sélectionnés doivent être issus du même jeu de liens que les jeux de points sélectionnés.

Exemple

Dans l'exemple ci-dessous, les distances de moindre coût entre les points sera calculées à partir des raster de coûts de chaque jeu de liens sélectionné et ce, pour chaque jeu de points sélectionné.

```
--pointdistance type=space distance=leastcost
```

2.5.5 –model : calcul du SDM

`--model` *`variable`* `distW=`*`min:inc:maxcost`* `[vars=`*`var1,...,varn`*`] [raster=`*`r1,...,rn`*`]`

Paramètres obligatoires

- `variable` : nom de la variable binaire contenu dans le(s) jeux(x) de points
- `distW=min:inc:maxcost` : pondération de la distance entre la tache et le point avec une probabilité de 0.05

Paramètres optionnels

- `vars=var1,...,varn` : ajoute d'autres variables de la couche des taches non liées au graphe (Capacity par exemple)
- `raster=r1,...,rn` : ajoute des variables provenant de raster externe

Description

Calcule un SDM avec chaque métrique locale existant dans chaque graphe sélectionné pour la variable binaire donnée sur chaque jeu de points sélectionné. La *variable* doit exister dans chaque jeu de point sélectionné. Pour chaque graphe, la commande cherche un jeu de points ayant le même jeu de lien que le graphe. Si il n'en existe pas, la commande émet une erreur, si il en existe plusieurs, le choix du jeu de point est indéterminé. Le paramètre `distW` défini la pondération par la distance entre la tache et le point avec une probabilité de 0.05.

Exemple

```
--model PRESENCE distW=1000,2000
```

Le résultat est enregistré dans le fichier model-PRESENCE-dW1000,2000.txt dans le répertoire du projet :

Graph	Metric	DistWeight	R2	AIC	Coef
2000m-3500cost	BC_d3500_p0.05_beta1	1000.00	0.229898	56.0689	3.63230e-07
2000m-3500cost	BC_d3500_p0.05_beta1	2000.00	0.134672	62.7547	4.89398e-08
2000m-3500cost	F_d3500_p0.05_beta1	1000.00	0.522162	35.5490	0.00155784
2000m-3500cost	F_d3500_p0.05_beta1	2000.00	0.266793	53.4785	0.000145906
2000m-3500cost	d_PC	1000.00	0.296785	51.3727	471.384
2000m-3500cost	d_PC	2000.00	0.292047	51.7054	460.552

Références

[Foltête et al.(2012b), Foltête et al.(2012a), Girardet et al.(2013), Clauzel et al.(2013)]

2.6 Ajout/Suppression d'élements

2.6.1 –delta : suppression d'un élément

`--delta` *global_metric_name* [`maxcost=`*valcost*`]` [*param1*`=[{]`*val*`[}]` ...] `obj=patch|link`
[`sel=`*id1,id2,...,idn* | `fsel=`*file.txt*`]`

Paramètres obligatoires

- `global_metric_name` : acronyme de la métrique globale (PC, ...)
- `obj=patch|link` : teste la suppression des taches (`obj=patch`) ou des liens (`obj=link`)

Paramètres optionnels

- `maxcost=valcost` : limite le calcul de chemin à *valcost* (ie. les chemins supérieurs à *valcost* ne seront pas calculés). Ce paramètre peut réduire significativement le temps de calcul d'une métrique basée sur le calcul de chemins, mais le résultat sera moins précis.
- `param1=[{]val[}]` : paramètre(s) de la métrique, si elle en a.
- `sel=id1,id2,...,idn` : limite le calcul aux éléments (taches ou liens) listés par leur identifiant
- `fsel=file.txt` : limite le calcul aux éléments (taches ou liens) listés dans le fichier *file.txt*. Le fichier doit contenir un identifiant par ligne.

Description

Calcule une métrique globale en mode delta sur les noeuds ou les liens en fonction du paramètre `obj` pour chaque graphe sélectionné.

Si le paramètre `sel` ou `fsel` est défini, le calcul est effectué seulement sur les éléments listés. Le résultat est stocké dans un fichier texte distinct pour chaque graphe dans le répertoire du projet. Le nom du fichier est la concaténation de 'delta-' + nom court de la métrique + nom du graphe.

Exemples

Calcule la métrique NC en delta pour toutes les taches :

```
--delta NC obj=patch
```

Calcule la métrique PC en delta pour les taches ayant l'identifiant 2 et 3 :

```
--delta PC d=1000 p=0.05 beta=1 obj=patch sel=2,3
```

Le résultat est stocké dans un fichier pour chaque graphe. Exemple :

```
Id    d_PC
Init  1.3170910074623971E-5
2     2.68908916812349E-3
3     1.738640713802898E-4
```

Init correspond à la valeur initiale du PC sans enlevé d'élément. Les valeurs suivantes correspondent à la perte relative de la métrique après avoir enlevé l'élément correspondant du graphe. En enlevant la tache 2, le PC diminue de 0.269%

2.6.2 –gremove : suppression de plusieurs éléments

`--gremove` *`global_metric_name`* `[maxcost=`*`valcost`*`] [`*`param1=val ...`*`]`
`[patch=`*`id1,id2,...,idn`*`|fpatch=`*`file.txt`*`] [link=`*`id1,id2,...,idm`*`|flink=`*`file.txt`*`]`

Paramètre obligatoire

- `global_metric_name` : acronyme de la métrique globale à calculer (PC, ...)

Paramètres optionnels

- `maxcost=valcost` : limite le calcul de chemin à *valcost* (ie. les chemins supérieurs à *valcost* ne seront pas calculés). Ce paramètre peut réduire significativement le temps de calcul d'une métrique basée sur le calcul de chemins, mais le résultat sera moins précis.
- `param1=val` : paramètre(s) de la métrique, si elle en a. Les intervalles ne sont pas permis pour cette commande.
- `patch=id1,id2,...,idn` : enlève les taches listées par leur identifiant
- `fpatch=file.txt` : enlève les taches listées dans le fichier *file.txt*. Le fichier doit contenir un identifiant par ligne.
- `link=id1,id2,...,idn` : enlève les liens listés par leur identifiant
- `flink=file.txt` : enlève les liens listés dans le fichier *file.txt*. Le fichier doit contenir un identifiant par ligne.

Description

Pour chaque graphe sélectionné, enlève l'ensemble des noeuds et/ou liens listés et calcule la métrique globale donnée. La liste des identifiants peut être donnée directement sur la ligne de commande ou bien dans un fichier séparé.

Le résultat est seulement affiché à l'écran. Pour chaque graphe, la commande affiche le nombre de taches et de liens réellement enlevés du graphe. Le nombre de taches ne varient pas, mais le nombre de liens peut varier car les liens connectés à une tache enlevée sont aussi enlevés. Si un identifiant n'existe pas, il sera ignoré.

Exemple

```
--gremove NC patch=2,3
```

Cette commande calcule la métrique NC sur chaque graphe sélectionné après avoir enlevé les taches 2 et 3.

Résultat affiché :

```
Global indice NC
Graph 2000m-3500cost
Remove 2 patches and 5 links
NC : 25.0

Graph 2000m-3500cost_comp
Remove 2 patches and 8 links
NC : 24.0

Graph 2000m_euclid
Remove 2 patches and 7 links
```

```
NC : 9.0

Graph 2000m_euclid_comp
Remove 2 patches and 9 links
NC : 9.0
```

La même commande peut être écrite comme suit :

```
--gremove NC fpatch=patch.txt
```

avec un fichier patch.txt dans le répertoire courant, contenant un identifiant par ligne :

```
2
3
```

2.6.3 –gtest : suppression et ajout itératif d'éléments

`--gtest` *`nstep global_metric_name`* `[maxcost=`*`valcost`*`]` `[`*`param1=val ...`*`]` `obj=patch|link` `sel=`*`id1,id2,...,idn`* `|` `fsel=`*`file.txt`*

Paramètres obligatoires

- `nstep` : nombre d'éléments successifs à ajouter
- `global_metric_name` : acronyme de la métrique globale (PC, ...)
- `obj=patch|link` : type d'éléments à tester, taches (`obj=patch`) ou liens (`obj=link`)
- `sel=id1,id2,...,idn` : liste des identifiants des éléments (taches ou liens) à tester
- `fsel=file.txt` : fichier contenant les identifiants des éléments (taches ou liens) à tester. Le fichier *file.txt* doit contenir un identifiant par ligne.

Paramètres optionnels

- `maxcost=valcost` : limite le calcul de chemin à *valcost* (ie. les chemins supérieurs à *valcost* ne seront pas calculés). Ce paramètre peut réduire significativement le temps de calcul d'une métrique basée sur le calcul de chemins, mais le résultat sera moins précis.
- `param1=val` : paramètre(s) de la métrique, si elle en a. Les intervalles ne sont pas permis pour cette commande.

Description

Cette commande commence par supprimer les éléments (taches ou liens) sélectionnés par le paramètre `sel` ou `fsel`. Puis elle remet l'élément supprimé qui maximise la métrique *global_metric_name*. La remise d'un élément est répétée autant de fois que le paramètre *nstep*. L'ensemble du traitement est effectué pour chaque graphe sélectionné. Pour chacun de ces graphes, les résultats sont enregistrés dans 2 fichiers textes. Le premier liste les éléments ajoutés avec la valeur de la métrique correspondante. Le second, plus détaillé, liste à chaque étape l'ajout de chaque élément.

Exemple

```
--gtest 3 IIC obj=patch sel=1,2,3,5,6,10,15,16
```

Cette commande enlève les 8 taches sélectionnées par leur identifiant et remet successivement les 3 taches qui maximisent la métrique globale IIC.

Pour le graphe *2000m_euclid*, le premier fichier se nomme gtest-2000m_euclid-IIC.txt et contient :

```
Step    Id        IIC
0        init    1.7825075712618167E-6
1        1        1.753327449219068E-6
2        15        1.7793775301389374E-6
3        16        1.780788239231567E-6
```

La première ligne (Step 0), correspond à la valeur de la métrique avant suppression des éléments. Les lignes suivantes donnent, à chaque étape, quel élément a été sélectionné ainsi que la nouvelle valeur de la métrique après l'ajout de l'élément.

Pour le même graphe, le fichier détaillé se nomme gtest-2000m_euclid-IIC-detail.txt et contient :

```
Step        Id        IIC
1        init        1.6852479916976776E-6
1        16        1.6866587007903073E-6
1        1        1.753327449219068E-6
1        2        1.685330641324248E-6
1        3        1.6852916014495828E-6
1        5        1.6858693558888445E-6
1        6        1.6852486177082585E-6
1        10        1.6854627861279089E-6
1        15        1.6994856702980223E-6
2        init        1.7533274492190677E-6
2        16        1.7547381583116972E-6
2        2        1.753410098845639E-6
2        3        1.7538298458957464E-6
2        5        1.753948813410235E-6
2        6        1.7533280752296487E-6
2        10        1.753542243649299E-6
2        15        1.7793775301389374E-6
3        init        1.7793775301389372E-6
3        16        1.780788239231567E-6
3        2        1.7794601797655085E-6
3        3        1.7799512085537507E-6
3        5        1.7799988943301041E-6
3        6        1.7795590940743919E-6
3        10        1.7795923245691686E-6
```

2.6.4 –addpatch : ajout itératif de taches

`--addpatch` *npatch global_metric_name* [*param1=val ...*]
`gridres=`*min:inc:max* [`capa=`*capa_file*] [`multi=`*npatch,size*]
`| patchfile=`*file.shp* [`capa=`*capa_field*]

Paramètres généraux

- `npatch` : nombre de taches à ajouter
- `global_metric_name` : acronyme de la métrique globale (PC, ...)
- `param1=val` : paramètre(s) de la métrique, si elle en a. Les intervalles ne sont pas permis pour cette commande.

Description

Cette commande teste successivement l'ajout d'une nouvelle tache parmi un ensemble prédéfini et conserve la tache qui maximise la métrique globale donnée. Ce processus est répété autant de fois que le paramètre

npatch et ce, pour chaque graphe sélectionné. En sortie, un sous répertoire du projet est créé pour chaque graphe, contenant le détail des résultats.

Pour chaque test d'une nouvelle tache, le graphe est recalculé incluant cette nouvelle tache ainsi que les liens reliant cette tache aux autres déjà existantes, ensuite la métrique globale est calculée sur ce nouveau graphe. Quand l'ensemble des taches possibles a été testé, celle maximisant la métrique donnée est ajoutée au projet. Ce processus est itéré jusqu'à obtenir *npatch* nouvelles taches dans le projet.

Les taches créées ont une emprise au sol d'un pixel, si le pixel d'occupation du sol correspondant à la tache à tester est déjà dans la catégorie habitat ou bien est en dehors de la zone d'étude, la tache est ignorée. Les taches à tester peuvent être donnée par l'intermédiaire d'une grille régulière ou bien d'un ensemble de points ou polygones provenant d'un shapefile.

Test sur un jeu de données vectoriel

`--addpatch` *npatch global_metric_name* [*param1=val ...*] `patchfile=`*file.shp* [`capa=`*capa_field*]

Pour chaque élément du shapefile (point ou polygone), le programme teste l'ajout d'une tache sur le(s) pixel(s) couvrant l'objet vectoriel, si le(s) pixel(s) ne sont pas en NoData ou bien déjà dans la classe habitat.

Le paramètre `capa` permet de définir un attribut du shapefile contenant une valeur de capacité pour chaque tache testée. Si le paramètre `capa` n'est pas renseigné, la capacité des nouvelles taches sera de 1.

Test sur une grille régulière

`--addpatch` *npatch global_metric_name* [*param1=val ...*] `gridres=`*min:inc:max* [`capa=`*capa_file*] [`multi=`*npatch*`,`*size*]

Teste l'ajout d'une tache sur une grille régulière dont la résolution est définie par `gridres`.

Le paramètre `capa` permet de spécifier un fichier raster (au format TIFF ou AsciiGrid) donnant une valeur de capacité potentielle en tout point de l'espace. Si la capacité est nulle, aucune tache ne sera testée à cette position. Le fichier raster peut avoir une autre résolution que la grille ou que la carte de paysage. Si le paramètre `capa` n'est pas renseigné, la capacité des nouvelles taches sera de 1.

Le paramètre `multi` permet de tester l'ajout de *npatch* taches simultané, dans un voisinage de rayon *size*gridres*.

Exemples

```
--addpatch 5 IIC gridres=100
```

Ajoute 5 taches maximisant la métrique IIC en testant l'ajout d'une tache tous les 100 mètres. Le sous répertoire *addpatch_n5_graph_IIC_res100_multi1_1* contient les résultats :

- addpatch_graph_IIC.shp : contient les taches ajoutées avec la valeur de la métrique au moment de l'ajout
- addpatch_graph_IIC.txt : contient le nombre de taches ajoutées avec la valeur de la métrique au moment de l'ajout
- links_graph_IIC.shp : contient l'ensemble des liens du graphe
- topo-links_graph_IIC.shp : contient l'ensemble des liens du graphe en version topologique
- detail/ : sous répertoire contenant le détail des tests d'ajout de tache à chaque étape
- detal/detail_i.shp : ensemble des points testés pour l'ajout de la i-ème tache avec la valeur de la métrique en attribut.

La ligne de commande ci-dessous est équivalente à la précédente mais s'exécutera à 3 résolutions différentes (100,200,500). Les résultats seront stockés dans 3 répertoires, un pour chaque résolution.

```
--addpatch 5 IIC gridres=100,200,500
```

Un autre exemple avec un jeu de points :

```
--addpatch 5 IIC patchfile=testpoint.shp
```

Les résultats seront stockés dans le répertoire *addpatch_n5_graph_IIC_shptestpoint.shp*.

Limitations

La commande `--addpatch` ne fonctionne qu'avec des graphes issus de jeu de liens complet et sans distances intra-taches.

Attention, cette commande modifie le nombre de tache du projet, l'exécution de commandes après celle-ci peuvent générer des incohérences. Il est fortement déconseillé d'ajouter des commandes à la suite de la commande `--addpatch`.

Références

[Clauzel et al.(2015a), Foltête et al.(2014)]

2.6.5 –remelem : suppression itérative d'éléments

`--remelem` *nstep global_metric_name* `[maxcost=`*valcost*`]` `[`*param1*`=`*val* `...] obj=patch|link [sel=`*id1,id2,...,idn* `| fsel=`*file.txt*`]`

Paramètres obligatoires

- `nstep` : nombre d'éléments successifs à supprimer
- `global_metric_name` : acronyme de la métrique globale (PC, ...)
- `obj=patch|link` : type d'éléments à supprimer, taches (`obj=patch`) ou liens (`obj=link`)

Paramètres optionnels

- `sel=id1,id2,...,idn` : sélection des identifiants des éléments (taches ou liens) à tester
- `fsel=file.txt` : fichier contenant les identifiants des éléments (taches ou liens) à tester. Le fichier *file.txt* doit contenir un identifiant par ligne.
- `maxcost=valcost` : limite le calcul de chemin à *valcost* (ie. les chemins supérieurs à *valcost* ne seront pas calculés). Ce paramètre peut réduire significativement le temps de calcul d'une métrique basée sur le calcul de chemins, mais le résultat sera moins précis.
- `param1=val` : paramètre(s) de la métrique, si elle en a. Les intervalles ne sont pas permis pour cette commande.

Description

La commande `--remelem` teste la suppression des éléments du graphe sélectionné (taches ou liens selon le paramètre `obj`) comme la commande `--delta`. Puis elle supprime l'élément qui minimise la métrique *global_metric_name*. La procédure est répétée autant de fois que le paramètre *nstep*. Si le paramètre `sel` ou `fsel` est utilisé, le test de suppression ne se fait que sur les éléments sélectionnés. L'ensemble du traitement est effectué pour chaque graphe sélectionné. Pour chacun de ces graphes, le résultat est enregistré dans un fichier texte qui liste les éléments supprimés avec la valeur de la métrique correspondante.

Exemple

```
--remelem 3 IIC obj=patch
```

Cette commande supprimera successivement les 3 taches qui minimisent la métrique globale IIC, pour chaque graphe.

Pour le graphe *2000m_euclid*, le fichier se nomme rempatch-IIC-2000m_euclid.txt et contient :

```
Step    Id      IIC
0       init    1.7825075712618167E-6
1       120     1.070922766241333E-6
2       145     8.001250544646545E-7
3       118     6.05326129866607E-7
```

La première ligne (Step 0), correspond à la valeur initiale de la métrique. Les lignes suivantes donnent, à chaque étape, quel élément a été supprimé ainsi que la nouvelle valeur de la métrique après la suppression de celui-ci.

Référence

[Foltête et al.(2016)]

2.7 Changements d'occupation du sol

2.7.1 –landmod : changements d'occupation du sol

`--landmod zone=`*`filezones.shp`* `id=`*`fieldname`* `code=`*`fieldname`* `[sel=`*`id1,...,idn`*`] [novoronoi]`

Paramètres obligatoires

- `zone=filezones.shp` : shapefile de polygones contenant les changements d'occupation du sol
- `id=fieldname` : nom d'un attribut du shapefile servant à identifier les polygones. Si les valeurs ne sont pas uniques, les polygones ayant le même identifiant seront appliqués en un seul changement.
- `code=fieldname` : nom d'un attribut du shapefile stockant la nouvelle catégorie d'occupation du sol du polygone

Paramètre optionnel

- `sel=id1,...,idn` : liste des identifiants de polygones à traiter. Si ce paramètre n'est pas renseigné tous les polygones du shapefile sont utilisés.
- `novoronoi` : à la création des nouveaux projets, ne calcule pas la topologie planaire. Cette option peut être utilisée pour diminuer les temps de calcul si la topologie planaire n'est pas utilisée dans les commandes suivantes.

Description

Cette commande doit être placée avant les autres commandes de calcul. Elle va dupliquer le projet pour chaque polygone du shapefile et modifier l'occupation du sol couvert par le polygone. Les commandes suivant celle-ci seront exécutées sur chaque projet modifié. Les projets créés ne contiennent pas les jeux de liens ni les graphes du projet initial.

Les projets créés seront nommés par l'identifiant du (ou des) polygones et stockés dans le répertoire du projet.

Exemple

```
--landmod zone=zones.shp id=ID code=CODE --linkset distance=euclid --graph
        --gmetric IIC
```

Pour chaque identifiant de polygones contenus dans le shapefile `zones.shp`, un nouveau projet nommés par la valeur de `ID` sera créé dans un sous répertoire du projet actuel. Sur chacun des ces projets, un jeu de liens en distance euclidienne, suivi d'un graphe seront créés et la métrique IIC sera calculée sur ce graphe. Au final, chaque sous répertoire du projet actuel contiendra un fichier `IIC.txt` contenant la valeur de la métrique tenant compte de chaque changement d'occupation du sol.

Référence

[Sahraoui et al.(2017)]

2.8 Options

2.8.1 -nosave

Cette option empêche l'enregistrement du projet par une commande. Elle est utile quand vous ne voulez pas qu'une commande modifie le projet.

2.8.2 -proc

Défini le nombre de processeurs (ou coeurs) utilisés par Graphab. Par défaut, la ligne de commande utilise la valeur définie dans la fenêtre des préférences. Pour plus de détails, voir la section Parallélisme.

2.8.3 -mpi

Cette option indique que Graphab est utilisé dans un environnement MPI. Pour plus de détails, voir la section Parallélisme.

Chapter 3

Exemples

Tous les exemples qui suivent peuvent être testés avec le projet exemple fourni sur le site.

3.1 Afficher le projet

Affiche les éléments du projet :

```
java -jar graphab-2.6.jar --project sample_project/Project.xml --show
```

Résultat :

```
===== Link sets =====
Complete_Euclid
Complete_cost
Planar_Euclid
Planar_cost

===== Graphs =====
2000m-3500cost
2000m-3500cost_comp
2000m_euclid
2000m_euclid_comp

===== Point sets =====
Presence_absence
```

3.2 Métrique globale à plusieurs distances

Calcule la métrique PC sur le graphe *2000m-3500cost* en faisant varier le paramètre d de 1000 à 5000 par pas de 1000 :

```
java -jar graphab-2.6.jar --project sample_project/Project.xml
    --usegraph 2000m-3500cost --gmetric PC d=1000:1000:5000 p=0.05 beta=1
```

Le résultat est stocké dans le fichier PC.txt dans le répertoire du projet :

```
Graph            d         p       beta     PC
2000m-3500cost   1000.0    0.05    1.0    1.3170910074623973E-6
2000m-3500cost   2000.0    0.05    1.0    1.5884005111333572E-6
2000m-3500cost   3000.0    0.05    1.0    1.7406772135728036E-6
2000m-3500cost   4000.0    0.05    1.0    1.8425812214129719E-6
2000m-3500cost   5000.0    0.05    1.0    1.918332024845314E-6
```

3.3 Métrique globale sur un graphe élagué à plusieurs distances

Crée 6 graphes élagués de 2000 à 2500, à partir du jeu de lien *Complete_cost* et calcule la métrique IIC pour chaque :

```
java -jar graphab-2.6.jar --project sample_project/Project.xml
    --uselinkset Complete_cost --graph threshold=2000:100:2500 --gmetric IIC
```

Le résultat est stocké dans le fichier IIC.txt dans le répertoire du projet :

Graph	IIC
thresh_2000.0_Complete_cost	1.3740319506984741E-6
thresh_2100.0_Complete_cost	1.3742497768764619E-6
thresh_2200.0_Complete_cost	1.3749834046381206E-6
thresh_2300.0_Complete_cost	1.3754601882078134E-6
thresh_2400.0_Complete_cost	1.3857662689627643E-6
thresh_2500.0_Complete_cost	1.4197576370824857E-6

3.4 Séquence SDM complète

Calcule un graphe sans élagage à partir du jeu de lien *Planar_Euclid*, calcule 2 métriques locales (Dg et F) sur le graphe créé, ajoute le jeu de point sur le jeu de lien *Planar_Euclid* et calcule le SDM pour les 2 métriques avec la variable PRESENCE, sans modifier le projet.

```
java -jar graphab-2.6.jar -nosave --project sample_project/Project.xml
    --uselinkset Planar_Euclid
    --graph
    --lmetric Dg --lmetric F d=1000 p=0.05 beta=1
    --pointset sample_project/Exo-Presence_absence.shp
    --model PRESENCE distW=1000
```

Le résultat est stocké dans le fichier model-PRESENCE-dW1000.txt dans le répertoire du projet :

Graph	Metric	DistWeight	R2	AIC	Coef
comp_Planar_Euclid	Dg	1000.00	0.999744	2.01795	24.8994
comp_Planar_Euclid	F_d1000_p0.05_beta1	1000.00	0.108353	64.6026	2.38405e-05

Chapter 4

Performances

4.1 Parallélisme

4.1.1 Un ordinateur : threads

Si votre ordinateur a plus qu'un coeur (la plupart), vous pouvez tirer partie de la parallélisation pour accélérer vos calculs avec Graphab. Il faut définir le nombre de coeurs (ou processeurs) que Graphab peut utiliser avec l'option `-proc` après la commande `--project` :

```
java -jar graphab-2.6.jar -proc 8 --project path2myproject/myproject.xml ...
```

Par défaut, en ligne de commande, le nombre de coeurs utilisés correspond au nombre défini dans la fenêtre Préférences de l'interface graphique.

En augmentant le nombre de coeurs utilisés par Graphab, vous augmentez, par la même occasion, la taille de la mémoire utilisée par Graphab.

4.1.2 Cluster : MPI

Graphab peut aussi être utilisé sur des clusters de calcul supportant Java avec OpenMPI.

```
mpirun java -jar graphab-2.6.jar -mpi --project path2myproject/myproject.xml ...
```

Seulement certaines commandes peuvent être utilisées dans l'environnement MPI : `--gmetric`, `--cmetric`, `--lmetric`, `--delta`, `--addpatch`,`--gtest`,`--remelem`,`--landmod`

4.2 Gestion mémoire

Par défaut, la taille de la mémoire utilisable par Graphab est dépendante du système et peut varier de 128 MB à plusieurs GB Si vous avez un gros projets, certaines commandes seront lentes voir même stopperont le programme à cause d'un manque de mémoire. Si l'exécution se termine par une erreur OutOfMemoryError ou GC overhead, vous devez augmenter la mémoire disponible pour Graphab.

Pour définir manuellement la mémoire maximale utilisable par Graphab, il faut utiliser l'option Java `-Xmx` :

```
java -Xmx4g -jar graphab-2.6.jar ... # 4Gb allocated
java -Xmx1500m -jar graphab-2.6.jar ... # 1500 Mb -> 1.5Gb allocated
```

Si vous ne pouvez pas allouer plus de 1 GB ou 1.5 GB alors que votre ordinateur a plus de mémoire vive, vous utilisez sûrement une version 32-bit de Java qui est limitée à moins de 2 GB. Pour tester votre version de Java :

```
java -version
```

Si vous avez une version 32-bit, installez une version 64-bit de Java pour allouer plus de mémoire à Graphab.

Bibliography

[Clauzel et al.(2015a)] Celine Clauzel, Cyrielle Bannwarth, and Jean-Christophe Foltete, 2015a. Integrating regional-scale connectivity in habitat restoration: An application for amphibian conservation in eastern france. *Journal for Nature Conservation*, 23 98 – 107.

[Clauzel et al.(2015b)] Celine Clauzel, Deng Xiqing, Wu Gongsheng, Patrick Giraudoux, and Li Li, 2015b. Assessing the impact of road developments on connectivity across multiple scales: Application to yunnan snub-nosed monkey conservation. *Biological Conservation*, 192 207 – 217.

[Clauzel et al.(2013)] Céline Clauzel, Xavier Girardet, and Jean-Christophe Foltête, 2013. Impact assessment of a high-speed railway line on species distribution: Application to the european tree frog (hyla arborea) in franche-comté. *Journal of Environmental Management*, 127 125 – 134.

[Foltête and Vuidel(2017)] Jean-Christophe Foltête and Gilles Vuidel, 2017. Using landscape graphs to delineate ecologically functional areas. *Landscape Ecology*, 32(2) 249–263.

[Foltête et al.(2012a)] Jean-Christophe Foltête, Céline Clauzel, and Gilles Vuidel, 2012a. A software tool dedicated to the modelling of landscape networks. *Environmental Modelling and Software*, 38 316 – 327.

[Foltête et al.(2012b)] Jean-Christophe Foltête, Céline Clauzel, Gilles Vuidel, and Pierline Tournant, 2012b. Integrating graph-based connectivity metrics into species distribution models. *Landscape Ecology*, 27(4) 557–569.

[Foltête et al.(2016)] Jean-Christophe Foltête, Geoffroy Couval, Marilyne Fontanier, Gilles Vuidel, and Patrick Giraudoux, 2016. A graph-based approach to defend agro-ecological systems against water vole outbreaks. *Ecological Indicators*, 71 87 – 98.

[Foltête et al.(2014)] Jean-Christophe Foltête, Xavier Girardet, and Céline Clauzel, 2014. A methodological framework for the use of landscape graphs in land-use planning. *Landscape and Urban Planning*, 124 140 – 150.

[Girardet et al.(2013)] Xavier Girardet, Jean-Christophe Foltête, and Céline Clauzel, 2013. Designing a graph-based approach to landscape ecological assessment of linear infrastructures. *Environmental Impact Assessment Review*, 42 10 – 17.

[Newman(2006)] M. E. J. Newman, 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23) 8577–8582.

[Sahraoui et al.(2017)] Yohan Sahraoui, Jean-Christophe Foltête, and Céline Clauzel, 2017. A multi-species approach for assessing the impact of land-cover changes on landscape connectivity. *Landscape Ecology*, 32(9) 1819–1835.