pgRouting extension for courses

calculation in a VTMS
AGENDA

• VTMS overview
• Decision Support in a VTMS
• Visibility Graph to model Open Sea
• Geospatial DB and GIS functions to solve SP
• Extended pgRouting for Visibility Graph
• Conclusions
AGENDA

• VTMS overview

• Decision Support in a VTMS

• Visibility Graph to model Open Sea

• Geospatial DB and GIS functions to solve SP

• Extended pgRouting for Visibility Graph

• Conclusions
VTMS
(Vessel Traffic Management System)
VTMS Services

Safe and laws enforcement

- Ships and ports safety monitoring
- Police actions support
- Fishing activities control
- Dangerous loads monitoring

Decision support to human life safeguard at sea

- Accidents prevention
- Environment protection
- Search and Rescue (SAR) support
- Emergency missions planning and monitoring
AGENDA

• VTMS overview

  • Decision Support in a VTMS

   • Visibility Graph to model Open Sea
   • Geospatial DB and GIS functions to solve SP
   • Extended pgRouting for Visibility Graph

• Conclusions
Emergency Missions

- SAR
- Hunting
- Anti-pollution
- SAR
Mission Planning
Mission Execution
AGENDA

• VTMS overview

• Decision Support in a VTMS

  • Visibility Graph to model Open Sea

  • Geospatial DB and GIS functions to solve SP

  • Extended pgRouting for Visibility Graph

• Conclusions
Open sea scenery is characterized by islands, peninsulas, buoys and many other obstacles that can be modeled as POLYGONS.

In this kind of scenery two objects cannot be connected by a single straight line if it intersects obstacles.

Open sea scenery can be reasonably modeled with a “Visibility Graph” where two objects are reachable one each other by a shortest path.

A shortest path is like a course in open sea.
Visibility Graph
Algorithm Input/Output

**Input:** A set of Polygons: P1,..,Pn

**Output:** A weighted graph G = (V,E)
Visibility Graph building Algorithm

- Given $V(G) =$ set of all input polygons vertexes
- Taken $E(G) =$ set of visibility graph edges

For each $u$ in $V(G)$, $v$ in $V(G)$
An edge $e=(u,v)$ is added to $E(G)$ if and only if $e$ doesn’t intersect any of polygons $P_i$ ($i=1,..n$)
Step 1:
Insertion of two input points $p_1$ and $p_2$ into the visibility graph
Step 2:
Shortest Path calculation between \( p1 \) and \( p2 \) by a routing algorithm as Dijkstra
AGENDA

• VTMS overview

• Decision Support in a VTMS

• Visibility Graph to model Open Sea

  • Geospatial DB and GIS functions to solve SP

• Extended pgRouting for Visibility Graph

• Conclusions
The need to analyze, elaborate and store geographic information and geometric data suggested to use **Geographic Information System**
PostgreSQL+Postgis

• PostgreSQL has been selected as object-relational DBMS

• PostgreSQL is supported from PostGIS for geographic data management

• PostGIS defines data types that allow to store spatial information as records of a database table

• PostGIS provides to DBMS functions to manage spatial data
Shortest Path by GIS-based DB

- pgRouting is a C library that provides routing functionality to PostGIS/PostgreSQL

- pgRouting already implements algorithms like the following:
  - Shortest Path Dijkstra, shortest path algorithm with exact result
  - Shortest Path A*, shortest path algorithm with heuristics
  - Traveling Sales Person (TSP)
Extended pgRouting for Visibility Graph
AGENDA

• VTMS overview

• Decision Support in a VTMS

• Visibility Graph to model Open Sea

• Geospatial DB and GIS functions to solve SP

• Extended pgRouting for Visibility Graph

• Conclusions
1. Visibility graph building starting from POLYGON type objects

2. New points insertion into a previously built visibility graph

3. Shortest path calculation by Dijkstra algorithm between two points into a visibility graph
1. `build_visibility_graph`

```sql
SELECT build_visibility_graph('polygons')
```
FUNCTION build_visibility_graph(tablename varchar)

Parameter:

- Name of DB table containing POLYGON type objects
  Constraint: Polygons have to be closed.

Description:

- Build a visibility graph by inserting a visibility edge between each
couple of input polygons points
- Assign an index to each node of built visibility graph
- Calculate length of each inserted edge
2. insert_point_into_graph

```
SELECT insert_point_into_graph('polygons', 'label', point_x, point_y);
```
2. insert_point_into_graph function

Datum insert_point_into_graph(tablename varchar,
label varchar,
p_x float8,
p_y float8)

Parameters:
  a. name of table containing POLYGONS for which visibility graph has been built
  b. label to identify edges to be included for input point p
  c. x and y coordinates of input point p

Description:
• Build visibility edges for input point p, of coordinates (x,y), in the visibility graph built from polygons table
• Assign a label to identify new included edges for point p
• Assign a new index to input point p in the visibility graph
3. shortest_path_into_visibilitygraph

SELECT shortest_path_into_visibilitygraph('polygons', p1_x, p1_y, p2_x, p2_y, 'label');
FUNCTION shortest_path_into_visibilitygraph(tablename varchar, p1_x float8, p1_y float8, p2_x float8, p2_y float8, label varchar)

Parameters:
  a. name of POLYGONS table for which visibility graph has been built
  b. label to identify edges to be included for input points
  c. x and y coordinates of input points p1, p2

Description:
• insert two points in the visibility graph, built from polygons table, by calling insert_point_into_graph function with ‘label’ as argument
• calculate shortest path between p1 and p2 by pgRouting Dijkstra SP function
Current pgRouting deploy

- FindPostgreSQL.cmake
- FindGAUL.cmake
- FindCGAL.cmake
- ...
Extended pgRouting deploy

- FindPostgreSQL cmake
- ... FindPostgis cmake

- driving_distance
- tsp
- visibility_graph

- src
- sql

- librouting.so
- librouting_vg.so
Extended pgRouting: cmake

FindPostgis.cmake

if(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
  set(POSTGIS_FOUND TRUE)
else(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
  FIND_PATH(POSTGIS_INCLUDE_DIR postgis_config.h /usr/local/pgsql/include/server
            /usr/local/include/pgsql/server
            ${PGROUTING_CORE_INCLUDE_DIR}
            $ENV{ProgramFiles}/PostgreSQL/*/include/server
            $ENV{SystemDrive}/PostgreSQL/*/include/server
  )
  FIND_PATH(POSTGIS_INCLUDE_DIR liblwgeom.h
            $POSTGIS_HOME/liblwgeom
  )

  find_library(POSTGIS_LIBRARIES NAMES libpostgis
               postgis
               PATHS
               /usr/lib
               ....
               $ENV{ProgramFiles}/PostgreSQL/*/lib/ms
               $ENV{SystemDrive}/PostgreSQL/*/lib/ms
  )
if(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
  set(POSTGIS_FOUND TRUE)
  message(STATUS "Found Postgis: 
  {POSTGIS_INCLUDE_DIR}, ${POSTGIS_LIBRARIES}"
           INCLUDE_DIRECTORIES(${POSTGIS_INCLUDE_DIR})
          )
else(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
  set(POSTGIS_FOUND FALSE)
  message(STATUS "Postgis not found.")
endif(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
mark_as_advanced(POSTGIS_INCLUDE_DIR
                  POSTGIS_LIBRARIES)
endif(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
Extended pgRouting: extra/visibility_graph

**C language**
- `src`
  - build_visibility_graph.c
  - insert_point_into_graph.c

**sql**
- routing_vg.sql
- routing_vg_util.sql

**plpgsql language**

**librouting_vg.so**
- **Datum**
  - build_visibilitygraph(PG_FUNCTION_ARGS)
- **Datum**
  - insert_point_into_graph(PG_FUNCTION_ARGS)
- **FUNCTION**
  - build_visibility_graph(tablename varchar)
- **FUNCTION**
  - shortest_path_into_visibilitygraph(…)

---

**src**

- `build_visibility_graph.c`
- `insert_point_into_graph.c`

**sql**

- `routing_vg.sql`
- `routing_vg_util.sql`
AGENDA

• VTMS overview

• Decision Support in a VTMS

• Visibility Graph to model Open Sea

• Geospatial DB and GIS functions to solve SP

• Extended pgRouting for Visibility Graph

• Conclusions
Conclusions

UP:
• Optimized DBMS GIS libraries can be exploited to easily represent and manage geographic objects

• The complexity of calculations is totally demanded to the DBMS set of functions

DOWN:
• Extended pgRouting approach turned out to be not effective for near real time application involving lots of multi sensor tracks e.g. In a scenery with more than 2000 tracks, the insertion of one node into a previously built visibility graph, on a PowerPC needs about 4 seconds. Moreover, time to calculate shortest path between two points is about 7,5 seconds

• Extended pgRouting is work in progress again and has been used only internally as object of study
Thank you!

Contacts:
Angela Pappagallo - angela.pappagallo@intecs.it
Domenico Balestrieri - domenico.balestrieri@intecs.it
Massimo Costantini - massimo.costantini@intecs.it
Roberto De Felici - rdefelici@selex-si.com (VTMS System)

Visit our web-sites

http://www.intecs.it
http://www.selex-si.com