TOPO

Appel au module TOPO

- `trafiPolluImp.py`:

```
def _dump_topo_export_(self):
 # DUMP
 ...
 # EXPORT
 # traitement de la requete d'export [EXPORT]
 self.module_export.export(True)
```

- `trafiPolluImp_EXPORT`:

```
def export(self, update_symu=False, outfilename=""):
 ...
 self.update_SYMUVIA()
def update_SYMUVIA(self):
 self.module_topo.build_topo()
 #
 self.update_TRONCONS()

 if b_export_connexion:
  self.update_CONNEXIONS()
  self.update_TRAFICS()
```

  - `module_TOPO` définit dans `trafiPolluImp`:
    `self.module_TOPO = tpi_TOPO.trafipolluImp_TOPO(**kwargs)`
    - `trafipolluImp_TOPO` dans 'trafipolluImp_TOPO'

- `trafipolluImp_TOPO`

  - `build_topo`:

```
def build_topo(self):
    self.build_topo_for_troncons()
    self.build_topo_for_interconnexions()
    self.build_topo_extrimites()
```

TRONCONS

- `build_topo_for_troncons`: Lance la conversion topologique des edges dumpées depuis le serveur StreetGen vers des troncons SYMUVIA
  - fonction clée: `build_pyxb_symutroncon_from_sg3_edge`, c'est dans cette fonction que se situe la logique algorithmique de conversion topologique pour les edges ->

troncons

- informations issues du DUMP utilisées:
  - `edge` StreetGen:
    - `group_lanes`: informations (liées à la TOPOlogie) sur les groupes de voies homogènes au sens SYMUVIA (uni-directionnelle)
- `group_lanes` contient des éléments TOPOlogiques important ! Du à un pb de design, et de non refactorisation du code, `group_lanes` ou sa construction si situe dans le module DUMP (au lieu d'être en TOPO).
  - `dump_lanes` dans `trafipolluImp_DUMP`
    - qui appelle `build_dict_grouped_lanes`-> algo fonctionnel (pas une bonne idée après coup de rajouter de la complexité à ce moment du dev/design ...) pour regrouper les voies uni-directionnelles en packet.

  => `group_lanes` contient une liste de nombre de voies consécutives uni-directionnelles (pour une `edge` StreetGen)
- dans `build_pyxb_symutroncon_from_sg3_edge` fonction importante => `_build_pyxb_symutroncon_from_sg3_edge_lane`
  - Construit une entité `typeTroncon` (issue du parser XSD via PyXB)
  - Fonction de construction utilisée: `build_pyxb_symuTroncon_with_lanes_in_groups`

    Cette fonction est le reliquat d'un mécanisme générique de construction des pyxb_symutroncon.

    Initialement il y avait plusieurs fonctions du type: `build_pyxb_symuTroncon_with_*`, car je pensais qu'il y avait plusieurs cas (topologiques) à traiter.

    Au final, la version utilisée est suffisament générique pour traiter tous les cas (à priori). Cependant le mécanisme générique est resté (à retirer/simplifier après refactorisation)

    - `build_pyxb_symuTroncon_with_lanes_in_groups`: s'occupe de la construction GEOmétrique de du troncon.

      Il y a une problématique de simplication ou discrétisation géométrique à ce niveau. Le niveau de précision ou l'atomicité des entités manipulées par StreetGen et SYMUVIA ne sont pas du même ordre/même échelle:

      - StreetGen: l'unité atomique est la voie qui possède une géométrie propre (segment axe 2D).
      - SYMUVIA: l'uniité atomique est le troncon (qui possède un segment axe 2D). Les voies/lanes des troncons sont définies

géomètriquement par le troncon auquel elles se rattachent.

INTERCONNEXIONS

- `build_topo_for_interconnexions`: Lance la conversion topologique des interconnexions. Les interconnexions au sens StreetGen/SYMUVIA sont les connexions présentes dans les zones d'intersections StreetGen/connexions SYMUVIA entre les amonts/avals des troncons/voies/edges connectés.

On utilise `dict_nodes` pour récupérer les informations dumpées à propros des nodes.

Dans la méthode, il y a une partie de simplification géométrique des connexions.