# TMA/TDPS Overview for System Developers

Draft 11/30/2011

Contents

[TMA/TDPS Overview for System Developers 1](#_Toc310411079)

[Scope 2](#_Toc310411080)

[Overview 2](#_Toc310411081)

[Minimum Path 2](#_Toc310411082)

[Analyst Path 2](#_Toc310411083)

[Isochronal Contours 2](#_Toc310411084)

[Multi-Source Contours 3](#_Toc310411085)

[Corridor 3](#_Toc310411086)

[Input to Dijkstra’s Algorithm 3](#_Toc310411087)

[ADRG Map Background 4](#_Toc310411088)

[SRTM Elevation Model 5](#_Toc310411089)

[VMAP1 Terrain Class Polygons, Road and River Line Vectors 5](#_Toc310411090)

[Mover Model 9](#_Toc310411091)

[Data Preparation 11](#_Toc310411092)

[Region Elevation Model 11](#_Toc310411093)

[AOI Terrain Raster and Roads/Rivers Vector Network 11](#_Toc310411094)

[Minimum Path 12](#_Toc310411095)

[Analyst Path 13](#_Toc310411096)

[Isochronal Contours 14](#_Toc310411097)

[Multi-Source Contours 15](#_Toc310411098)

[Corridor 16](#_Toc310411099)

[Current Interface 17](#_Toc310411100)

[Common Parameters 17](#_Toc310411101)

[const char \*regionName 17](#_Toc310411102)

[const char \*aoiId 17](#_Toc310411103)

[const char \*moverName 17](#_Toc310411104)

[const char \*dataset 17](#_Toc310411105)

[const char \*rnfName 17](#_Toc310411106)

[SbVec2f &nw, SbVec2f &se 17](#_Toc310411107)

[int lines, int samples 17](#_Toc310411108)

[Minimum Path (minpath.C: tmaMinPath()) 17](#_Toc310411109)

[SbVec2f &start, SbVec2f &end 17](#_Toc310411110)

[Isochronal Contours (contours.C: tmaContours()) 17](#_Toc310411111)

[SbVec2f &start 17](#_Toc310411112)

[Multi-Source Contours (contours.C: tmaMultiSourceContours()) 18](#_Toc310411113)

[vector <SbVec2f> & locations 18](#_Toc310411114)

[Corridor (contours.C: tmaCorridor()) 18](#_Toc310411115)

[SbVec2f &start, SbVec2f &end 18](#_Toc310411116)

# Scope

This document describes the Tactical Movement Analyzer (TMA) analysis functions, while referencing some capabilities of the Terrain Data Preparation System (TDPS).

# Overview

The TMA functions are applications of Dijkstra’s Algorithm, which calculates the minimum distance or cost between a source point and multiple destination points. In TMA, the Dijkstra cost is travel time in minutes for a modeled mover. Following is a short description of the five analysis types. Further below are example analysis results.

## Minimum Path

Answers the question “What is a minimum path in travel time between two specified points, and what is the distance (in km) and travel time (in minutes) on that path?”

## Analyst Path

An iterated minimum path. Answers the question “What is a minimum path, path length, and time, for traveling through a specified ordered sequence of points?”

## Isochronal Contours

Answers the question “Given a starting point and a few evenly spaced time intervals, what are the geographic regions that could be touched by the mover during each interval?

## Multi-Source Contours

Answers the question “Given a set of movers distributed at particular locations at some initial time, and given a few evenly spaced time intervals, what are the geographic regions that could be touched by at least one of the movers during each interval?

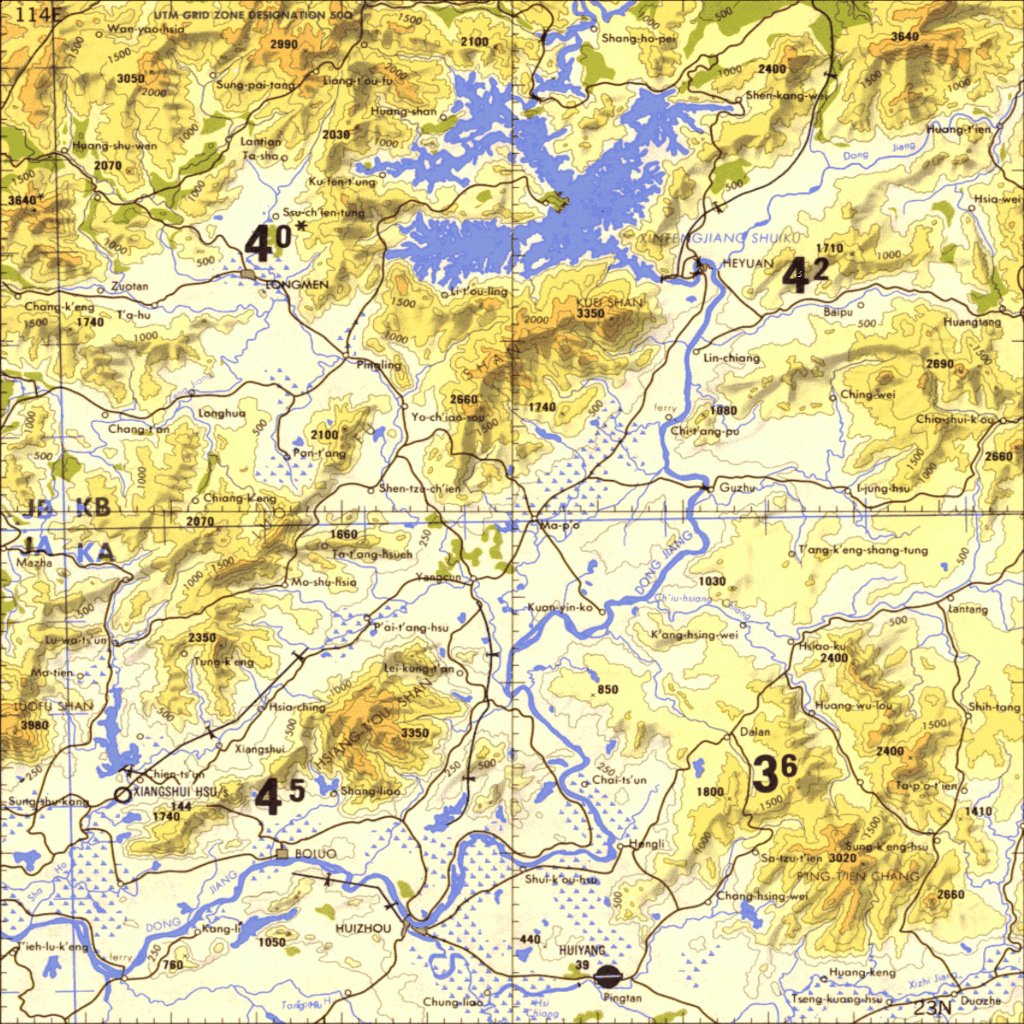
## Corridor

Answers the question “If a vehicle was seen full of cargo at Location\_1 at 1:00, and empty at Location\_2 at 2:00, where might it have dumped the cargo?”

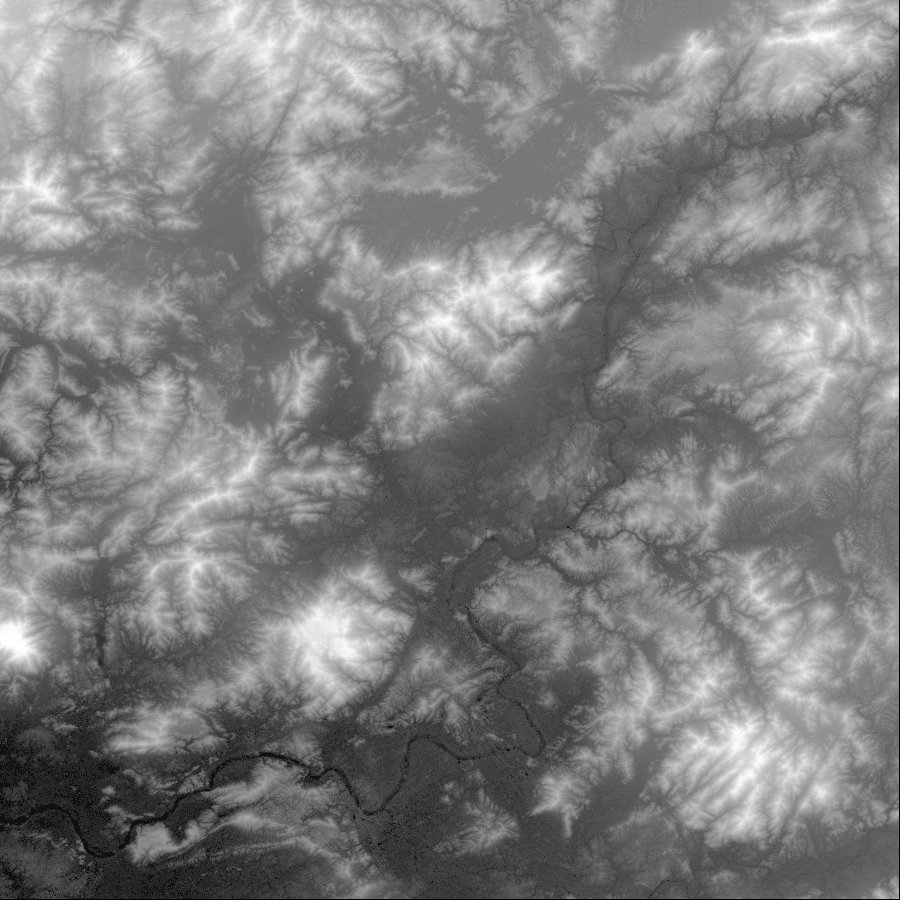
# Input to Dijkstra’s Algorithm

The TMA analysis functions require a model of terrain type polygons, an elevation model, road and drainage vectors, and a mover model describing the speed at which the mover travels through various terrain types, along various road types, across various drainage types, and up and down various elevation gradients, and through various weather conditions. The TMA user interface also requires a map background to allow the user to select points, and to display analysis results. However, if TMA were used as a software service, it would have no use for the map background. In the following examples, a region from 114W to 115W and 23N to 24N was extracted from the various input sources. ADRG was used for the map background. SRTM was used for the elevation model. VMAP Level 1 was used for the terrain polygons, and road and drainage vectors.

## ADRG Map Background



## SRTM Elevation Model



## VMAP1 Terrain Class Polygons, Road and River Line Vectors

The vector display below was generated by the TDPS. The data originated in VMAP1 form, and was transformed based on project requirements based on filters such as the following:

# {From trackl}

Roads1\_Loose\_Surface\_trackl {

"$f\_code" == "AP010"

}

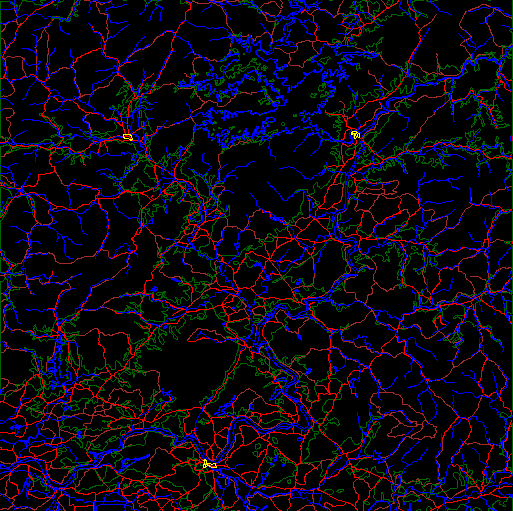
# {From traill}

Roads1\_Loose\_Surface\_traill {

"$f\_code" == "AP050"

}

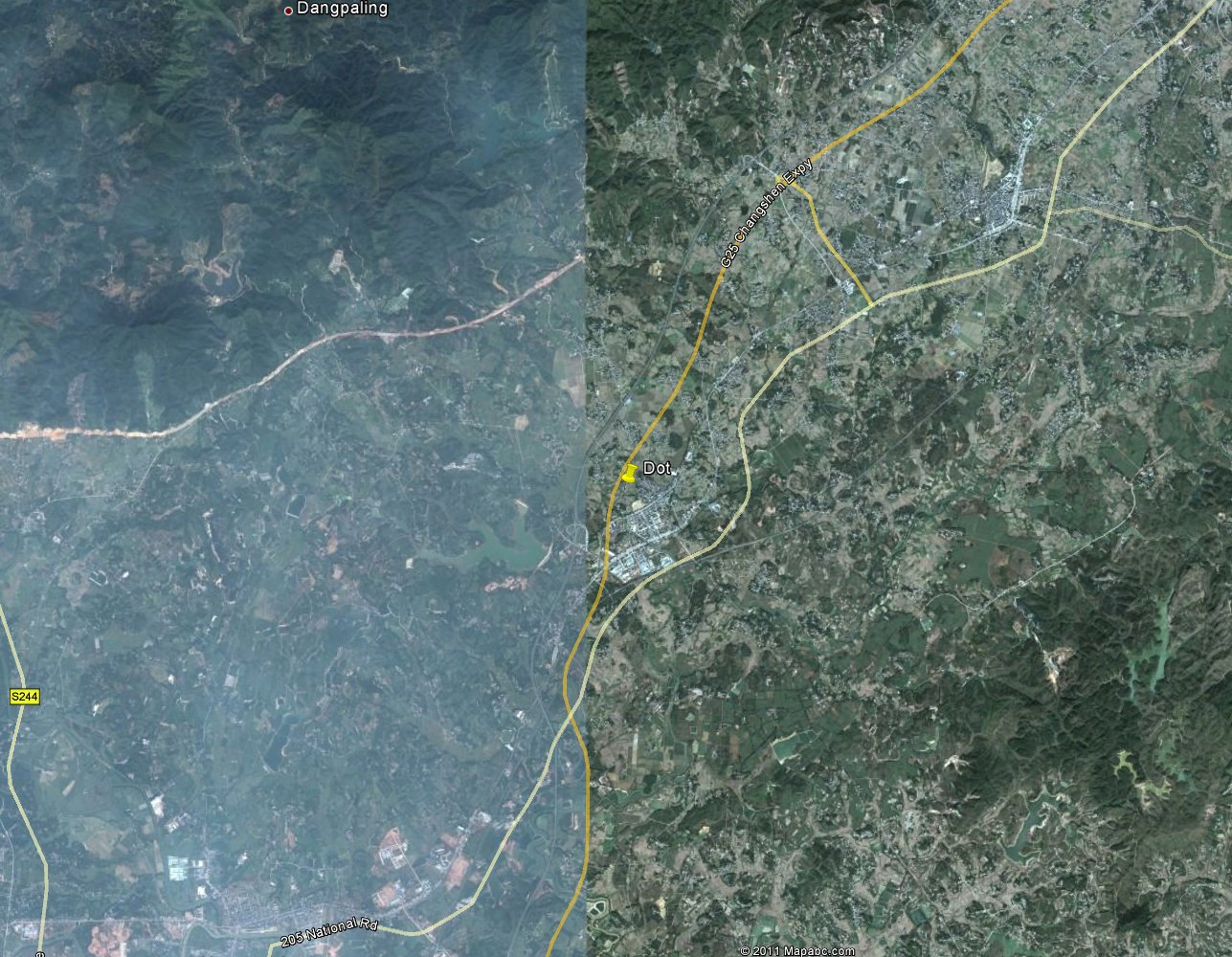
These two TDPS filters map VMAP1 trackl class vectors with feature code AP010 (Cart Track) and trail class vectors with feature code AP050 (Trail) to the TDPS vector type “roads1”, a type having meaning in the TDPS project. The image below displays roads in red, drainage in blue, built-up areas in yellow, and vegetation areas in green. Note that the vector data is rather dated. Only dirt and small paved roads appear in the vector data. Below the vector image is a recent Google Earth view of the same area, showing a number of multi-lane highways, and much more built-up area.



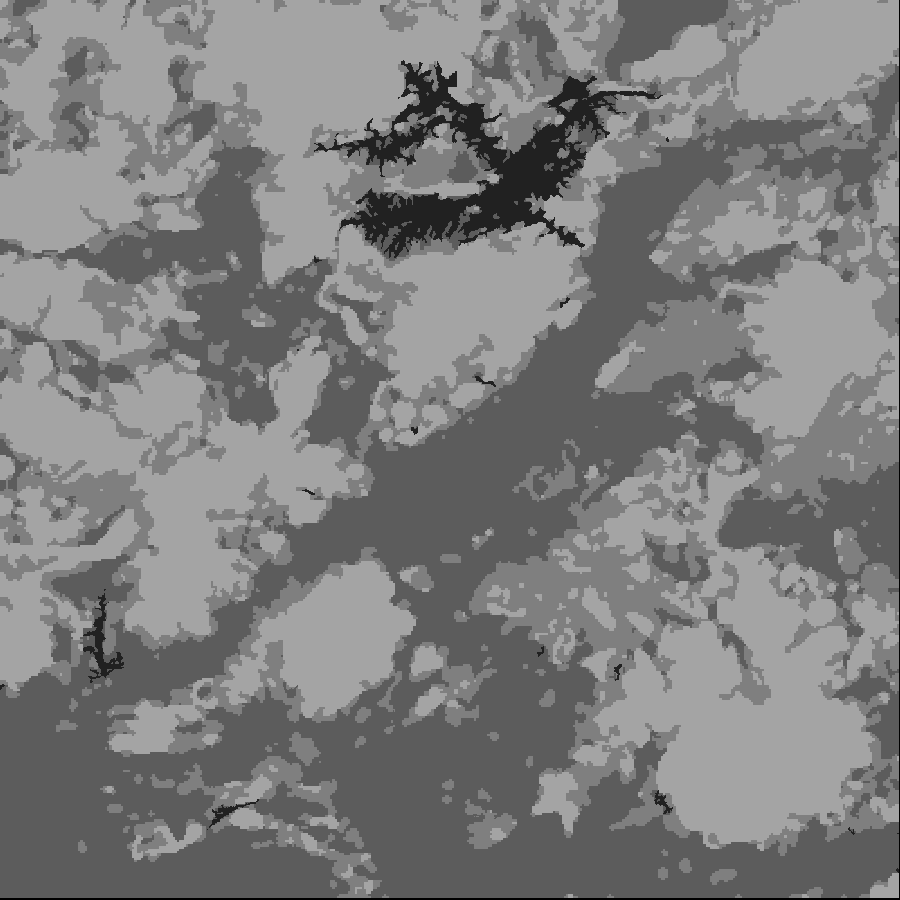
The Google pushpin labeled “Dot” in the center below corresponds to the village named Ma-po in the ADRG image. That label seems to be missing from the Google image. Zooming in on that location

## Google1x1.jpg

yields the next Google image. The built-up area just south of the pushpin may have been known as Ma-po. Note that little remains of the road shown in the ADRG image running south-east of Ma-po. Also note the misregistration between Google’s image and its highway vector data set. A conclusion to draw from the disagreement in existence and location of features in the various sources discussed above, and the fact that a user may have more current data than is available through standard sources, is that the user needs to be able to easily view and understand the data driving analysis results in order to explain (and believe or disregard) TMA results. Another conclusion is that the user needs a way to efficiently edit the data to reflect current ground truth. TMA includes a capability to add weather polygons, having impeding effects as defined for each mover type. It also provides a capability to add impenetrable obstacles, for example to represent a destroyed roadway. The TDPS tool allows arbitrary creation, deletion, and editing of vector data. A user likely would need all of these capabilities.



Below is a rasterized set of VMAP1 terrain polygons, with each terrain class painted with its own shade of gray:



## Mover Model

A TMA mover’s speed is determined by how its type interacts with

* Terrain type
* Elevation slope
* Road type
* River type (when crossing)
* Weather effects

The mover model that was used for the example analyses above, called “Humvee”, includes the following characteristics.

**Terrain Slope**

|  |  |  |
| --- | --- | --- |
| Slope (degrees) | TMA SLOPE CODE | Speed Multiplier |
| -2.5 to 2.5 | FLAT | 1.0 |
| 2.5 to 5 | UP\_1 | 0.9 |
| 5 to 10 | UP\_2 | 0.75 |
| 10 to 20 | UP\_3 | 0.5 |
| 20 to 30 | UP\_4 | 0.15 |
| 30 to 45 | UP\_5 | 0.15 |
| > 45 | UP\_6 | 0 |
| -5 to -2.5 | DOWN\_1 | 1.1 |
| -10 to -5 | DOWN\_2 | 1.3 |
| -20 to -10 | DOWN\_3 | 0.9 |
| -30 to -20 | DOWN\_4 | 0.7 |
| -45 to -30 | DOWN\_5 | 0.5 |
| < -45 | DOWN\_6 | 0 |

**Travel Speed**

|  |  |
| --- | --- |
| Terrain Type | Speed (km/hr) |
| Water | 0 |
| Wetlands | 0 |
| Ice\_Snow | 0 |
| Urban | 40 |
| Forest | 0 |
| Shrub | 12 |
| Agriculture | 12 |
| Grassland | 12 |
| Barren | 12 |
| Loose\_Roads (TDPS Roads1 in case above) | 60 |
| Hard\_Roads (TDPS Roads2 in case above) | 90 |

**Weather**

|  |  |
| --- | --- |
| Feature | Speed Multiplier |
| Dust | 0.8 |
| Fog | 0.8 |
| Lit\_Rain | 0.9 |
| Med\_Rain | 0.8 |
| Hvy\_Rain | 0.7 |
| Lit\_Snow | 0.5 |
| Med\_Snow | 0.3 |
| Hvy\_Snow | 0.1 |

These attributes are easily defined and changed. However, mapping from the source VMAP data set to a condensed set of terrain classes, as used by TDPS, takes careful analysis of the particular VMAP vector ontology, and the context of tool use. A particular user community likely would want to choose their particular VMAP interpretation, which would then be fairly durable.

## Data Preparation

### Region Elevation Model

A world region is selected with integral latitude and longitude boundaries. The elevation image is extracted from a world SRTM elevation model by AFIDS.

### AOI Terrain Raster and Roads/Rivers Vector Network

An analytical area of interest (AOI) is selected in terms of corner latitude and longitude, and lines and samples. The TDPS is used to extract a set of road and river vectors into ARC format, based on the AOI corners. That data is converted to TMA vector format. The terrain polygon and vectors are used to generate a raster terrain class image covering the AOI. The road and river line vectors are used to generate a road and river network vector data set for the AOI.

Road Intersections

Roads, Rivers, Areas

RNF

(binary)

RIF

(binary)

TVF

intersect

rasterize

Road/River Network

rnf

PGM

(binary)

VMAP

Vectors

ARC

Vectors

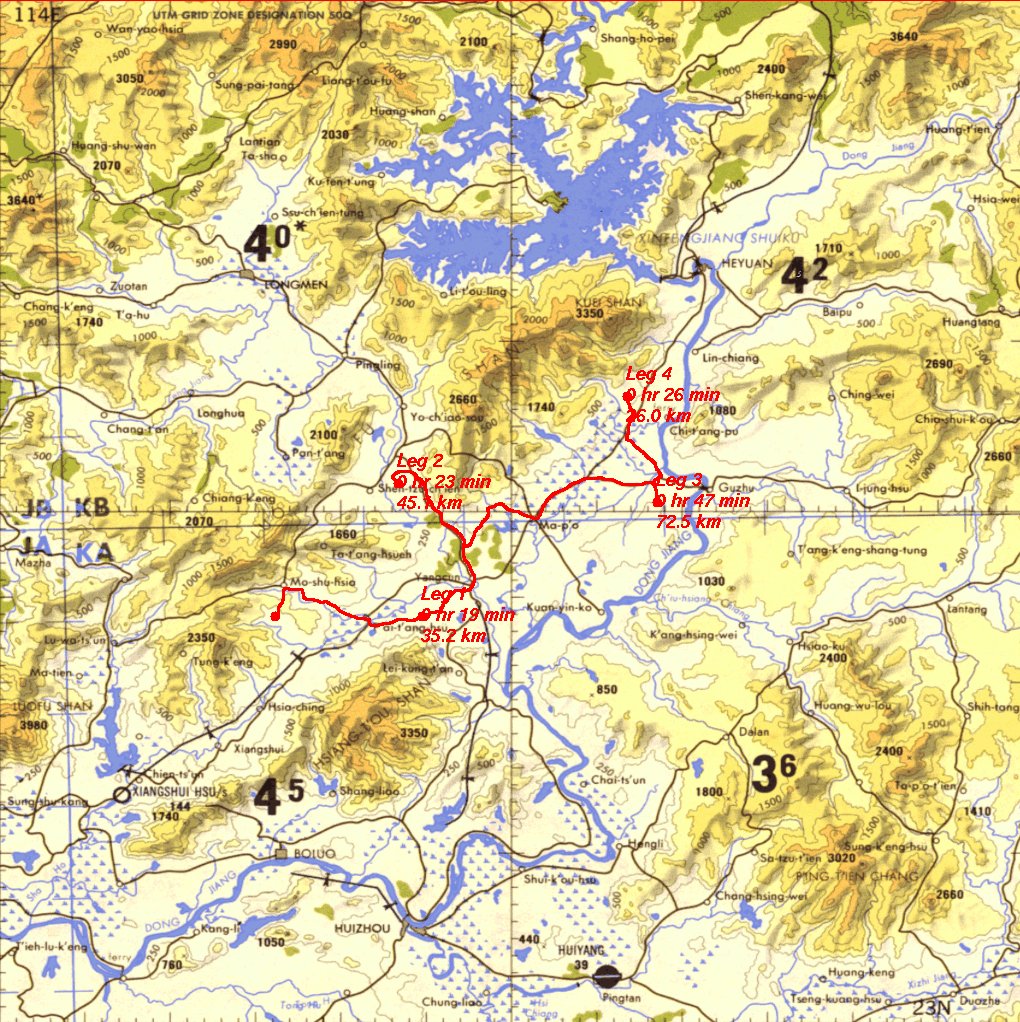
Convert

TDPS

# MinPath.jpg

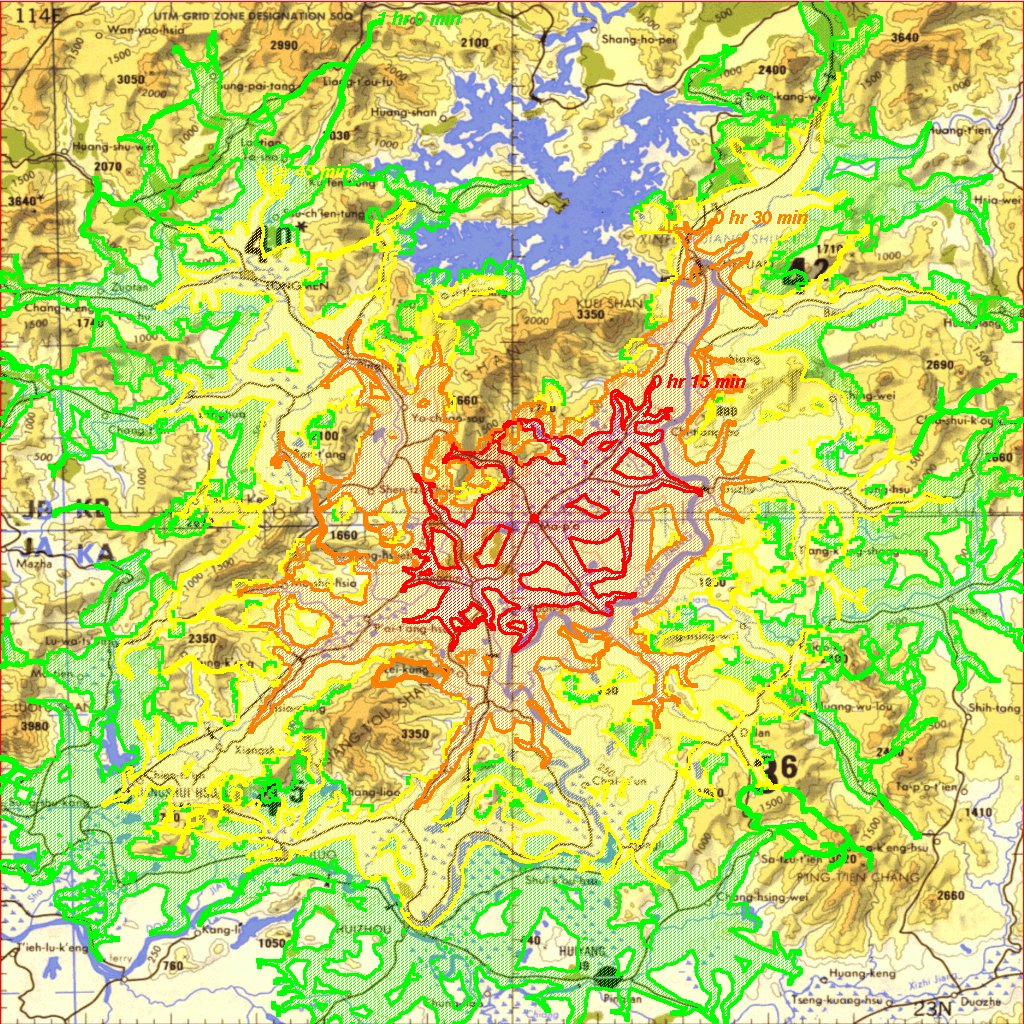
# Minimum Path

Given a source and destination point, the minimum path function calculates a sequence of points that connect the source and destination with a minimum travel time for a chosen mover. Note that there may be many possible minimum paths. TMA calculates only one. Below, moving from left to right, TMA uses a road not drawn on the ADRG map to reach an intersection of roads. Then it travels along a number of roads, eventually leaving the road network, traveling off-road to the destination. Note that the vector data is well-registered to the ADRG.



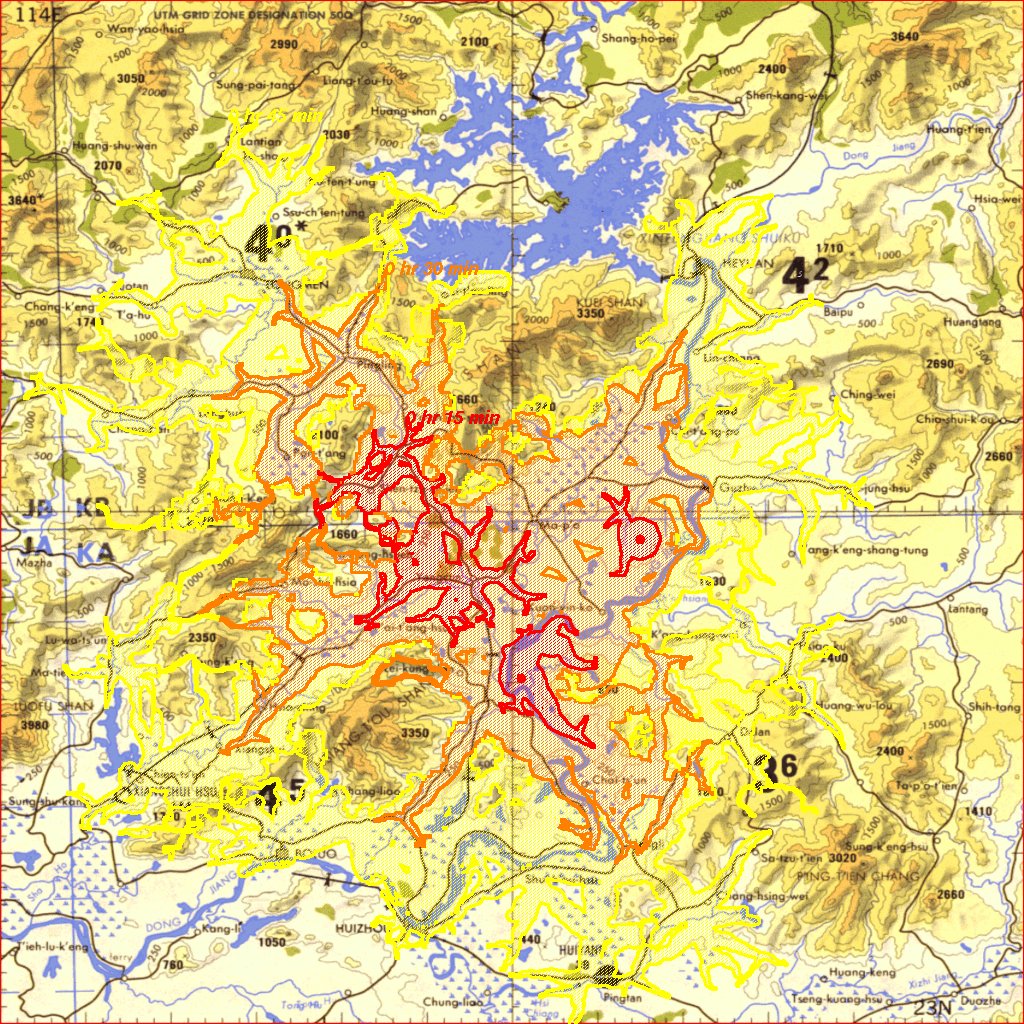
# Analyst Path

The analyst path function executes a series of minimum path calculations, pair wise connecting an ordered set of points provided by the user. The leftmost red dot is the initial point. The mover retraces part of its path along Leg 3, traveling between the third and fourth points. The northern most point is the final destination.



# Isochronal Contours

Given a starting point and a time interval, the isochronal contour analysis calculates a set of nested polygons containing possible locations of a mover at fixed time intervals since beginning movement from the starting point. Below, the red dot in the center of the red shaded region is the starting point. The mover could reach any point within the red region within 15 minutes, any point within the orange region within 30 minutes, any point within the yellow region within 45 minutes, and any point within the green region within one hour. Note that these regions may have holes.



# Multi-Source Contours

The multi-source contours analysis calculates Dijkstra cost surfaces for multiple starting points, calculates a minimum of those cost surfaces, and then calculates isochronal contours from that minimal cost surface. Below are three red dots, which are the starting points. Within 15 minutes, every point within the red regions could be reached by at least one mover. Here, the 15-minute regions are disjoint, so we can see that each red region can be reached only by its local mover during that interval. Within 30 minutes, every point within the orange region can be reached by at least one mover. Likewise, every point within the yellow region within 45 minutes.

# Corridor.jpg

# Corridor

A corridor is calculated containing all the points that the vehicle may have visited during a transit time interval between two geographic points. The corridor analysis calculates Dijkstra cost surfaces for two starting points, then computes a sum of those two surfaces, yielding a corridor boundary around the two points. Below, the red dots at Yangcun and Guzhu are the two corridor end points. Any point within the red region (excluding the internal voids) could be reached by the mover traveling between Yangcun and Guzhu.

# Current Interface

## Common Parameters

### const char \*regionName

Used to look up elevation and terrain class raster image file names from an internal database.

### const char \*aoiId

The name of an analysis AOI. This selects a particular attribute tuple from an internal database, including the geolocation of its corners, and its count of lines and samples. Computationally, this is used to map between geolocation and pixel space.

### const char \*moverName

The name of the mover model. This selects a set of attributes describing movement cost effects such as terrain type and elevation slope.

### const char \*dataset

Selects a particular version of the mover model. Current possible values are “pdc” for the Pacific Disaster Center, and “mil” for military applications.

### const char \*rnfName

The file name of the road network model file of the analysis AOI.

### SbVec2f &nw, SbVec2f &se

The geographic corners of the analysis AOI, can be derived from aoidId.

### int lines, int samples

The lines and samples of the analysis AOI, can be derived from aoiId.

## Minimum Path (minpath.C: tmaMinPath())

Returns nothing. Writes output to ${TMA\_DATA}/minPath.dat

Parameters:

### SbVec2f &start, SbVec2f &end

Locate the path start and end points in latitude and longitude.

## Isochronal Contours (contours.C: tmaContours())

Returns nothing. Writes Dijkstra cost grid to ${TMA\_DATA}/minTimeSoFar.dat (no contours yet). A utility mintime2contours then finds and paints contours into a PGM image.

Parameters:

### SbVec2f &start

Locates the start point in latitude and longitude.

## Multi-Source Contours (contours.C: tmaMultiSourceContours())

Returns nothing. Writes Dijkstra cost grid to ${TMA\_DATA}/minTimeSoFar.dat (no contours yet). A utility mintime2contours then finds and paints contours into a PGM image.

Parameters:

### vector <SbVec2f> & locations

Locates the start points in latitude and longitude.

## Corridor (contours.C: tmaCorridor())

Returns nothing. Writes Dijkstra cost grid to ${TMA\_DATA}/minTimeSoFar.dat (no contours yet). A utility mintime2contours then finds and paints one contour into a PGM image.

Parameters:

### SbVec2f &start, SbVec2f &end

Locate the start and end points in latitude and longitude. Note that there is no algorithmic distinction between these two points.