TRANSPORT & MOBILITY LEUVEN

Your link to integrated analyses !

TRANSPORT & MOBILITY LEUVEN
VITAL DECOSTERSTRAAT 67A BUS 0001
3000 LEUVEN
BELGIË

http://www.tmleuven.be/

TEL +32 (16) 31 77 30
FAX +32 (16) 31 77 39

Auteur: Sven Maerivoet
Laatste wijziging: 5 januari 2009

Cube — LTM Intersection Converter Methodologie

Deliverable voor WP 2.13

1. Algemene opzet

1.1 Overzicht conversieproces

Merk op dat we bij de conversie uitgaan van een 1-op-1 match tussen het LTM netwerk en het oorspronkelijke Cube/OmniTRANS netwerk.

Inladen Cube en LTM netwerkdefinities:

- *loadCubeNodes()*
- *loadLTMNetwork()*
- *loadLTMNodeTypeExceptions()*
- *loadNodeIDTranslationTable()*
- *loadCubeJunctions()*
- *loadCubeMissingApproaches()*

Aanmaken LTM netwerk:

- *redefineLTMNodes()*
- *createLTMMovements()*
- *createLTMLanes()*
- *createLTMSignals()*

Bewaren LTM netwerk:

- *storeLTMNodes()*
- *storeLTMMovements()*
- *storeLTMLanes()*
- *storeLTMSignals()*

1.2 Uitvoeren programma

De intersection converter is een Java programma dat de Java VM 1.6 vereist. Uitvoeren ervan gaat via volgende opdracht:

```
java -classpath %CLASSPATH%;smtools.jar flexsys.ltm.IntersectionConverter
```

Indien men de status- en foutmeldingen wilt bewaren, voegt men "`> run-log.txt`" toe aan voorgaande opdracht.

In de folder waar het programma uitgevoerd wordt, dienen volgende subfolders en bestanden aanwezig te zijn:

```
/data/input/cube-junctions.txt
/data/input/cube-missing-approaches.txt
/data/input/cube-nodes.txt
/data/input/ltm-network.txt
/data/input/ltm-node-type-exceptions.txt
/data/input/node-ids-cube-ltm-translation-table.txt
/data/ouput/
/data/output/
```

Nadat het programma is uitgevoerd, zijn volgende bestanden aangemaakt:

```
/data/output/ltm-nodes.txt
/data/ouput/ltm-movements.txt
/data/output/ltm-lanes.txt
/data/output/ltm-signals.txt
```

2. Inladen Cube en LTM netwerkdefinities

2.1 loadCubeNodes()

De file `/data/input/cube-nodes.txt` bevat volgende structuur:

nodeID `<int>`, xCoordinate `<double>`, yCoordinate `<double>`

2.2 loadLTMNetwork()

De file `/data/input/ltm-network.txt` dient volgende twee secties te bevatten (de andere secties zoals timeslices, routes en odmatrix worden niet gebruikt door het programma):

`nodes=`
nodeID `<int>`, xCoordinate `<double>`, yCoordinate `<double>` [, type[1] `<int>`]

`links=`
linkID `<int>`, fromNodeID `<int>`, toNodeID `<int>`,
jamDensity `<int>`, capacity `<int>`, freeFlowSpeed `<double>`, length `<double>`

Links waarvan de beginNodeID nul is, zijn *Origin-links*; indien de toNodeID nul is, dan zijn het *Destination*-links.

2.3 loadLTMNodeTypeExceptions

De file `/data/input/ltm-node-type-exceptions.txt` bevat een lijst met handmatige uitzonderingen op de LTM node types; zij heeft volgende structuur:

ltmNodeID `<int>`, ltmType `<int>`

Met ltmType gedefinieerd zoals in Sectie 2.2.

2.4 loadNodeIDTranslationTable()

De file `/data/input/node-ids-cube-ltm-translation-table.txt` bevat volgende structuur:

cubeNodeID `<int>`, ltmNodeID `<int>`

Het kan dat voor een cubeNodeID de corresponderende ltmNodeID niet gegeven is; dit maakt dat er typisch meer Cube nodes dan LTM nodes in een netwerk zijn.

[1] Het type node kan een van volgende zijn:

(1) *Signalised intersection.*
(2) *Unsignalised intersection.*
(3) *Demand-proportional intersection.*
(4) Roundabout.

2.5 loadCubeJunctions()

De file `/data/input/cube-junctions.txt` bevat de Cube junctions zoals ze geëxporteerd werden.

In Cube bestaan er drie types junction: *TwoWayStop*, *AdaptiveSignal* en *Roundabout*.

2.5.1 Cube TwoWayStop junction

```
JUNCTION,
  NODE = <int>,
  TYPE = TwoWayStop,
  APPROACH1 = <int>,
  ENABLE = <int>,
  FOURLANEMAJOR = <int>,

  APPROACH = <int>,
    RANDOMNESS = <double>,
    AVELANEWIDTH = <double>,
    EXITLANES = <int>,
    EXITONLY = <int>,
    SINGLELANE = <int>,
    MOVEMENT = Left,
      BANTURN = <int>,
      CRITICALGAP = <double>,
      FOLLOWUPTIME = <double>,
    MOVEMENT = Right,
      BANTURN = <int>,
      CRITICALGAP = <double>,
      FOLLOWUPTIME = <double>,
    ...
  ...
```

2.5.2 Cube AdaptiveSignal junction

```
JUNCTION,
  NODE = <int>,
  TYPE = AdaptiveSignal,
  APPROACH1 = <int>,
  ENABLE = <int>,
  LANEADJUST = <int>,
  CYCLETIME = <double>,
  MINIMUM = <double>,
  MAXIMUM = <double>,
  PHASE = <int>,
    ACTUALGREEN = <double>,
    MINIMUM = <double>,
    MAXIMUM = <double>,
  PHASE = <int>,
    ACTUALGREEN = <double>,
    MINIMUM = <double>,
    MAXIMUM = <double>,
  ...
  APPROACH = <int>,
    AVELANEWIDTH = <double>,
    RANDOMNESS = <double>,
    EXITLANES = <int>,
    EXITONLY = <int>,
    MOVEMENT = Left,
      PHASES = <int>,
      EXCLUSIVELANES = <int>,
    MOVEMENT = Through,
      PHASES = <int>,
      EXCLUSIVELANES = <int>,
      BANTURN = <int>,
```

```
    CANSHARERIGHT = <int>,
  MOVEMENT = Right,
    PHASES = <int>,
    EXCLUSIVELANES = <int>,
    CANSHARELEFT = <int>,
...
```

2.5.3 Cube Roundabout junction

```
JUNCTION,
  NODE = <int>,
  TYPE = Roundabout,
  APPROACH1 = <int>,
  ENABLE = <int>,
  GAPACCEPTANCE = 0,
  APPROACH = <int>,
    CAPACITYINTERCE = <double>,
    CAPACITYSLOPE = <double>,
    RANDOMNESS = <double>,
    EXITONLY = <int>,
  ...
```

2.6 loadCubeMissingApproaches()

De file `/data/input/cube-missing-approaches.txt` bevat een lijst met handmatige uitzonderingen voor elke Cube APPROACH die niet gevonden kan worden bij een Cube junction; zij heeft volgende structuur:

ltmIntersectionID `<int>`, cubeJunctionID `<int>`, upstreamLTMNodeID `<int>`, oldCubeApproachID `<int>`, newCubeApproachID `<int>`

3. Aanmaken LTM netwerk

3.1 redefineLTMNodes()

Vermits in principe bij het inlezen van het LTM netwerk niet geweten is wat het types van elke node is, wordt in deze stap voor elke LTM node gekeken of er informatie gevonden kan worden bij de overeenkomstige Cube junction.

Methode:

- Scan alle LTM nodes in het netwerk.
- Zoek de overeenkomstige Cube junction:
  - Indien deze bestaat:
    - Stel het LTM node type in volgens dat van de Cube junction.
  - Indien deze niet bestaat:
    - Stel het LTM node type in op demand proportional.
- Indien een LTM node type uitzondering bestaat, pas deze dan toe en wijzig het node type.

Het resultaat is een nieuwe `nodes=` sectie.

3.2 createLTMMovements()

De LTM movements worden in eerste instantie louter op basis de informatie uit het LTM netwerk (*/data/input/ltm-network.txt*) aangemaakt. Hierbij wordt voor elke intersectie gekeken naar alle inkomende en uitgaande links; alle combinaties hiervan vormen de LTM movements.

In een tweede stap wordt gekeken naar de informatie van de overeenkomstige Cube junctions (*/data/input/cube-junctions.txt*). Op basis van deze informatie worden de voorrangsregelingen van de verschillende assen opgesteld.

In een optionele derde stap worden niet-toegestane movements weggefilterd, dit op basis van de EXITONLY-tags uit de Cube junctions. In principe is deze stap niet nodig aangezien de enige resterende LTM links unidirectioneel zijn, en verboden richtingen reeds bij het opstellen van het netwerk weggefilterd werden.

In een vierde stap wordt voor elke LTM movement bepaald of het een linkse, rechtdoor of rechtse movement betreft, dit op basis van het aantal Cube APPROACH-tags in de overeenkomstige Cube junction.

Methode:

- **Opstellen exhaustieve combinaties movements (enkel voor signalised en unsignalised LTM nodes):**
  - Scan alle LTM nodes in het netwerk.
  - Verzamel alle inkomende en uitgaande links.
  - Maak alle toegelaten combinaties voor alle movements (heen en terug links over dezelfde node paren worden niet toegestaan, dit zou namelijk een U-turn op het kruispunt betekenen).

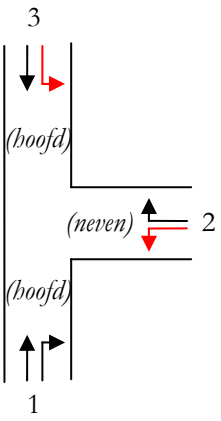
- **Instellen voorrangsregelingen voor alle movements:**
  - Beschouw een LTM intersectie.
  - Zet LTM intersection/node ID om naar een Cube junction ID.
  - Zoek de overeenkomstige Cube junction.
  - Zet de voorrangsregeling voor de hoofdas (i.e., 1e en 3e APPROACH-tags) op nul.
  - Zet de voorrangsregeling voor de nevenassen (i.e., 2e, 4e en hogere APPROACH-tags) gelijk aan de movement IDs van de hoofdas.

  ➔ *Indien tijdens deze stap geen Cube APPROACH-tag kan gevonden worden die overeenkomt met de LTM node, dan wordt een waarschuwing gegenereerd. Het idee is dat deze 'ontbrekende' APPROACH-tag manueel wordt aangereikt (zie ook Sectie 2.6 waar de manueel aangereikte tags worden ingelezen; er wordt bij het instellen van de voorrangsregeling wel nagekeken of de tag manueel beschikbaar is gesteld).*

- **(OPTIONEEL) Wegfilteren niet-toegestane movements:**
  - Scan alle LTM nodes in het netwerk.
  - Beschouw alle movements voor de huidige LTM node.
  - Beschouw de inkomende link voor de huidige movement.
  - Zoek de overeenkomstige Cube junction en APPROACH voor de begin node van de inkomende link.
  - Als de EXITONLY-tag van de Cube APPROACH gezet is, voeg dan de movement toe aan de wegwerplijst.
  - Beschouw alle movements voor de huidige LTM node.
    - Als de movement deel uitmaakt van de wegwerplijst, verwijder ze dan uit de lijst van movements.
    - Anders, verwijder alle movements uit de wegwerplijst uit alle conflicterende voorrangsmovement IDs.

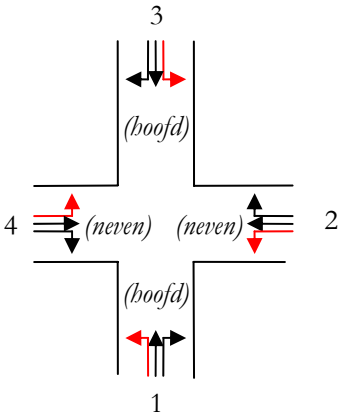
  Merk op dat indien een movement geschrapt wordt, de IDs van de overige movements niet aangepast worden, waardoor deze niet meer aaneensluitend zijn.

- **Bepalen van de soort movement (links, rechtdoor of rechts):**
  - Scan alle LTM nodes in het netwerk.
  - Zet LTM intersection/node ID om naar een Cube junction ID.
  - Zoek de overeenkomstige Cube junction.
    - Scan alle LTM movements.
    - Beschouw de LTM movements voor de huidige LTM node.
    - Kijk telkens naar de inkomende links van deze LTM movements.
    - Kijk telkens naar de begin node van deze LTM movements.
    - Als beide gevonden werden in Cube, dan wordt de richting van de LTM movement als volgt bepaald:



- Als het aantal Cube APPROACH-tags = 3:
  - *begin = 1, einde = 2* ➔ *rechtsaf.*
  - *begin = 1, einde = 3* ➔ *rechtdoor.*
  - *begin = 2, einde = 1* ➔ *linksaf.*
  - *begin = 2, einde = 3* ➔ *rechtsaf.*
  - *begin = 3, einde = 1* ➔ *rechtdoor.*
  - *begin = 3, einde = 2* ➔ *linksaf.*

  ➔ Merk op dat beide linksaf-bewegingen voorrang dienen te verlenen:
  - Indien de movement van de <u>minor-arm</u> vertrekt (dus begin = 2 en einde = 1), dan is dit al automatisch in orde.
  - Indien de movement van een <u>major-arm</u> vertrekt (dus begin = 3 en einde = 2), dan dient deze voorrang te verlenen aan de rechtsaf-movement met begin = 1 en einde = 2, en de rechtdoor-movement met begin = 1 en einde = 3.



- Als het aantal Cube APPROACH-tags = 4:
  - *begin = 1, einde = 2* ➔ *rechtsaf.*
  - *begin = 1, einde = 3* ➔ *rechtdoor.*
  - *begin = 1, einde = 4* ➔ *linksaf.*
  - *begin = 2, einde = 1* ➔ *linksaf.*
  - *begin = 2, einde = 3* ➔ *rechtsaf.*
  - *begin = 2, einde = 4* ➔ *rechtdoor.*
  - *begin = 3, einde = 1* ➔ *rechtdoor.*
  - *begin = 3, einde = 2* ➔ *linksaf.*
  - *begin = 3, einde = 4* ➔ *rechtsaf.*
  - *begin = 4, einde = 1* ➔ *rechtsaf.*
  - *begin = 4, einde = 2* ➔ *rechtdoor.*
  - *begin = 4, einde = 3* ➔ *linksaf.*

  ➔ Merk op dat alle vier de linksaf-bewegingen voorrang dienen te verlenen:
  - Indien de movement van een <u>minor-arm</u> vertrekt dan is dit al deels in orde. Er dient echter nog extra voorrang verleend te worden:
    - Als begin = 2 en einde = 1: aan de movements met begin = 4 en eindes = 1, 2 en 3.
    - Als begin = 4 en einde = 3: aan de movements met begin = 2 en eindes = 1, 3 en 4.
  - Indien de movement van een <u>major-arm</u> vertrekt, dan dient deze als volgt voorrang te verlenen:
    - Als begin = 1 en einde = 4: aan de movements met begin = 3 en eindes = 1 en 4.
    - Als begin = 3 en einde = 2: aan de movements met begin = 1 en eindes = 2 en 3.

➔ *Indien tijdens deze stap geen Cube APPROACH-tag kan gevonden worden die overeenkomt met de LTM node, dan wordt een waarschuwing gegenereerd. Het idee is dat deze 'ontbrekende' APPROACH-tag manueel wordt aangereikt (zie ook Sectie 2.6 waar de manueel aangereikte tags worden ingelezen; er wordt bij het instellen van de voorrangsregeling wel nagekeken of de tag manueel beschikbaar is gesteld).*

- Indien er voor de movement in kwestie een BANTURN-tag is gezet, dan wordt deze movement weggefilterd.
  Merk op dat indien een movement geschrapt wordt, de IDs van de overige movements <u>niet</u> aangepast worden, waardoor deze niet meer aaneensluitend zijn.

3.3 createLTMLanes()

Op basis van de informatie uit het LTM netwerk (*/data/input/ltm-network.txt*) wordt gekeken naar het type van elke intersectie. Vervolgens wordt het aantal rijstroken bepaald met behulp van de informatie uit de overeenkomstig Cube junctions (*/data/input/cube-junctions.txt*), dit afhankelijk van het type Cube junction. De eerder opgestelde movements worden vervolgens gekoppeld aan de rijstroken.

Methode:

- Itereer over all *signalised* en *unsignalised* LTM intersecties/nodes.
- Zoek de overeenkomstige Cube junction.
- Scan alle APPROACH-tags van de Cube junction, in volgorde.
  ➔ *Tijdens deze stap kan een 'ontbrekende' APPROACH-tag manueel worden aangereikt (zie ook Sectie 2.6 waar de manueel aangereikte tags worden ingelezen).*
- Bekijk het type Cube junction:
  - TwoWayStop:
    - Bepaal het aantal rijstroken voor de huidige APPROACH:
      - Als het de eerste of derde Cube APPROACH is (*dit is een hoofdas*):
        - Als FOURLANEMAJOR van de Cube junction gelijk is aan 0:
          ➔ #rijstroken = 1.
        - Als FOURLANEMAJOR van de Cube junction gelijk is aan 1:
          ➔ #rijstroken = 2.
      - Als het de tweede of vierde of hogere Cube APPROACH is (*dit is een nevenas*):
        - Als SINGLELANE van de Cube APPROACH gelijk is aan 0:
          ➔ #rijstroken = 2.
        - Als SINGLELANE van de Cube APPROACH gelijk is aan 1:
          ➔ #rijstroken = 1.

      De EXITLANES- and EXITONLY-tags worden genegeerd aangezien zij niet van toepassing zijn.
    - Itereer over alle LTM movements.
    - Beschouw enkel de LTM movements die vertrekken vanaf de huidige inkomende link.
      - Als het aantal rijstroken voor een APPROACH gelijk is aan 1:
        - Ken al deze LTM movements toe aan de rijstrook in kwestie. Kies als capaciteit de hoogste (1500 PAE/uur voor afslaande bewegingen, 1800 PAE/uur voor rechtdoorgaande bewegingen).
      - Als het aantal rijstroken voor een APPROACH gelijk is aan 2:
        - Verdeel deze LTM movements over de twee rijstroken:
          - Linksaf en rechtdoor gaan naar de meest linkse rijstrook (= 2).
          - Rechtdoor en rechtsaf gaan naar de meest rechtse rijstrook (= 1).

- AdaptiveSignal:
  - Bepaal het aantal rijstroken voor de huidige APPROACH:
    - Als SINGLELANE van de Cube APPROACH gelijk is aan 1:
      ➔ #rijstroken = 1.
      - Itereer over alle LTM movements.
      - Beschouw enkel de LTM movements die vertrekken vanaf de huidige inkomende link.
      - Ken al deze LTM movements toe aan de rijstrook in kwestie. Kies als capaciteit de hoogste (1500 PAE/uur voor afslaande bewegingen, 1800 PAE/uur voor rechtdoorgaande bewegingen).
    - Als SINGLELANE van de Cube APPROACH gelijk is aan 0 of niet gedefineerd werd:
      - Itereer over alle LTM movements.
      - Beschouw enkel de LTM movements die vertrekken vanaf de huidige inkomende link.
        - Als EXCLUSIVELANES groter dan of gelijk is aan 1:
          ➔ 1 exclusieve rijstrook voor deze movement.
        - Als EXCLUSIVELANES gelijk is aan 0 of niet gezet werd:
          ➔ 1 rijstrook voor alle movements waarvoor CANSHARELEFT of CANSHARERIGHT gelijk is aan 1.
      - Ken al deze LTM movements toe aan de rijstroken in kwestie. Kies als capaciteit de hoogste (1500 PAE/uur voor afslaande bewegingen, 1800 PAE/uur voor rechtdoorgaande bewegingen). Indien EXCLUSIVELANES strikt groter is dan 1, vermenigvuldig dan de capaciteit van de LTM movement met het aantal exclusieve rijstroken voor de movement in kwestie.

    ➔ #rijstroken >= de som van het aantal exclusieve movements.

3.4 createLTMSignals()

Op basis van de informatie uit de Cube junctions (*/data/input/cube-junctions.txt*) berekenen we de totale cyclustijd, en de individuele tijden voor de rode en groene phases. Vervolgens worden deze aan de verschillende LTM rijstroken gekoppeld.

Methode:

- Itereer over all *signalised* LTM intersecties/nodes.
- Zoek de overeenkomstige Cube junction.
    - **(OPTIONEEL) Stel het cyclusschema van deze Cube junction op:**
        - Bepaal de totale cyclustijd als volgt:

$$TCT = \text{MAXIMUM van de CYCLETIME}$$

        - Bepaal de totale groentijd als volgt:

$$TGT = \sum_{\text{PhaseID}} \text{MAXIMUM van de PHASE}$$

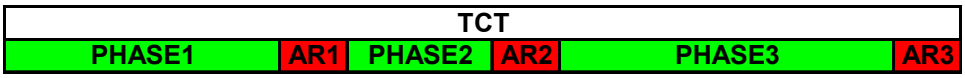
        - Bepaal de totale roodtijd als volgt:

$$TRT = TCT - TGT$$

        - Bij N phases is de all-roodtijd na elke phase p gelijk aan:

$$ARp = TRT / N$$

        - In principe worden de phases als volgt verdeeld (stel er zijn 3 phases):



    ➔ *Merk op dat we enkel de MAXIMUM tijden voor de Cube phases beschouwen.*

    - Vermits we de richting van de movement weten, hebben we ook beschikking over de bijhorende Cube phases. We scannen nu alle LTM rijstroken voor de huidige intersectie.
    - Vervolgens weten we voor elke rijstrook welke movements hierbij horen.
    - We scannen nu alle LTM movements, kijken welke telkens matchen bij de movements van de rijstrook, en bepalen zo de bijhorende phases.

4. Bewaren LTM netwerk

4.1 storeLTMNodes ()

De file `/data/output/ltm-nodes.txt` bevat volgende sectie::

```
nodes=
nodeID <int>, xCoordinate <double>, yCoordinate <double>, type <int>
```

Het node-type is gedefinieerd zoals in Sectie 2.2.

4.2 storeLTMMovements()

De file `/data/output/ltm-movements.txt` bevat volgende sectie::

```
movements=
nodeID <int>, localMovementID <int>, incomingLinkID <int>, outgoingLinkID <int>
        conflictingPriorityMovementIDs <int; int; …>
```

4.3 storeLTMLanes()

De file `/data/output/ltm-lanes.txt` bevat volgende sectie::

```
lanes=
nodeID <int>, linkID <int>, localLaneID <int>, localMovementIDs <int; int; …>
        capacity <double>
```

4.4 storeLTMSignals()

De file `/data/output/ltm-signals.txt` bevat volgende sectie::

```
signals=
nodeID <int>, phaseID <int>, phaseTime <int>, lanesWithGreen <int; int; …>
```

De phaseTime wordt gegeven in seconden.