ERDC/CERL TR-06-28

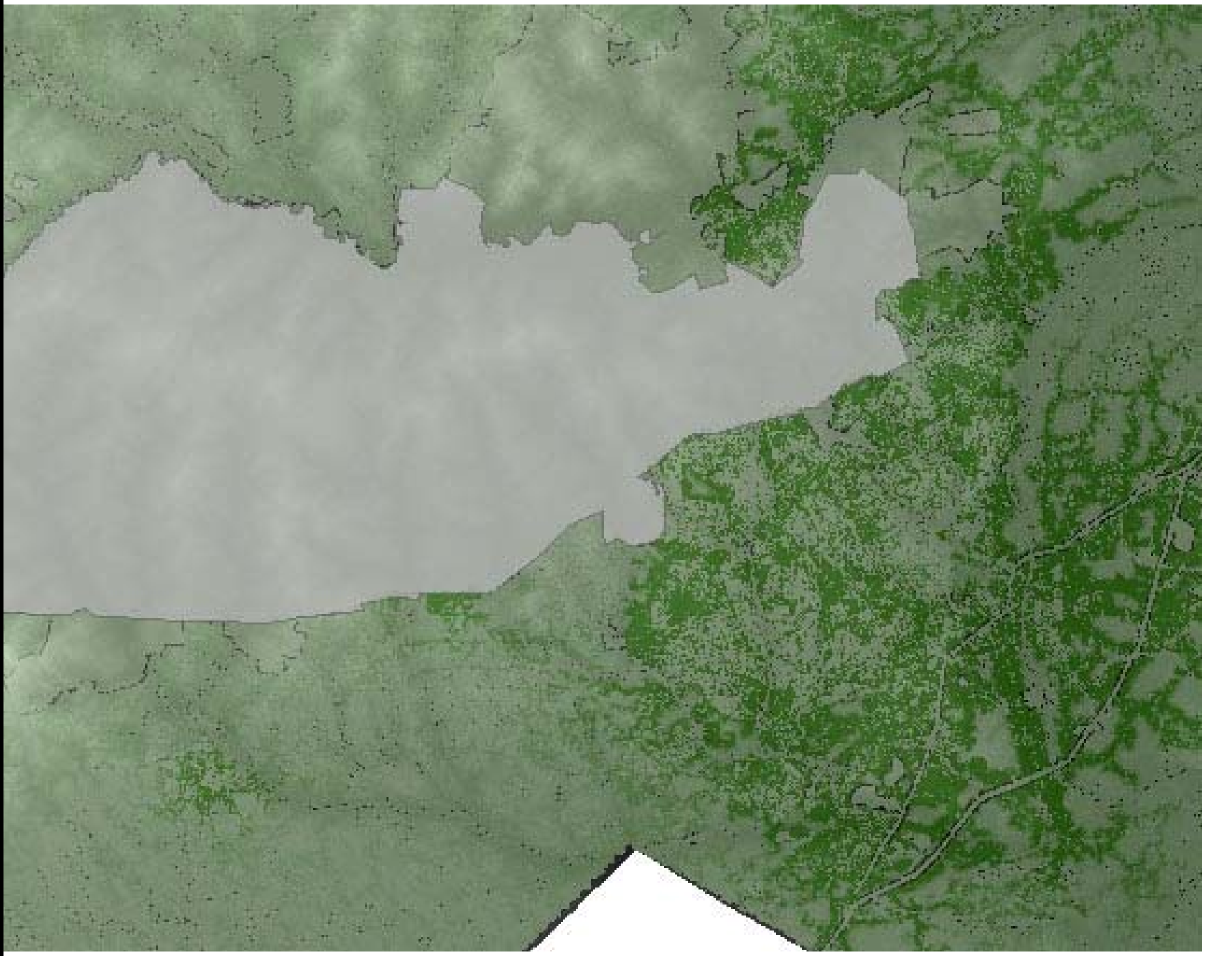
Construction Engineering Research Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

LEAMram™: Land use Evolution and impact Assessment Model Residential Attractiveness Model

James D. Westervelt and Joseph Rank

September 2006



ERDC/CERL TR-06-28
September 2006

LEAMram™: Land use Evolution and impact Assessment Model Residential Attractiveness Model

James D. Westervelt and Joseph Rank

Construction Engineering Research Laboratory (CERL)
U.S. Army Engineer Research and Development Center
2902 Newmark Dr.
Champaign, IL 61824

Final Report

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under Work Unit CS-1257

Abstract: This document describes a GIS approach that can quickly and inexpensively identify areas around military installations that are attractive to residential development – based on proposed regional investments and policies. Analysis results can then be used to inform regional planning and to help identify long-term impacts of residential development on an installation's future opportunity to train and test. Proposed state and local investment in regional planning policies and projects can alter the expected patterns of growth and should be evaluated with respect to their short and long-term impacts on the ability to conduct training and testing at the installation. For demonstration purposes, the approach is demonstrated for a multi-county area around Fort Bragg, NC.

Contents

List of Figures and Table v

Preface vi

Unit Conversion Factors vii

1 Introduction 1

1.1 Background 1
1.2 Objectives 2
1.3 Approach 3
1.4 Scope 3
1.5 Mode of Technology Transfer 4

2 Review of Urban Growth Models 5

3 Approach for Predicting Future Urban Patterns 9

3.1 Overview 9
3.2 GIS Hand Processing 15
3.2.1 smallCity 15
3.2.2 mediumCity 15
3.2.3 largeCity 15
3.2.4 xlCity 15
3.2.5 noGrowth_att 16
3.2.6 dem 16
3.2.7 boundary 16
3.2.8 landcover 16
3.2.9 interstates 17
3.2.10 otherroads 17
3.3 Development of Attractor Maps 17
3.3.1 cities_att 18
3.3.2 slope_att 18
3.3.3 ramp_att 18
3.3.4 staterd_att 19
3.3.5 intersect_att 19
3.3.6 water_att 20
3.3.7 forest_att 20
3.3.8 road_att 20
3.3.9 smallCityTime 21
3.3.10 mediumCityTime 21
3.3.11 largeCityTime 21
3.3.12 xlCityTime 22
3.3.13 travelTime90 22
3.3.14 travelTime30 22
3.3.15 travelSpeed30 22

3.3.16 travelSpeed90 23
3.3.17 roadTravelSpeed30 23
3.3.18 landTravelSpeed30 24
3.3.19 intersection 25
3.3.20 hwyBuff 25
3.3.21 developable 25
3.4 Development of Probabilities Map 26
3.5 Development of Final Maps 27
3.5.1 residential_att 28
3.5.2 residential_prob 28
3.6 Assumptions and Caveats 28

4 The Approach Implementation 31
4.1 Prerequisites 32
4.2 Create Maps 32
4.3 Run LEAMram™ 32

5 A Sample Application 34

6 Summary and Recommendations 42

Bibliography 43

Appendix A: Unix Input File To Create LEAM Landuse Evolution Input Files Using GRASS Commands 44

Appendix B: Acquiring and Processing Required GIS Data 63

Report Documentation Page 69

List of Figures and Table

Figure

1 Past trends are not necessarily good at predicting the future 2
2 Cause-effect chains 3
3 Rural to urban population shift 8
4 The maps on the left are to be turned into a future land use map 10
5 LEAMram data flow 12
6 Graph relating an attractor to probability of residential development 13
7 Range of potential attractor to percent developed graphs 13
8 Computer and human steps 14
9 Sample graph for converting an attractor to a probability of urban area 27
10 Attraction to cities 35
11 Attraction to roads 36
12 Attraction to ramps on limited access highways 36
13 Attraction to major intersections 37
14 Attraction to different slopes 37
15 Attraction to lots bordering forest 38
16 Attraction to lots bordering water 38
17 Summary attraction to residential development 39
18 Summary attraction along southern border of Fort Bragg 40

Table

1 Collection of probability maps 27

Preface

This study was conducted for the Strategic Environmental Research and Development Program (SERDP) Office under Work Unit CS-1257, “The Evolving Urban Community and Military Installations: A Dynamic Spatial Decision Support System for Sustainable Military Communities.” The technical monitor was Dr. Robert W. Holst, Compliance and Conservation Program Manager, SERDP. Mr. Bradley Smith is the Executive Director of SERDP.

The work was performed jointly by the Ecological Processes Branch (CN-N) of the Installations Division (CN), Construction Engineering Research Laboratory (CERL) and the University of Illinois’ Department of Urban and Regional Planning. The CERL Principal Investigator was Dr. James Westervelt. The UI Principal Investigator was Dr. Brian Deal. Stephen E. Hodapp is Chief, CEERD-CN-N, and Dr. John T. Bandy is Chief, CEERD-CN. The Director of ERDC-CERL is Dr. Ilker R. Adiguzel.

CERL is an element of the U.S. Army Engineer Research and Development Center (ERDC), U.S. Army Corps of Engineers. The Commander and Executive Director of ERDC is COL Richard B. Jenkins, and the Director of ERDC is Dr. James R. Houston.

Unit Conversion Factors

Multiply	By	To Obtain
acres	4,046.873	square meters
cubic feet	0.02831685	cubic meters
cubic inches	1.6387064 E-05	cubic meters
cubic yards	0.7645549	cubic meters
degrees (angle)	0.01745329	radians
degrees Fahrenheit	(F-32)/1.8	degrees Celsius
feet	0.3048	meters
gallons (U.S. liquid)	3.785412 E-03	cubic meters
horsepower (550 foot-pounds force per second)	745.6999	watts
inches	0.0254	meters
inch-pounds (force)	0.1129848	newton meters
ounces (mass)	0.02834952	kilograms
ounces (U.S. fluid)	2.957353 E-05	cubic meters
pints (U.S. liquid)	4.73176 E-04	cubic meters
pints (U.S. liquid)	0.473176	liters
quarts (U.S. liquid)	9.463529 E-04	cubic meters
square feet	0.09290304	square meters
square inches	6.4516 E-04	square meters
square miles	2.589998 E+06	square meters
square yards	0.8361274	square meters
tons (long) per cubic yard	1,328.939	kilograms per cubic meter
yards	0.9144	meters

1 Introduction

1.1 Background

The growth on the U.S. population over the past several decades, combined with advances in communication and transportation technologies, has supported more diffused urban development patterns. Installations originally sited in rural areas are often now at the fringe or in the midst of large urbanized or urbanizing areas. Many of the increasing restrictions on training and testing are caused indirectly by patterns of urban development that result in incompatible land uses on either side of installation fence lines. Increased complaints of dust, noise, smoke, and radio and TV interference are directly related to the development of urban patterns near installations. Increases in pressure to protect endangered species and sensitive habitat is directly related to the percent of the habitat in the region on the military installations. Appropriately dark training areas can be lost as city lights follow urban development.

A tool that could predict patterns of urban development would be valuable to help installations better plan for, or entirely avoid such land use conflicts. Various techniques have been developed and applied to predict the potential for the incompatible land uses around military installations, ranging from simple questionnaires answered by installation personnel to more complex, spatially explicit dynamic simulation modeling.

Perhaps the most common approach involves identifying 1-, 3-, and/or 5-mile buffers around installations using current and historic digital land use maps, and then by measuring the urbanized area in these areas using a time-series of maps. This method creates a simple graph of time vs. total amount of urbanized area (Lozar 2003). These simple trends can be very useful and often sufficient to predict growth based on past trends. However, extrapolating future growth based on past trends can be misleading (Figure 1). Growth might continue to accelerate where plenty of land is available for development and where the fringe of a major urbanized area represents an area of economic interest. Growth might moderate if an urbanized center is already in the buffer area, and is growing along the edge of an installation. Growth might decelerate if the there is little or no more land available to develop.

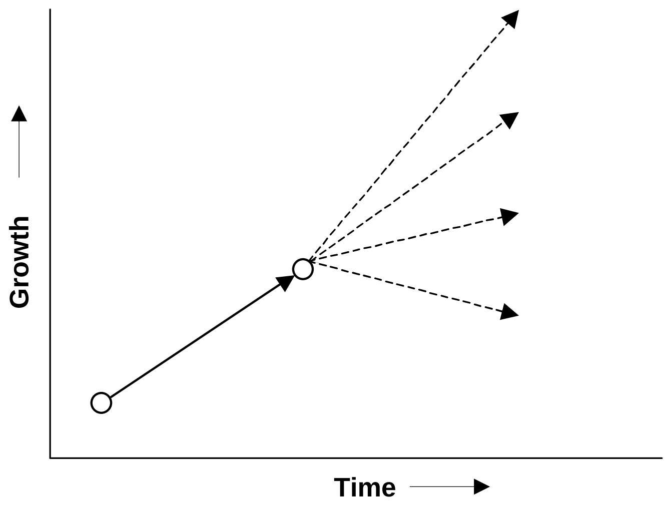


Figure 1. Past trends are not necessarily good at predicting the future.

A more accurate prediction of urban development patterns (and potentials for incompatible land use) would require a careful analysis of the spatial relationships of growing urban centers, available land, natural barriers (e.g., rivers) or man-made barriers (e.g., limited access highways), and zoning. To date, the more involved procedures are very expensive, time consuming, and have substantial data requirements. Consequently, there is a need to describe and demonstrate a quick and inexpensive GIS-based analysis that uses nationally available digital maps to identify the relative attractiveness of land in the vicinity of an installation to residential development in response to proposed regional planning policies and investments. The results would then be useful to explain the long-term direct and indirect consequences of proposed regional investments and policies on local military training and testing opportunities.

1.2 Objectives

The objective of this study was to create an inexpensive GIS-based approach for predicting future urban growth patterns that:

1. Uses nationally available data
2. Requires minimal data preparation
3. Is readily calibrated
4. Does not require time-series data
5. Identifies the relative attractiveness of areas around military installations much better than the current image-processing based trend

analyses and much less expensively than currently available urban growth simulation models.

1.3 Approach

The analysis is first described in detail and then a sample application is explored. All instructions and scripts are included that will allow a GIS specialist familiar with UNIX to conduct the analysis for their areas of interest.

1.4 Scope

Note that this work describes an *approach* for predicting future urban growth patterns. Results generated by the approach are not ready to be used in informing local planning processes.

Many of the encroachment factors affecting training are a direct result of the regional urban pattern. This pattern is the result in many areas of free-market property exchange and ownership. Land attractiveness to the free market participants (e.g., homeowners) is based on local county and city investments and policies including roads, utilities, and zoning. Future training and testing capacities on installations will be directly and indirectly affected by regional policies and investments so that those capacities are also, in part, a function of regional land use investments and policies (Figure 2).

$$\text{Training/Testing Capacity} = f\left(\begin{array}{l}\text{soil, weather, doctrine, terrain, vegetation,}\\ \text{habitat, regional habitat, noise effects,}\\ \text{dust/smoke effects, ambient light, radio}\\ \text{frequency interference [RFI]}\end{array}\right)$$

$$\begin{aligned}\text{regional habitat} &= f\,(\text{urban patterns})\\ \text{noise effects} &= f\,(\text{urban patterns})\\ \text{dust/smoke effects} &= f\,(\text{urban patterns})\\ \text{ambient light} &= f\,(\text{urban patterns})\\ \text{RFI} &= f\,(\text{urban patterns})\\ \text{Urban patterns} &= f\,(\text{free-market land development})\\ \text{Development} &= f\,(\text{regional land use investments and policies})\end{aligned}$$

therefore:

$$\begin{array}{c}\text{Training and}\\ \text{testing capacity}\end{array} = f\left(\begin{array}{c}\text{regional land use}\\ \text{investments and policies}\end{array}\right)$$

Figure 2. Cause-effect chains.

The effort described in this report produced a map showing the relative attractiveness of properties across the landscape to future urbanization. The model created to generate this map can then be used to test for alternative policies and investments including zoning, property purchases, and investments in roads.

1.5 Mode of Technology Transfer

It is anticipated that the information in this report and accompanying digital maps will directly assist military installations, their supporting organizations, and the Office of Economic Adjustment (OEA). This report will be made accessible through the World Wide Web (WWW) at URL:

http://www.cecer.army.mil

2 Review of Urban Growth Models

Different organizations have developed many urban growth simulation models for a variety of purposes ranging from research and education to city and regional planning (USEPA 2000). Each model captures a different combination of the factors involved in growth to answer specific questions. Generally, models can be separated into two distinct spatial scales: city and region. Models such as DRAM/EMPAL and MEPLAN are popular in both the United States and overseas, and focus on identifying growth by income and housing costs. These and other models focus on the city itself and deal with growth over the short term (no more than a couple of decades).

Regional models address questions regarding gross growth patterns over the long term (many decades) and the associated implications with respect to environmental measures. Regional models can help military installations address such questions as:

- Where are people likely to be living in the next 50 years?
- What percent of important habitat in the region will be on local military installations?
- What limitations will regional land use patterns place on military installation activities?

Models that specifically address regional growth patterns over the course of many decades include the California Urban Futures model, version 2 (CUF-2), SLEUTH, Landuse Evolution Assessment Model (LEAM™), Smart Places, and What If?:

- *CUF-2* uses a set of econometric models to project future population, household, and employment. The landscape is gridded into one-hectare developable land units (DLU). The value of each DLU for each competing land use is computed and land use change is based on comparing these relative prices.
- *SLEUTH* derives its name from the types of raster GIS data inputs: slope, land use, urban, exclusion, transportation, and hillshading (Clarke and Gaydos 1998). SLEUTH is a cellular automaton model that

updates the state of each cell in each time step based on the state of each cell and its immediate neighbors.

- *LEAM*™ takes a similar approach (Deal 2003). LEAM uses a 30-meter grid-cell cellular automation approach, in which each cell is assigned a variety of development attractor values and the landscape evolves based on equations that assign future cell states based on the state of cells and their immediate neighbors.
- *Smart Places* is implemented as an extension to ESRI's ArcView GIS, and was designed to be easy to use for those familiar with that system (ESRI 1999).
- *What If?* is a stand-alone product that uses ESRI Shapefiles (Klosterman 2003). Parcels are represented as entities using the ArcView Shapefiles that change through time based on the growth needs of the region, the state of the parcel, and the state of its neighbors.

To be useful for projection, all urban growth models must be calibrated (a process that can be time consuming and difficult). The SLEUTH model has been calibrated by testing thousands of combinations of coefficients using very fast computers. LEAM model was calibrated using a statistical technique (George Gertner, personal communication). These and other models can be run for a location using default calibrations, which can be useful for educational purposes and for generating thoughtful conversation among stakeholders and other participants.

Many land use change models have been developed and are being used to test alternative land use policies with respect to their impact on future land patterns in and around cities and towns (USEPA 2000). The Corps of Engineers is adopting LEAM™ to help evaluate how alternative regional policies and land ownership patterns affect future land development. The Corps' primary interest is to help minimize future land use conflict resulting from the development of new uses in areas that are and will be impacted by military training and testing activities.

Although prediction of urban growth is more art than science, it is extremely important as part of the necessary conversations among stakeholders. In the short term (e.g., up to 10-15 years), urban development will likely proceed along the lines established by stakeholders—especially landowners and those who invest in anticipation of development. Such stakeholders follow strategic purchases of land with aggressive and persistent pressure on and participation in the political processes concerned with infrastructure investments and zoning issues. Urban growth models developed to predict the short-term patterns must capture the goals, ambitions, and expectations of landowners, investors, and the political processes. Modeling at this scale will be extremely data intensive; and the model itself can actually change the process and therefore the future.

In the long term (30, 50, and more years), unanticipated changes in technology can dramatically change the way land use patterns develop. For example, the advent of the automobile allowed unprecedented connections of cities to surrounding agricultural land. Adding tractors and other automated implements to the farm continues to reduce the farm labor requirement. The percent of population in the cities has consequently jumped from 6 percent in 1800 to 40 percent in 1900 and 75 percent in 2000 (Figure 3).

Future technology drivers may continue this trend or begin reversing it, or change the patterns of human settlement in entirely new ways. For example, businesses that rely primarily on information exchange may soon have sufficient Internet bandwidth to support remote employees to a far greater extent. A home office may integrate a wall "window" that can be virtually shared with anyone, anywhere with a similar device. With a simple flip of a switch, people may virtually share office space across the world, as needed, nearly eliminating the need for businesses to support a central office location. When the "window" is not being used for business, the office worker may switch to any of hundreds of live scenes from across the world, the moon, or Mars. With no need to physically "go to work," the rural landscape may see a dramatic infusion of business people.

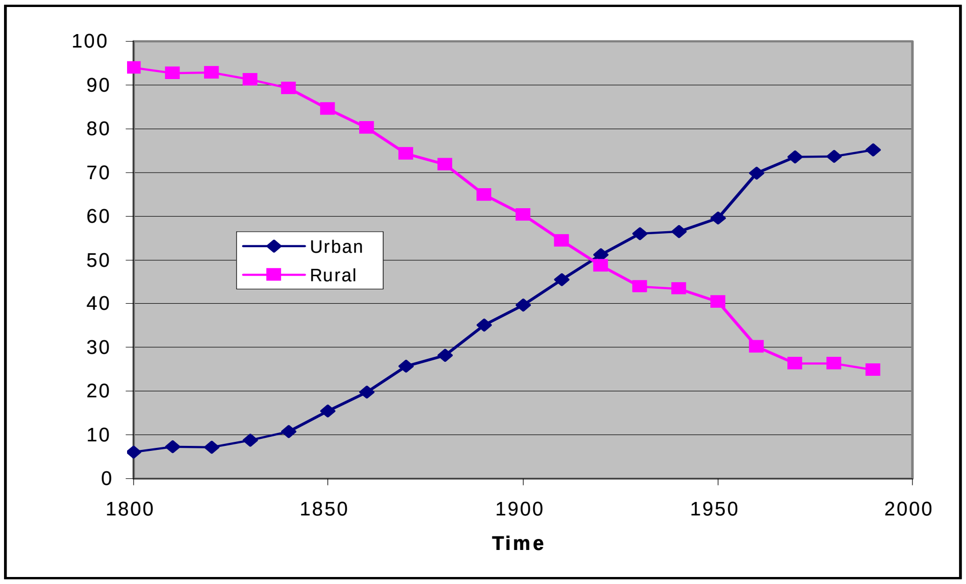


Figure 3. Rural to urban population shift.

Urban growth projection models must be applied very carefully. The potential for a single model to predict urban growth anywhere is very limited. Differences in physical and economic geography make it necessary to calibrate any model for the local circumstances. Differences in culture, demographic patterns, and personal income levels will result in different overall settlement patterns. For example, the poor tend to live more densely and simply, while the rich tend to own and occupy land at much lower densities. Calibration must account for current, historic, and anticipated demographic makeup of populations.

3 Approach for Predicting Future Urban Patterns

3.1 Overview

Urban development in the United States results from interactions among political forces, commercial development, and individual homeowners. Investments through political processes into various infrastructure projects including, roads, highways, interstate highways, and bridges. The same processes develop policies in the form of zoning, tax incentives, and exchange of property rights. Commercial developers create neighborhoods, industrial parks, and shopping areas. Individual homebuyers purchase homes. People make all the associated decisions, which they base on formal and informal analyses of the attractiveness of locations for meeting various goals and objectives. The challenge of this study is to quickly and inexpensively convert readily available GIS maps into projections of where people will live (Figure 4). The following steps identify the attractiveness of undeveloped land to residential development:

1. Acquire land use data for the area of interest.
2. Identify a list of attractors for growth (human habitat suitability).
3. For each attractor:
   a. develop the attractor map
   b. develop a probability of finding residential areas within each attractor category
   c. convert the attractor map to a probability map.
4. Combine all probability maps into an overall attractor map.
5. Develop a probability of finding residential areas within each attractor category.
6. Convert the attractor map to a probability map.

The prediction of future land development is based on the demonstration of land preferences for residential development based on historic decisions reflected in the current land use patterns. The U.S. Geological Survey (USGS) National Land Cover Data (NLCD) program provides a standard set of readily available land use data across the entire United States in the early 1990s and a partial set representing the early 2000s. Data from either time period is required.

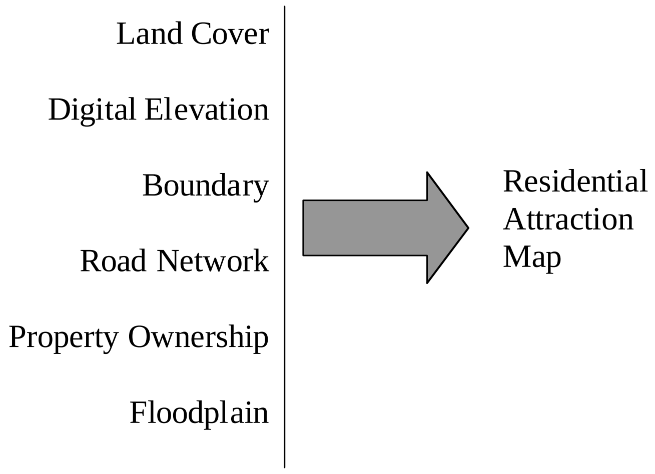


Figure 4. The maps on the left are to be turned into a future land use map.

The next step is to develop a list of residential attractors that are simple to address, sufficient to predict urban patterns, and sufficient across socio-economic classes. (Note that the NLCD data is imagery based and makes no distinction between socio-economic classes.) For the demonstrations reported here, the factors considered are:

- current land use
- slope
- distances to:
  a. water and forest
  b. cities (city centers / employment centers)
  c. county road
  d. state/Federal highway
  e. limited access highway
  f. nearest intersection.

These factors are considered without specific regard for socioeconomic level because, to predict potential change in training/testing capacity at an installation and in its surrounding region, it is most important to predict locations of residences, which can be sensitive to military training activity. Also, while there is ready availability of input maps that indicate location of residential areas, acquiring data indicating the socioeconomic level of neighborhoods can be expensive.

The factors listed above were chosen for their: (1) apparent importance in consideration of constructing residences, and (2) universal availability of supporting information across the United States. Factors other than those in the above list can be considered by acquiring or developing additional input maps using the algorithm described below. Next, the attractor maps must be developed in three general steps:

1. Acquisition of base data
2. Manual processing of GIS data
3. Machine processing of data.

Ideally, acquisition relies on no more than downloading nationally available maps from public (or at least inexpensive) web sites. Manual processing must be minimized to allow quick application to multiple sites. Typically, manual steps that are difficult to automate include assembling the base data, reformatting the data into a common format, and converting the coordinate system and projections to a selected standard.

Finally, processes that can be automated to create the required attractor maps are applied to the results of the manual processing. The attractor maps are factually based and represent information in relevant units, such as miles, meters, slope percent, and minutes. For example, an attraction-to-cities map will be expressed in minutes of travel time; an attraction-to-water map may be shown in meters.

Finally, each attractor map is individually converted to a residential-probability map. This is done by developing a graph relating the percent of land development to the attractor map categories (Figure 5). In the graph, the Data Flows from left to right following the arrows, calculating the probability of residential development from the input maps on the left. The input maps are used in different combinations to create a number of interim maps, which are used together to calculate the residential probability map (Figure 6). This graph is developed by cross-tabulating the number of residential and potential residential locations with attractor ranges. The full range of the attractor map values is divided into sub-ranges (e.g., 10). The percent developed in each range is calculated as the number of residential cells divided by the sum of the residential and potentially residential cells. Residential cells are NLCD categories 21 and 22 and potential residential cells are all other categories that could be developed into residential cells (avoiding categories like water, swamp, highways, etc.)

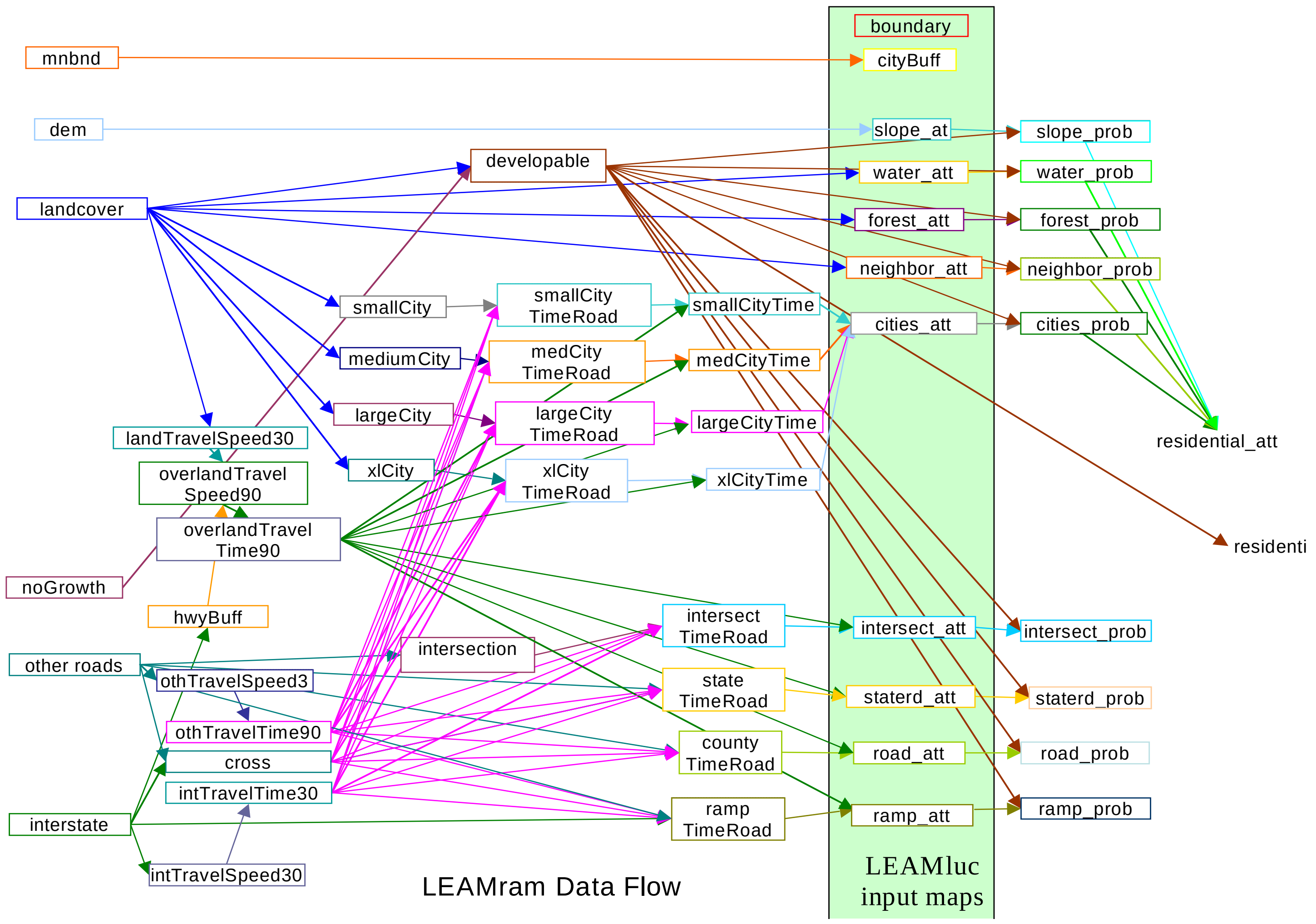


Figure 5. LEAMram data flow.

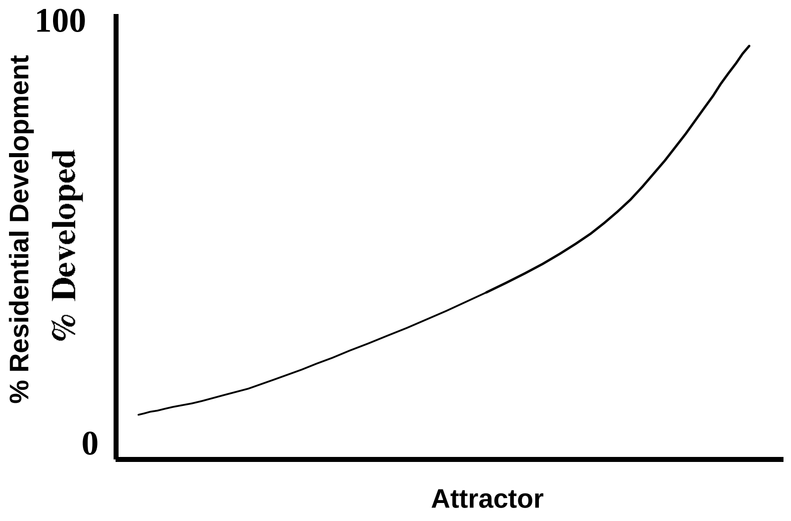


Figure 6. Graph relating an attractor to probability of residential development.

A wide variety of graphs are possible. For example, the top left graph in Figure 7 might represent the probability of development surrounding some negative influence – like an oil refinery. The top right is typical of positive attractors such as city centers. A variant shown in the bottom right shows a different power in the attractiveness decay. The bottom left indicates an attractor that has little influence on the attractiveness. Forest or water in some parts of the country might have a strong positive influence, but in others where water and trees are extremely common, the influence might be represented by the lower-left graph.

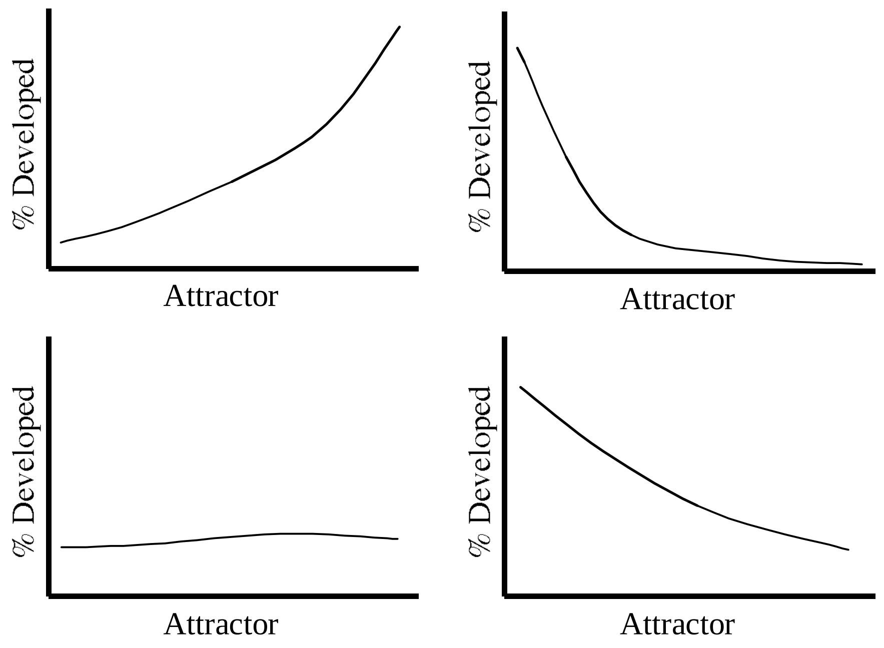


Figure 7. Range of potential attractor to percent developed graphs.

At this point, each attractor is associated with a respective percent-developed graph (values 0-1.0) and each attractor map can be converted, with this graph, to a probability map for that attractor. The probability maps are then integrated into a combined attractor map by adding all the probability maps, then dividing the result by the total number of maps This results in an index map with values between 0 and 1.0. This combined attractor map is then processed in a manner identically to the individual attractor maps: (1) the combined map is divided into a number of ranges; (2) the percentage of land that *could be* residential that actually *is* residential is determined; and (3) a graph is generated from that data; and (4) it is used to convert the combined attractor map into a combined probability map. This is the final product, which indicates the relative attractiveness of land to residential development. Chapter 4 details this process.

Figure 8 details the steps outlined in Figure 4; it shows the steps, processes, and results of creating the maps described in Figure 5. The grey arrows represent three key steps once starting maps have been downloaded from the Internet:

1. Projection and coordinate systems conversion
2. GIS hand processing
3. GIS scripts.

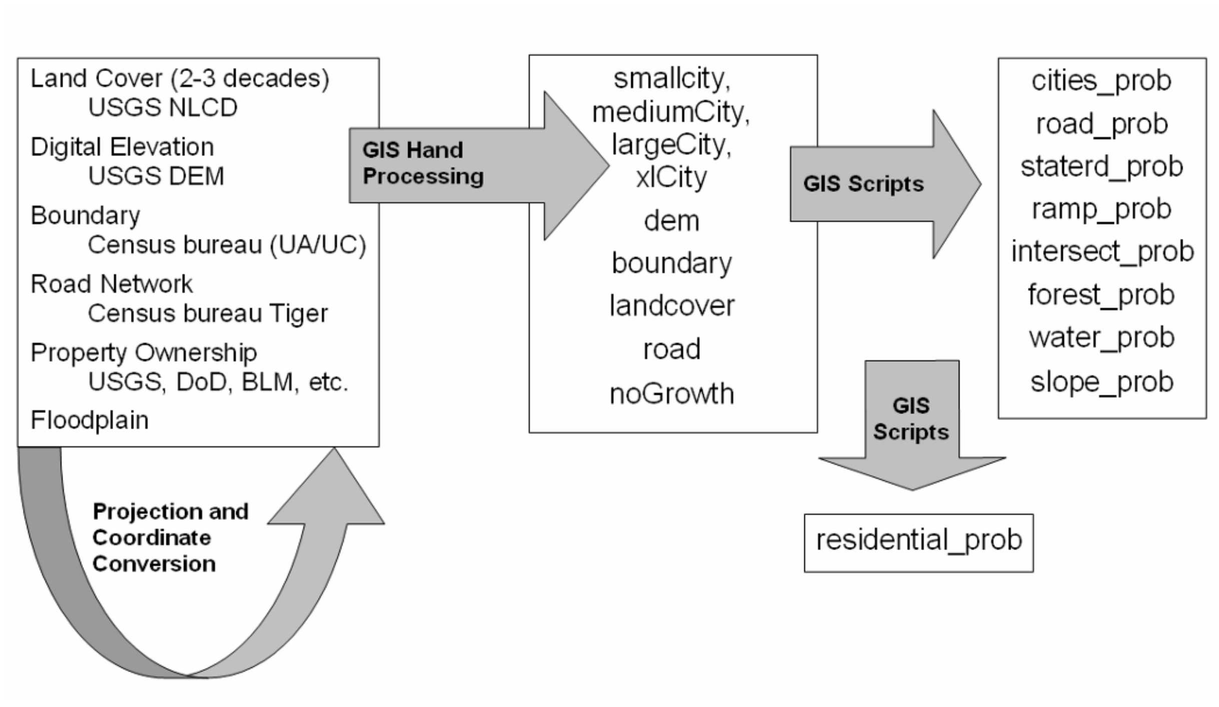


Figure 8. Computer and human steps.

3.2 GIS Hand Processing

The description of each of the “hand processed” maps in the top center of Figure 8 are described below. Each must contain information with specific category information as indicated below. (Information in Appendix B will help GIS technicians develop these maps.)

3.2.1 smallCity

This is a user-developed raster map identifying the central areas of small cities to represent an attractor point to shopping and employment locations. The size of these cities is user defined, but an average population size of the cities must also be determined to assist in the attractiveness processing to these cities. The Census Bureau maps can be used as a guide, but it is the city business centers that need to be developed.

Categories:

0 – no city
1 – city center area.

3.2.2 mediumCity

This is identical to the smallCity map except for medium cities.

Categories:

0 – no city
1 – city center area.

3.2.3 largeCity

This is identical to the mediumCity map except for large cities.

Categories:

0 – no city
1 – city center area.

3.2.4 xlCity

This is identical to the largeCity map except for the largest cities.

Categories:

0 – no city
1 – city center area.

3.2.5 noGrowth_att

This user-developed map identifies where residential development cannot occur due to property owner preferences or to local zoning requirements. No growth areas are derived from property ownership maps including Federal land, and from landcover maps.

Categories:

0 – growth permitted
1 – residential development not allowed.

3.2.6 dem

This is a digital elevation model containing integer values of meters above sea level. Generally, digital elevation models are useful for this application without any special processing.

3.2.7 boundary

This map identifies the extent within which the map processing is accomplished – the study area. The extent is typically defined by a set of contiguous counties.

Categories:

0 – outside the study area
1 – inside the study area.

3.2.8 landcover

This map is based on: "boundary" (above)

This user-provided map can be based on the USGS NLCD series and, in fact, uses the NLCD categories. Advanced users might develop their own maps from processed imagery. The map is clipped using the study-area extent map – boundary.

Categories (NLCD):

0 – No classification
11 – Open water
12 – Perennial Ice/Snow
21 – Low Intensity Residential
22 – High Intensity Residential
23 – Commercial/Industrial/Transportation
31 – Bare Rock/Sand/Clay

32 – Quarries/Strip Mines/Gravel Pits
33 – Transitional
41 – Deciduous Forest
42 – Evergreen Forest
43 – Mixed Forest
51 – Shrubland
61 – Orchards/Vineyards/Other
71 – Grasslands/Herbaceous
81 – Pastures/Hay
82 – Row Crops
83 – Small Grains
84 – Fallow
85 – Urban/Recreational Grasses
91 – Woody Wetlands
92 – Emergent Herbaceous Wetlands.

3.2.9 interstates

This map is based on: “boundary” (page 16)

This is also user supplied, but based on readily available maps provided through the internet. It is, like all of the other maps, raster based.

Category:

Limited access highway.

3.2.10 otherroads

This map is based on: “boundary” (page 16)

This is also user supplied, but based on readily available maps provided through the internet. It is, like all of the other maps, raster based.

Categories:

0 – no road
1 – Limited access highway
2 – Federal highway
3 – State highway
4 – County road
5 –
6 – Ramps to limited access highways.

3.3 Development of Attractor Maps

The maps in this section and those following are generated with a GIS script. This section first lists the attractor maps that hold factual information about the relationship of each cell to surrounding attractors (they end in _att). The remainder of the maps are interim ones, upon which the attractor maps are based.

3.3.1 cities_att

This map is based on:

smallCityTime (page 21)
mediumCityTime (page 21)
largeCityTime (page 21)
xlCityTime (page 21).

This map combines the travel times to the nearest cities from the four categories into an overall attraction to cities map. This map represents the attractiveness of cities based on their employment, shopping, and human contact possibilities.

The steps of the algorithm consist of:

1. The average population of each city category is divided by the respective driving time and the four results are summed.
2. This result is histogram stretched to a scale of 0-255.

3.3.2 slope_att

This map is based on: “dem” (page 16).

The steps of the algorithm consist of:

1. A standard raster-GIS analysis technique is used that calculates the maximum slope from the center of each cell to one of the eight surrounding cells.
2. The results are given in degrees.

3.3.3 ramp_att

This map is based on:

travelTime30 (page 22)
travelTime90 (page 23)
interstates (page 17)
otherroads (page 17).

The steps of the algorithm consist of:

1. Find all roads categorized as ramps in the road map that border limited access highways.

2. Compute the cumulative driving time from these out to a total driving time of 30 minutes at a 30-meter resolution – saving the results.
3. Compute the cumulative driving time from the 30-minute edge in the previous step to the remainder of the study area at a 90-meter resolution.
4. Combine the results at a 30-meter resolution with the first results taking precedence over the second.

3.3.4 staterd_att

This map is based on:

travelTime30 (page 22)
travelTime90 (page 23)
interstates (page 17)
other roads (page 17).

The steps of the algorithm consist of:

1. Find all roads categorized as state highways in the road map.
2. Compute the cumulative driving time from these out to a total driving time of 30 minutes at a 30-meter resolution – saving the results.
3. Compute the cumulative driving time from the 30-minute edge in the previous step to the remainder of the study area at a 90-meter resolution.
4. Combine the results at a 30-meter resolution with the first results taking precedence over the second.

3.3.5 intersect_att

This map is based on:

travelTime30 (page 22)
travelTime90 (page 23)
intersection (page 25).

The steps of the algorithm consist of:

1. Compute the cumulative driving time from intersections out to a total driving time of 30 minutes at a 30-meter resolution – saving the results.
2. Compute the cumulative driving time from the 30-minute edge in the previous step to the remainder of the study area at a 90-meter resolution.

3. Combine the results at a 30-meter resolution with the first results taking precedence over the second.

3.3.6 water_att

This map is based on:
landcover (page 16).

The steps of the algorithm consist of:

1. Find all cells in the landcover map categorized as open water or swamp.
2. Generate 1, 2, 3, and 4 cell buffer zones around these cells.
3. Multiply these to make sure they represent meter-distances from cells.

3.3.7 forest_att

This map is based on:
landcover (page 16).

The steps of the algorithm consist of:

1. Find all cells in the landcover map categorized as forest.
2. Generate 1, 2, 3, and 4 cell buffer zones around these cells.
3. Multiply these to make sure they represent meter-distances from cells.

3.3.8 road_att

This map is based on:
travelTime30 (page 22)
travelTime90 (page 23)
interstates (page 17)
otherroads (page 17).

The steps of the algorithm consist of:

1. Identify roads classified as county roads in the road map.
2. Compute the cumulative driving time from these out to a total driving time of 30 minutes at a 30-meter resolution – saving the results.
3. Compute the cumulative driving time from the 30-minute edge in the previous step to the remainder of the study area at a 90-meter resolution.

4. Combine the results at a 30-meter resolution with the first results taking precedence over the second.

3.3.9 smallCityTime

This map is based on:

smallCity (page 14)
travelTime30 (page 22)
travelTime90 (page 23)

The steps of the algorithm consist of:

1. Compute the cumulative driving time from small cities out to a total driving time of 30 minutes at a 30-meter resolution – saving the results.
2. Compute the cumulative driving time from the 30-minute edge in the previous step to the remainder of the study area at a 90-meter resolution.
3. Combine the results at a 30-meter resolution with the first results taking precedence over the second.

Result:

The result is a driving time map for every cell in the study area to the nearest small city – with greater fidelity within for locations within 30-minute driving times.

3.3.10 mediumCityTime

This map is based on:

mediumCity (page 15)
travelTime30 (page 22)
travelTime90 (page 23)

The steps of the algorithm consist of:

Identical to the smallCityTime algorithm, except with driving times computed to the nearest medium city.

3.3.11 largeCityTime

This map is based on:

largeCity (page 15)
travelTime30 (page 22)
travelTime90 (page 23)

The steps of the algorithm consist of:

Identical to the smallCityTime algorithm, except with driving times computed to the nearest large city.

3.3.12 xlCityTime

This map is based on:

xlCity (page 15)
travelTime30 (page 22)
travelTime90 (page 23)

The steps of the algorithm consist of:

Identical to the smallCityTime algorithm, except with driving times computed to the nearest extra-large city.

3.3.13 travelTime90

This map is based on: "travelSpeed90" (page 23)

The steps of the algorithm consist of:

Multiply the travel speed to get the equivalent travel time for traversing 30 meters.

3.3.14 travelTime30

This map is based on: "travelSpeed30" (page 22)

The steps of the algorithm consist of:

Multiply the travel speed to get the equivalent travel time for traversing 30 meters.

3.3.15 travelSpeed30

This map is based on:

noGrowth_att (page 16)
boundary (page 16)
hwyBuff (page 25)
roadTravelSpeed30 (page 23)
landTravelSpeed30 (page 24).

The steps of the algorithm consist of:

1. Combine the input maps using the following logic:
   if the cell is outside the study area, set it to null
   otherwise, if the cell travel speed is null, set it to null

otherwise if the road travel speed is greater than 0 set it to the road travel speed,
otherwise, if growth is not allowed in the cell, set it to null,
otherwise, if the cell is within the limited access highway set it to null,
otherwise, set it to the land travel speed.

2. The result is a map that provides the travel speed across every grid cell for which travel is permitted or possible.

3.3.16 travelSpeed90

This map is based on:

hwyBuff (page 25)
roadTravelSpeed30 (page 23)
landTravelSpeed30 (page 24)
travelableLandSpeed (page 24)l

The steps of the algorithm consist of:

1. Although this results in values useful for later analyses at 90-meter resolutions, it is calculated using 30-meter resolution cells.
2. Find the maximum road travel speed in the surrounding 9-cell neighborhood.
3. Find the maximum land travel speed in the surrounding 9-cell neighborhood.
4. Find out if there is a limited-access-highway buffer in the 9-cell neighborhood.
5. If the maximum road speed is non-zero, choose that speed
   otherwise, if there is a highway buffer, set to null
   otherwise, choose the maximum overland speed.

3.3.17 roadTravelSpeed30

This map is based on:

interstates (page 17)
otherroads (page 17).

This is simply an assignment of travel speeds (in miles per hour [mph]). Note that the travel speed on roads is set and assumes that road speed does not change with volume. Over the long term, this might be a reasonable assumption if cities respond to congestion by widening roads. Travel speed assignments are made as follows:

Cat	Description	Speed (mph)
0	No road	0
1	Limited access highway	70
2	Federal highway	50
3	State highway	40
4	County road	30
5		20
6	Ramps to limited access highways	30
7		10

3.3.18 landTravelSpeed30

This map is based on:

landcover (page 16)
noGrowth_att (page 16).

This also is a simple assignment of overland travel speeds. This accommodates the notion that driveways to homes can be developed off-road. No-growth areas are assigned the null value to indicate the inability to travel there. Otherwise travel speed assignments are made based on the land-cover categories:

Cat	Description	Speed (mph)
0	No classification	0
11	Open water	0
12	Perennial Ice/Snow	0
21	Low Intensity Residential	0.5
22	High Intensity Residential	0.5
23	Commercial/Industrial/Transportation	0.5
31	Bare Rock/Sand/Clay	1
32	Quarries/Strip Mines/Gravel Pits	1
33	Transitional	1
41	Deciduous Forest	1
42	Evergreen Forest	1
43	Mixed Forest	1
51	Shrubland	1
61	Orchards/Vineyards/Other	1
71	Grasslands/Herbaceous	1
81	Pastures/Hay	1
82	Row Crops	1
83	Small Grains	1
84	Fallow	1
85	Urban/Recreational Grasses	0
91	Woody Wetlands	0
92	Emergent Herbaceous Wetlands	0

3.3.19 intersection

This map is based on:

interstates (page 17)
otherroads (page 17).

The steps of the algorithm consist of:

1. Find all roads categorized as county or state.
2. Thin the results to make sure that no road is wider than a single pixel.
3. Find all road pixels in the results that border more than two other road pixels (i.e., intersections), and save the results.

3.3.20 hwyBuff

This map is based on:

interstates (page 17)
otherroads (page 17).

The purpose of this analysis is to ensure no traffic movement off limited-access highways except at intersections with other roads. (Note that this analysis is associated with an unresolved challenge that involves restricting access to these highways – except at ramps. That is, traffic should be allowed to flow under/over limited access highways without connecting to them.

The steps of the algorithm consist of:

1. All road cells are given category 0 (no buffer).
2. All cells that border road category 1 (limited access highway) cells are given category 1 (buffer).

3.3.21 developable

This map is based on:

landcover (page 16)
noGrowth_att (page 16).

This is simply a convenient map that is used by a number of other analyses, which indicates whether or not cells can be developed for residential. Cells marked as no-growth or containing landcover categories 11, 23, 91, 92, or 85 are given category 0 (no-development). Cells containing landcover categories 21, 22, and 23 are given category 2 (developed), and the remaining cells are developable (category 1).

3.4 Development of Probabilities Map

The above attractor maps mostly contain factual information that must now be converted to development probability maps. The algorithm for each is identical and provides a probability of finding residential areas within each category in each attractor map. The result is a probability graph relating attractor map category values to probability of finding residential areas. This graph is then applied to the attractor map to create a probability map.

The implemented algorithm works as follows:

1. The attractor map is normalized.
    a. The highest category in the map is determined.
    b. The map is divided by this highest value and multiplied by 10. The result is an integer map with values of 0 to 10 that represent ranges of the original categories.
2. This result is cross-tabulated with the "developed" map, which results in a coincident tabulation of the categories in each map.
3. For each of the 10 categories, the number of developed areas are divided by the sum of the number of developed and developable areas to give the probability of finding developed areas in each category, which produces a graph. (Figure 9 shows an example.)
4. The resulting graph is applied to the attractor map to create the probability map. The straight line overlaid on the histogram in the figure represents the graph applied.

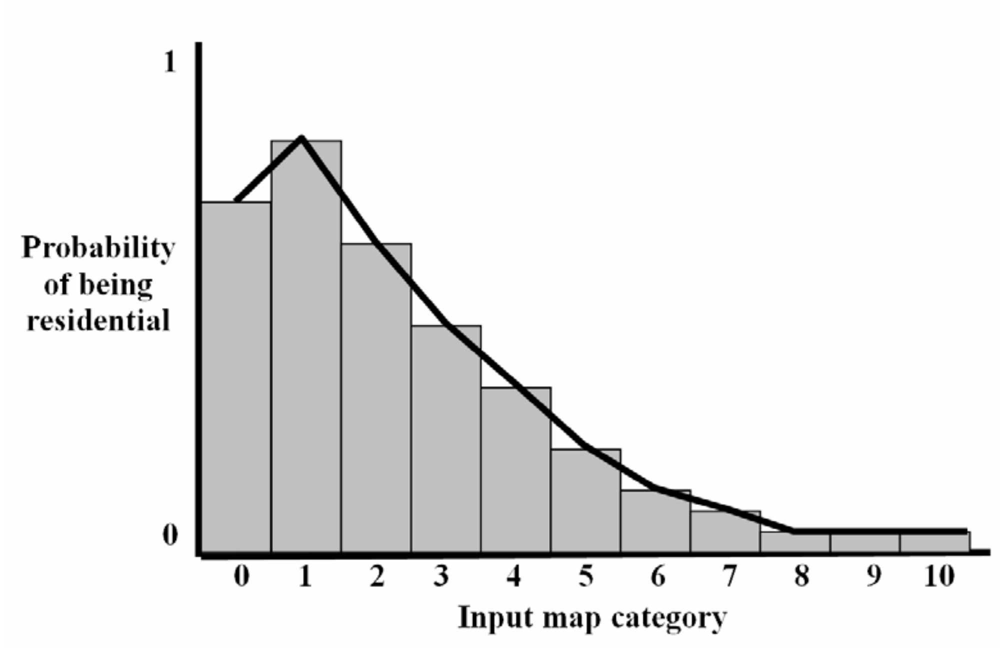


Figure 9. Sample graph for converting an attractor to a probability of urban area.

The end result is a factual map showing the probability of finding urban areas within the categories of the attractor map. The collection of probability maps (Table 1) can then be combined to generate an overall probability map.

Table 1. Collection of probability maps.

Probability Map Name	Based on
cities_prob	cities_att (page 17), developable (page 25)
road_prob	road_att (page 20), developable (page25)
ramp_prob	ramp_att (page 18), developable (page 25)
intersect_prob	intersect_att (page19), developable (page 25)
tree_prob	forest_att (page20), developable (page 25)
water_prob	water_att (page20), developable (page 25)
slope_prob	slope_att (page 18), developable (page 25)
staterd_prob	staterd_att (page 19), developable (page 25)

3.5 Development of Final Maps

Three useful maps can now be generated in sequence. The first is a residential attractor map, which is a combination of the probability maps. The second is a probability map based on this attractor map and is developed with the same algorithm described above. Finally, using this map, a future pattern of land development can be generated.

3.5.1 residential_att

This map is based on:

developable (page 25)	road_prob (page27)
slope_prob (page 27)	ramp_prob (page 27)
staterd_att (page 27)	tree_prob (page 27)
water_prob (page 27)	cities_prob (page 27)
intersect_prob (page 27)	noGrowth_att (page 16)
landcover (page 16)	boundary (page 16)

This provides the first combination of all of the selected attractors to residential development. The values are simply summed and divided by the total number of values to give a result in the range of 0-1. While it is possible to weight these, the importance of each is fully captured in the process generating each. The total number of residential areas in a study area is represented by the area under the curve in Figure 9. Regardless of the shape of the curve (see samples in Figure 7, p 13), the area under the curve will be the same. Attractors with little attraction will be associated with relatively flat graphs and will therefore have little consequence on the results of combining the various probabilities associated with all attractors. Therefore, different weights are not used, nor are any justified with this simple approach.

3.5.2 residential_prob

This map is based on:

residential_att (page27)
developable (page 25).

The above residential attractor map (residential_att) can now be processed with the identical algorithm associated with Figure 9, above. This is a final product that provides the best available picture of the attractiveness to urban growth across the study area based on the statistical evaluation of the attractiveness of chosen factors.

3.6 Assumptions and Caveats

The algorithm described has a number of important assumptions and caveats. It was developed to address a specific question within time and budget constraints. Each important assumption is discussed below.

Only Certain User Questions Are Answered

LEAMram™ was developed to identify the attractiveness of areas within counties to future urban residential development. It is designed to help regional planners understand regional growth potentials. Other approaches and models must be employed to predict details of urban growth within cities or to project future urban growth.

The Current Urban Pattern Is Adequate for Predicting Future Urban Residential Attractiveness

Towns and cities develop over very long periods of time in response to many technical, economic, social, and geographic drivers. Resulting patterns tend to persist, but new development can be different due to technological developments – especially those in transportation and communication. LEAMram™ analyzes the current land use patterns – assuming that they represent the current and future settlement preferences. Models such as LEAM™ analyze land use change or registered housing starts as a better representation of current preferences. Challenges remain to predict the future residential development preferences in response to future transportation costs and communication potentials offered through significantly enhanced internet communication opportunities.

The Input Map Files Are Accurate

Like all computer software analyses, the results are based on a presumption of the accuracy of the inputs. This process relies on nationally available data sets. In particular, the National Land Cover Data set provides land cover classifications at a resolution of 30 meters across the United States. This centrally developed data set is primarily based on image processing and may, for some applications, have an inadequate level of accuracy. Of course, local, better data may be used.

Road Systems Cannot Be Overloaded

A most important part of the LEAMram™ analysis is a set of predictions of the travel time from every 30-meter square parcel to important attractors (intersections, cities, highways, water, etc.). Roads are associated with fixed travel times, thereby making the assumption that road capacity will not be reached, or if reached will be increased through a construction response.

There Are No Differences among Owners and Parcels

No characteristics of owners are considered including wealth, age, philosophies, primary business, etc. In reality, motivations and philosophies of owners can vary significantly resulting in different land use desires. Similarly, differences in parcels, such as size, are not considered. Development may be more likely on large parcels that can be purchased through negotiations with a single owner rather than on an area that involves many parcels and owners. Residential development is more cost effective for larger developments.

The Value of Land for Other Purposes Is Fixed

This is an important assumption that will not be true. The value of land is the amount that the highest bidder is willing to pay. Land may convert to residential when the value of the land for residential (and business/industrial) rises above its value in forest or agriculture. However, it may convert if the latter value drops. For example, a region that traditionally grows a crop that drops in value may see more development than a region with solid or rising crop prices.

Development Attractiveness Is Inherent in Existing Urban Patterns

That is, where people want to live in the future is reflected in the settlement patterns of the past. Perhaps a more accurate, but more expensive approach is to not make this assumption. Instead of calibrating the model on the complete development patterns, calibrate on only the recent development derived from construction start data or on differences between consecutive land use maps – separated by a decade or more. Only the study area is taken into account. There are attractors outside the study area that can affect the study area. However, despite these assumptions and caveats, LEAMram™ results can be very useful in a regional context to identify the relative attractiveness of development in a rapidly developing area.

4 The Approach Implementation

The algorithms described in Chapter 2 have been implemented using the GRASS GIS, ver. 5.3. GRASS was originally developed though a collaborative effort by various U.S. government agencies, universities, and private companies (Westervelt, et al., 1987). Active GRASS development currently involves universities, agencies, and corporations across the world (Netler and Mitasova, 2004). GRASS remains a freely distributable software package that runs under UNIX and Linux systems. GRASS is particularly suitable for the cost-effective support of the LEAMram™ project because of the wealth of raster GIS analysis routines, its freely distributable nature, and the ability to write any necessary new software by adapting existing software source code.

The primary implementation of the algorithms is through a UNIX "makefile." The "make" program reads file dependency information from a file, the makefile, and executes instructions necessary to create missing or out of data files by running specific UNIX instructions on the dependency files. For example, consider the following makefile:

```
target: file-a file-b
cat file-a file-b > target
file-a:
echo "hello " > file-a
file-b:
echo "world " > file-b
```

Invoking the make program on this makefile causes the program to identify the existence and date of creation of target, file-a, and file-b. If file-a and file-b exist, and the creation date on "target" is later than file-a and file-b, nothing happens. If the creation date is later, or the "target" does not exist, the following command is executed: "cat file-a file-b > target," creating "target." If file-a or file-b do not exist, the instructions following their entries, respectively, are executed. These have no dependencies; only their existences are checked. In summary, invoking the make program on this file will result in three files being created, in this case with text.

The algorithms described in the previous chapter were captured in a makefile reproduced in Appendix A. The general format of this file is based on the simple example above. Comments are preceded on lines in the makefile with a pound (#) sign. You are encouraged to read through this file, cross-referencing with the previous algorithm descriptions. Note that the GRASS GIS programs are command-line UNIX programs that mix powerfully with standard UNIX commands such as *cat*, *grep*, *sed*, *awk*, and others.

4.1 Prerequisites

The following prerequisites must be addressed to run the LEAMram™ analysis. First, LEAMram™ is GRASS based and GRASS is primarily available for UNIX and Linux systems. Versions of GRASS are available through the Internet free of charge. GRASS 5.3 or better is required for this application. Acquire and install GRASS on a Linux computer of your choice and become familiar with Linux and GRASS commands. Also, the make program must be available on your Linux system. It is often part of Linux software development packages, which may not have been initially installed.

4.2 Create Maps

Acquire the basic nationally available maps from the Internet. At this point, any raster-based GIS of your choice can be employed to import the maps into a common coordinate system and projection. First, create the LEAMram™ base input maps, adhering to the map names and contents. Then, bring the maps into a GRASS database. Geographic locations in GRASS are stored as sets of maps called mapsets. Locations are directories (folders) in the Linux operating system and mapsets are folders within the location. If needed, create a new GRASS location, which will create the "PERMANENT" mapset. Ensure that the projection and coordinate system matches that chosen when the maps were input and processed. Once a location and mapset are created, run GRASS, selecting them. Import your base maps into GRASS using any of a variety of GRASS map import commands, depending on the format of your maps. You may need to output your maps from the GIS used to initially process the maps into a format "understood" by one of the GRASS import commands.

4.3 Run LEAMram™

Create a makefile with the contents of Appendix A. (We recommend that you contact* the author of this report for a latest version of this makefile as the file was under development at the writing of this document.) Call the

* Contact information is available by searching on the title of this report through URL: http://www.cecer.army.mil/techreports

file “makefile” and place it into the “cell” directory of the current mapset. You can identify the directory location of the cell file by looking at the GRASS environment variables returned by the *g.gisenv* program. Change directories to the database, then the location, and finally the mapset. Then change to the “cell” directory at that level. Place the makefile in this directory.

Edit the makefile and locate the names of the base maps under the comment “Provide names of the input maps.” Using GRASS display commands, review these maps and check to make sure they have contents that match their descriptions at the top of the makefile. Once satisfied, run the command “make” while in the “cell” directory. You will get a response similar to the following:

```
make LEmaps - to create LUC input maps
make indexmaps - to create LUCs index maps
make residential_att - to create LUCs residential attractor map
make residential_prob - to create LUCs residential probability map
make futureLanduse - to create a future landuse map
make clean - to remove temporary maps
make veryclean - to remove all maps generated by this makefile
```

These are some of the major targets. If all goes well, you will be able to successfully run the command: “make residential_prob.” The entire process can take many hours depending on the speed of your computer, its amount of main memory, and the size of your study area.

5 A Sample Application

Fort Bragg, NC and its surrounding communities and counties are economically and environmentally linked. Anticipated changes in mission, weapon systems, and training will require substantially larger training areas at a time when anticipated population growth will result in increased demands for urban land. These demands require closer analysis of the landscape, predictions of future needs, identification of potential conflicts, and analysis of the direct and indirect consequences of proposed local investments and policies. The effort described here is an approach for identifying areas associated with a significant probability of urban growth in the next decades.

Fort Bragg, like many military installations with training and testing missions, was placed in a relatively remote area where few or no human settlements nearby would be adversely affected by the military activities, and where the large contiguous tracts of land necessary to support assorted large-scale activities were readily and inexpensively accessible. Purchased property boundaries were demarcated with fences and signs. Military training and testing could then take place "within the fence line." However, some impacts of those activities extended beyond the fence line including noise, dust, smoke, and radio transmissions. Also, in the intervening years human settlements and agriculture have impacted important habitats and populations of threatened or endangered species – leaving the installation with an increasing percentage of the overall habitat and populations. This has resulted in growing concerns about community and public interest in the potential reduction of on-base training and testing.

The procedures described in the previous chapter were applied to the Fort Bragg area, which included all local counties participating in the Sustainable Sandhills effort. The following images show a study area consisting of nearly 10 million 30-meter square grid cells (parcels) across the multi-county area. Fort Bragg is centered in the middle of the images. The projection is Albers Equal Area resulting in the north direction being slightly to the left of vertical. Each image is associated with a color table spanning the scale from 0 to 1. Note that some images are richer in color indicating a tighter correlation between urban areas and the particular driver.

The production of the “attraction to cities” map (Figure 10) is significantly different from the creation of the other maps (Figures 11 through 16). The goal is to produce a map that captures the attractiveness to jobs, shopping, and friends as a function of the innate attractiveness of those qualities mediated by distance from them. Production of this map in an automated way that can be readily applied to any location is challenging. The first challenge is to identify the areas or points that represent centers of attraction, which is accomplished during the manual preparation process. Ideally a separate analysis would be done for each small town and for each city’s neighborhoods and employment centers. This has been determined to be impractical. The current approach is to divide cities into four categories: small, medium, large and extra-large. Each is associated with an average population for the group, which provides the strength of attraction. Centers of this attraction are then determined for each separate site and are typically the densest area near the center of the city or town. The automatic processing then consists of computing minimal travel times from each cell in the analysis area to the nearest center of attraction in each category. These are converted to attractiveness values, which are the population size divided by the square of the distance from the attraction center.

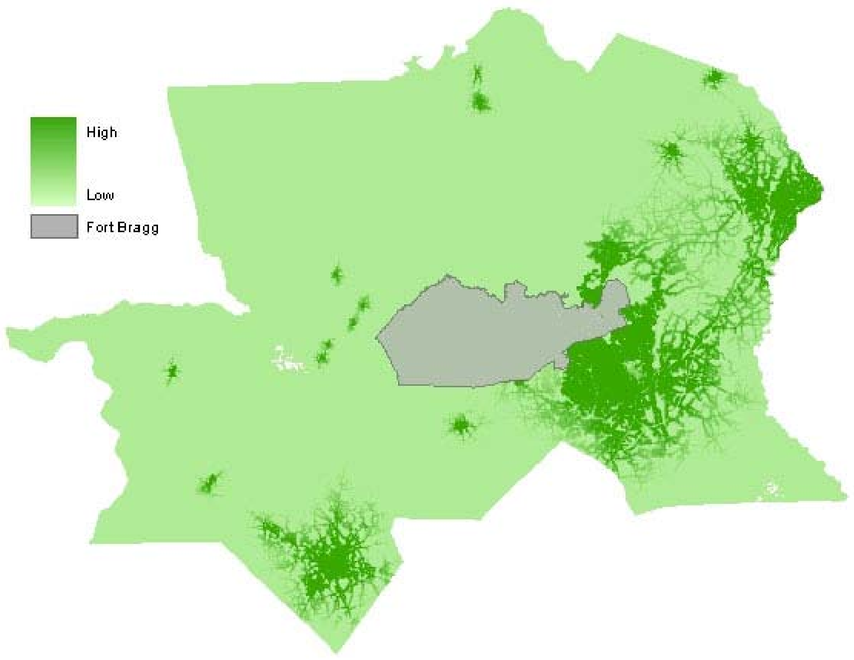


Figure 10. Attraction to cities.

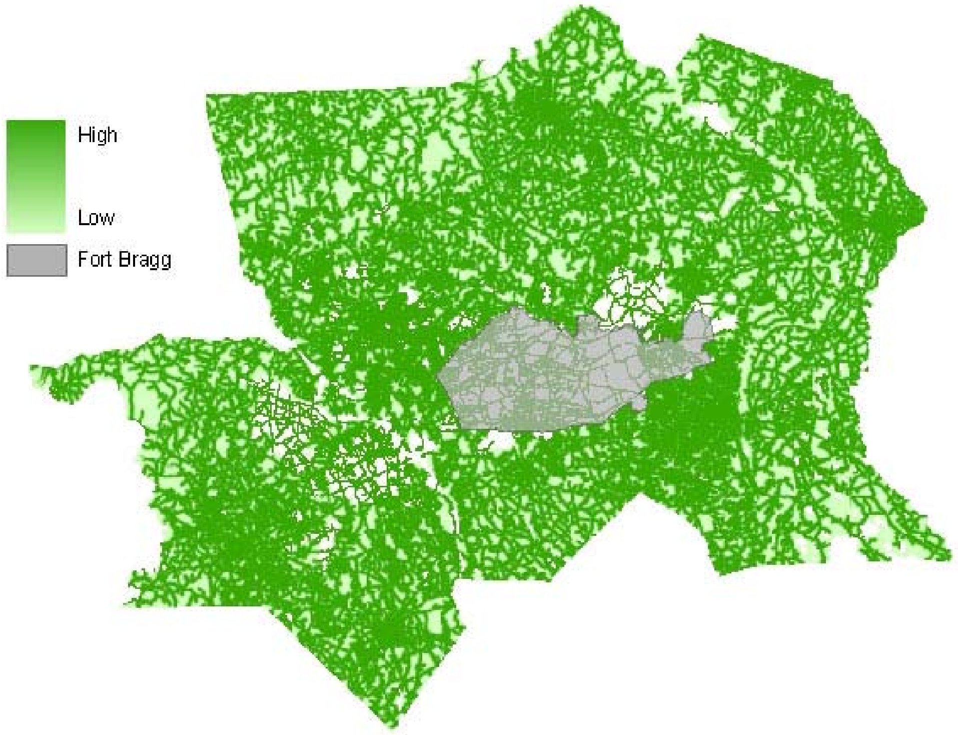


Figure 11. Attraction to roads.

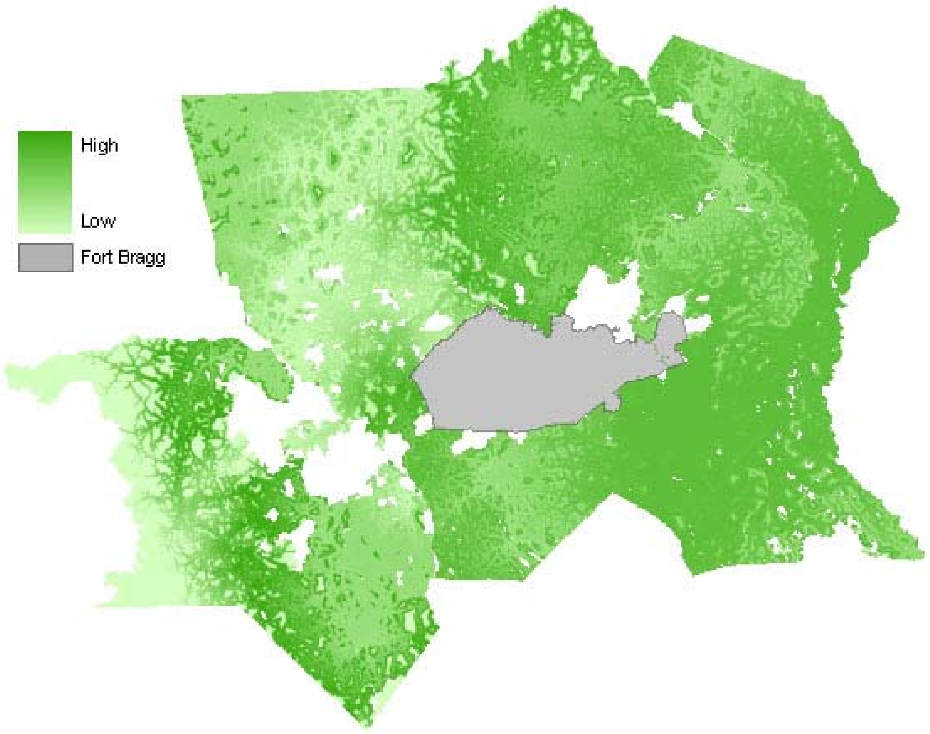


Figure 12. Attraction to ramps on limited access highways.

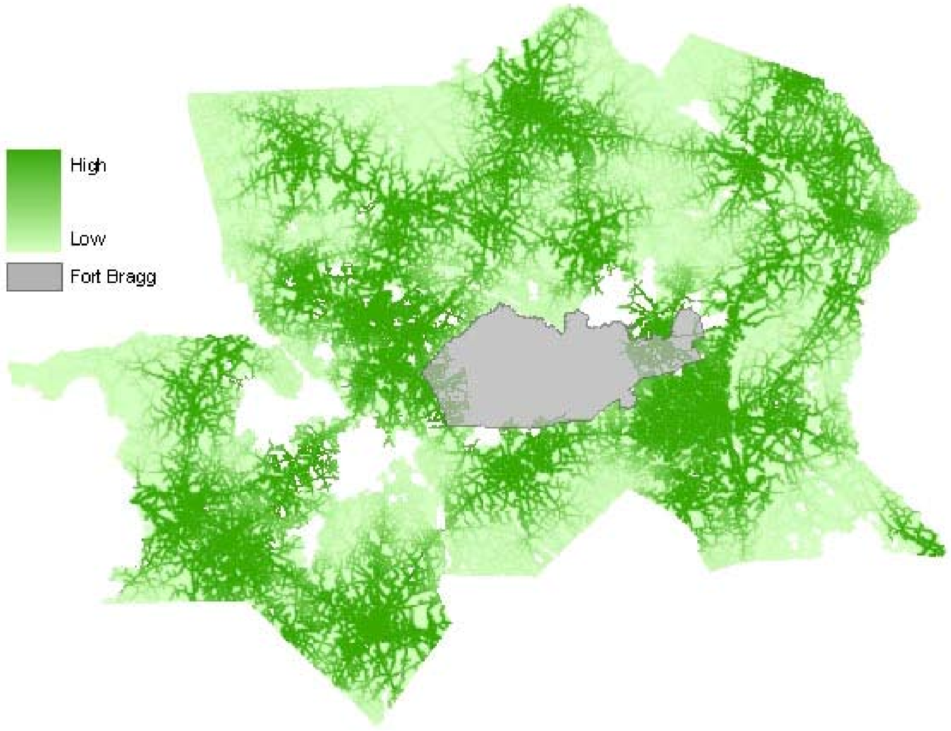


Figure 13. Attraction to major intersections.

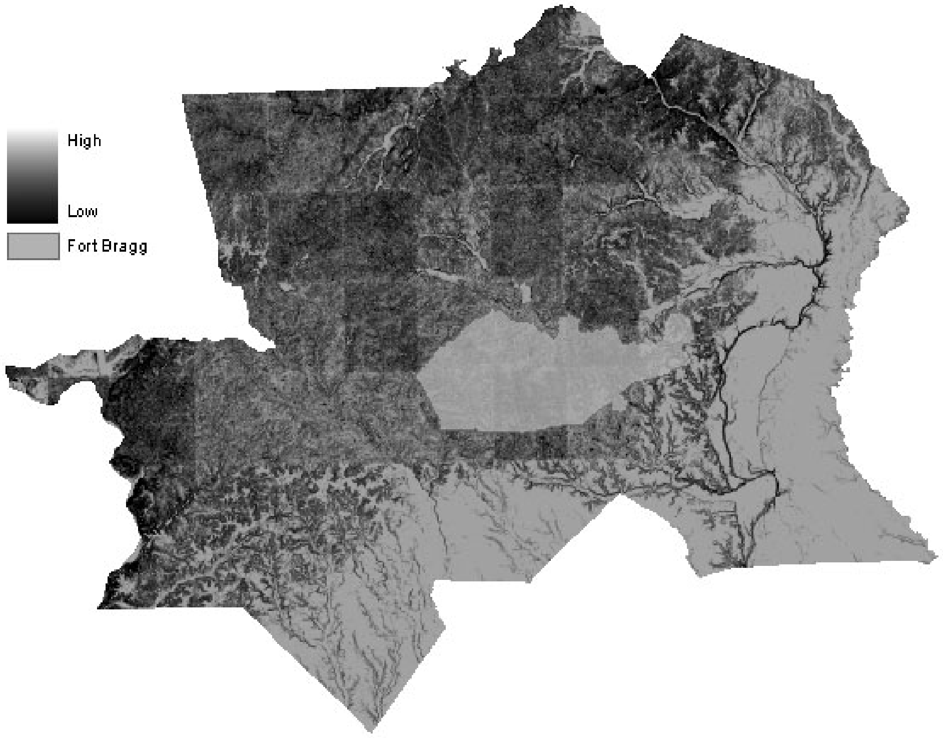


Figure 14. Attraction to different slopes.

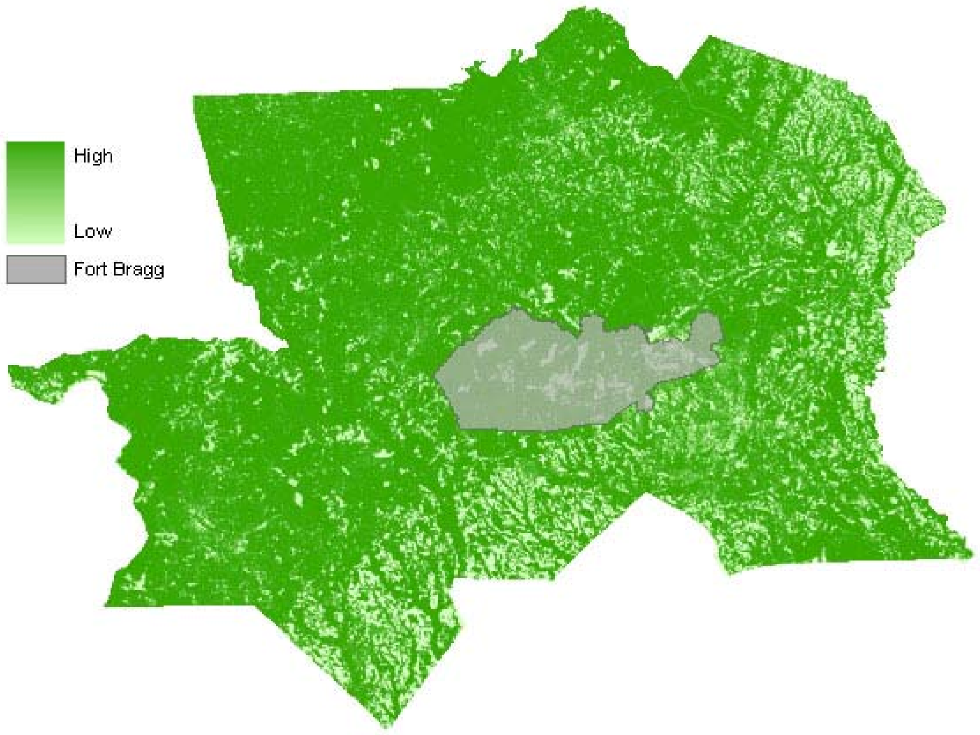


Figure 15. Attraction to lots bordering forest.

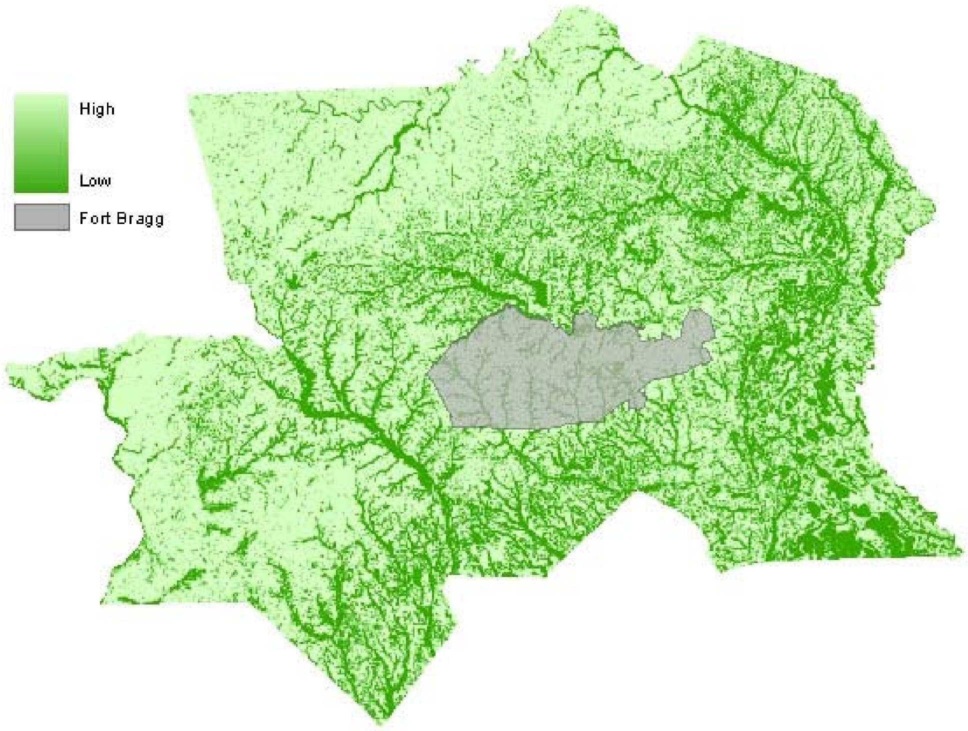


Figure 16. Attraction to lots bordering water.

The combination of the attractors, displayed in Figure 17, show significant attraction to new growth around Fayetteville and the Dunn/Erwin area located in the northeast corner. Zooming into the southern boundary of the installation and the western edge of Fayetteville, reveals significant preference areas close to the city. However, some attraction is visible associated with existing rural neighborhoods closer to the city.

The final map created in this effort (Figure 18) provides a relative index value for every location representing its attractiveness to residential growth after considering a number of location factors listed again here:

- location within the study area
- whether residential growth is possible
- current land use
- distance to water
- distance to forest
- slope
- distance to cities (city centers / employment centers)
- distance to county road
- distance to state/federal highway
- distance to limited access highway
- distance to nearest intersection.

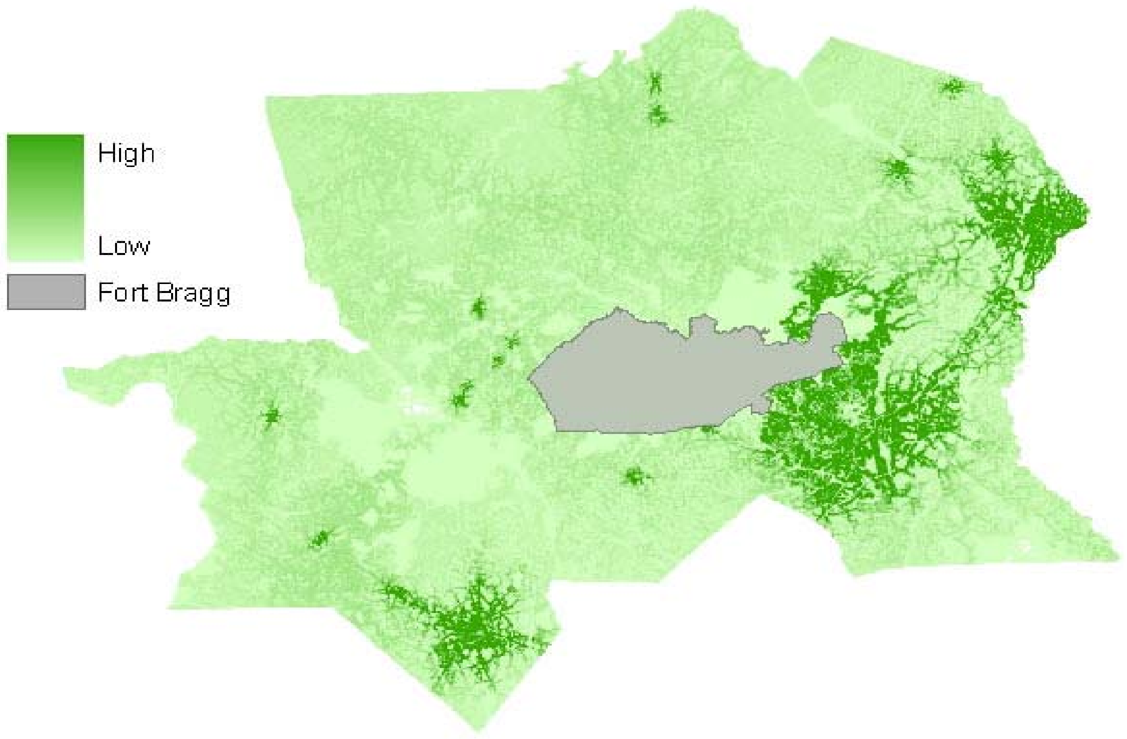


Figure 17. Summary attraction to residential development.

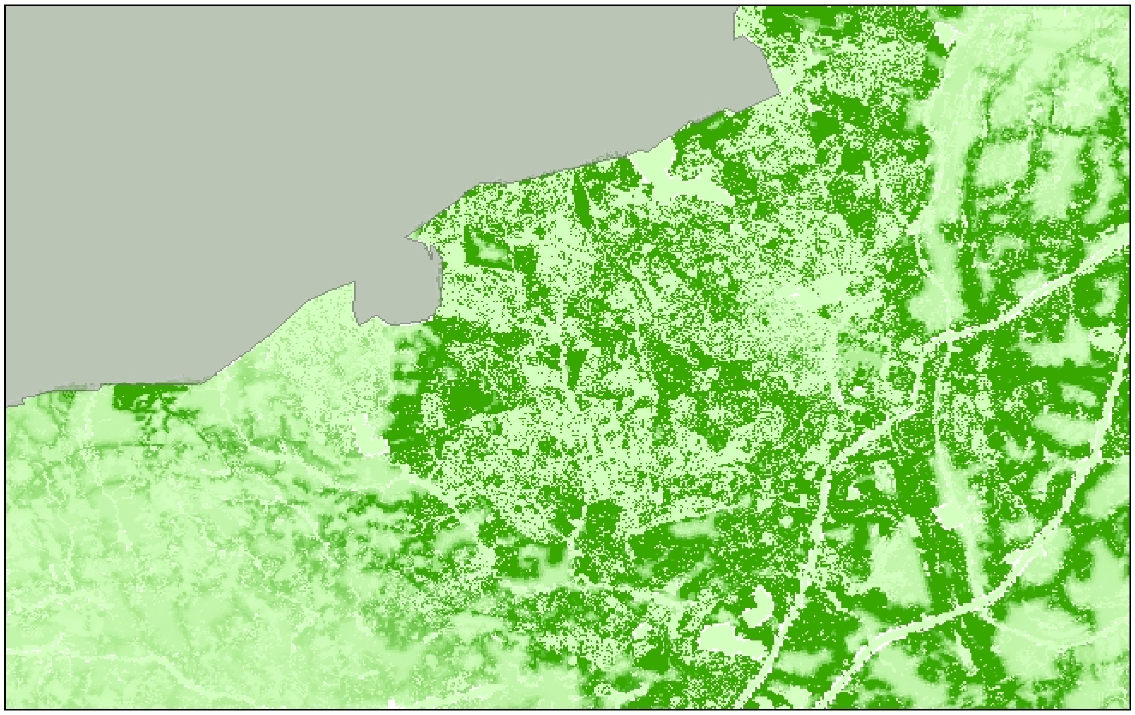


Figure 18. Summary attraction along southern border of Fort Bragg.

Several assumptions and caveats need to be recognized:

- Other potential attraction considerations could be added to this list and some on the list could be dropped for any particular location. Note, for example, that drinking water availability is not considered and can be critical from a legal standpoint (e.g., western water rights) or from a geologic perspective. Some areas offer more well water opportunities than others.
- Zoning is not considered in this analysis and can be important in the attraction of urban growth.
- This analysis assumes no new investments in roads or in the development of new neighborhoods, but can be rerun with such developments provided as inputs. It also does not recognize the affect of development affecting travel times on roads.
- Travel times are assumed to be optimal for each type of road. Hence, the resulting maps provide a snapshot in time of the attraction to new growth, but do not consider the impact of new growth on the overall attraction.
- The analysis identifies attractiveness to new development on a cell-by-cell basis – parcels approximately the size of city lots. While some growth happens this way, much development occurs as part of new

neighborhood development sites that can be roughly 800 meters square.

- Parcel size and ownership is not considered. Developers looking to build a new neighborhood are more likely to purchase a single large parcel rather than piece together many smaller contiguous parcels – making urban development less likely.
- Negative attractors are not considered in this analysis. Urban development tends to avoid being co-located with industrial sites.
- The attractiveness to urban growth can be outbid by attractiveness to other land uses such as parks and industrial areas. This competition is not identified here.
- The attractiveness to cities map must be developed in close consultation with local planners to best capture understandings of the location and attractiveness of local population, employment, and shopping centers.

6 Summary and Recommendations

This work has demonstrated an approach (process and procedure) that can be used in close coordination with regional planners to test the consequences of alternative regional plans using an inexpensive GIS-based approach for predicting future urban growth patterns that:

1. uses nationally available data
2. requires minimal data preparation
3. is readily calibrated
4. does not require time-series data
5. identifies the relative attractiveness of areas around military installations much better than the current image-processing based trend analyses and much less expensively than currently available urban growth simulation models.

(Note that results generated by the approach are not ready to be used in informing local planning processes.)

This effort is part of a larger effort that includes the development of a simulation model to project future landscape settlement patterns. The larger effort is called the Land use Evolution and impact Assessment Model (LEAM™) and supports the development of the LEAM Land Use Change (LEAM-LUC™) model. This model and the larger effort are addressing some of the caveats associated with the limited analysis presented here. In particular, LEAM-LUC™ considers the effect of development on future development and considers competition for turning undeveloped areas into residential, commercial, and open space.

The next available steps are:

1. Review the analysis process with local planners and modify inputs to better capture the local planning scenarios. This will allow creation of a development attractiveness map that best matches the local urbanization dynamics.
2. Test alternative planning scenarios that might include upgrading roads and highways or building new, closing roads, and zoning ideas.
3. Run the LEAM-LUC™ model to generate future scenarios that could develop in direct and indirect response to economic, population, and employment conditions.

Bibliography

Clarke, K. C., and L. Gaydos (1 November 1998). “Long Term Urban Growth Prediction Using a Cellular Automaton Model and GIS: Applications in San Francisco and Washington/Baltimore.” *International Journal of Geographical Information Science*. Vol 12, No. 7.

Deal, B. (2003). *LEAM™. Urban and Regional Planning*. Urbana/Champaign, University of Illinois.

Klosterman, R. (1999). “The What if? Collaborative Planning Support System.” *Environment and Planning, B: Planning and Design*. Vol. 26, pp 393-408.

Lozar, R. C., C.R. Ehlschlaeger, and J. Cox. (2003). *A Geographic Information Systems (GIS) and Imagery Approach to Historical Urban Growth Trends Around Military Installations*. ERDC/CERL TR-03-9. Champaign, IL: Engineer Research and Development Center Construction Engineering Research Laboratory (ERDC-CERL).

Netler, M., and H. Mitasova. (2004). *Open Source GIS: A GRASS GIS Approach*. 2nd ed. Boston/Dordrecht/London: Kluwer Academic Publishers.

U.S. Environmental Protection Agency (USEPA). (2000). *Projection Land-Use Change: A Summary of Models for Assessing the Effects of Community Growth and Change on Land-Use Patterns*. Cincinnati, OH: National Exposure Research Laboratory.

Westervelt, J.D., M. Shapiro, W.D. Goran, and D.P. Gerdes. (June 1992). *Geographic Resources Analysis Support System (GRASS) Version 4.0 User's Reference Manual*. Report Number N-87/22 rev./ADA255218. Champaign, IL: Construction Engineering Research Laboratory (CERL).

Appendix A: Unix Input File To Create LEAM Landuse Evolution Input Files Using GRASS Commands

```
#######################################################################################
# Last modified: Tue Apr 20 11:25:57 CDT 2004
#
# This file is used by the unix make program to create the
# LEAM landuse evolution input files using grass commands.
#
# To use,
#  Start GRASS and use a mapset that will contain the results
#  Create a directory named cell in this mapset (if not already there)
#  Copy this Makefile to that directory
#  Set the name of the needed input maps (see below)
#  Set the size of the cities (see below)
#  Set the attractor weights (see below)
#  From the cell directory, run make on this file
#######################################################################################
#  Input Maps and Map Categories
#  Make sure the input maps use the following categories ...
#    SMALL_CITY - Location of “small cities”
#        0 No small city
#        1 Small city
#    MED_CITY - Location of “medium cities”
#        0 No small city
#        1 Small city
#    LARGE_CITY - Location of “large cities”
#        0 No small city
#        1 Small city
#    NO_GROWTH - Explicit identification of areas of no growth
#        0 No restrictions
#        1 No growth areas
#    DEM
#        Categories are meters above sea level
#    MUNICIPAL_BOUNDARY
#        City boundaries
#    STUDY_AREA
#        0 Outside study area
#        1 Inside study area
#    INTERSTATES
#        0 No road
#        1 Limited access highway
#    OTHERROADS
#        0 No road
#        2 Federal highway
#        3 State highway
#        4 County/neighborhood road
#        5 (New?) county/neighborhood road
#        6 Ramps
#        7 Private road
#        8 Neighborhood road
#    LANDCOVER  - NLCD Landcover Map
#       11 Open water
#       12 Perennial Ice/Snow
#       21 Low Intensity Residential
#       22 High Intensity Residential
#       23 Commercial/Industrial/Transportation
#       31 Bare Rock/Sand/Clay
#       32 Quarries/Strip Mines/Gravel Pits
#       33 Transitional
#       41 Deciduous Forest
#       42 Evergreen Forest
#       43 Mixed Forest
#       51 Shrubland
#       61 Orchards/Vineyards/Other
#       71 Grasslands/Herbaceous
#       81 Pastures/Hay
#       82 Row Crops
#       83 Small Grains
#       84 Fallow
#       85 Urban/Recreational Grasses
#       91 Woody Wetlands
#       92 Emergent Herbaceous Wetlands
#
#############################################################################
# Provide names of the input maps (name@mapset)
SMALL_CITY=small_city@leamBase
MED_CITY=medium_city@leamBase
LARGE_CITY=large_city@leamBase
XL_CITY=xl_city@leamBase
NO_GROWTH=nogrowth@leamBase
DEM=dem@leamBase
MUNICIPAL_BOUNDARY=mnbnd@leamBase
STUDY_AREA=county@leamBase
LANDCOVER=landcover@leamBase
INTERSTATES=interstates@leamBase
OTHERROADS=otherroads@leamBase
#
#############################################################################
# Set the population average for large, medium, and small cities. These
# will be used to create the cities_attractor map, which combines these
# populations driving time maps.
XLPOP=150000
LARGEPOP=95000
MEDIUMPOP=50000
SMALLPOP=7000
POP2000=756544
DENSITY=1.68
#
#############################################################################
# Set the multipliers for residential growth drivers. These are multiplied
# times their respective 0-1 index maps. The results are summed and divided
# by the sum of the multipliers to give an overal 0-1 growth attractor
# index map.
# NOTE: DOES NOT MATCH THE GERTNER, ET AL. AND LUC EQUATION
intersect_prob_MULT=1
road_prob_MULT=1
slope_prob_MULT=1
ramp_prob_MULT=1
staterd_prob_MULT=1
forest_prob_MULT=1
water_prob_MULT=1
cities_prob_MULT=1
neighbor_prob_MULT=1

#############################################################################
#############################################################################
#############################################################################
# IDEALLY NO EDITING IS REQUIRED BELOW HERE
#############################################################################
#############################################################################
#############################################################################
```

```
LEmaps=landcover water_att forest_att road_att ramp_att cityBuff intersect_att staterd_att
slope_att noGrowth boundary cities_att2 neighbor_att cities_att2

IndexMaps=staterd_prob slope_prob water_prob forest_prob intersect_prob \
ramp_prob road_prob cities_prob

OtherMaps=hwyBuff road travelTime30 landcover municipalboundary \
intersection smallCity mediumCity largeCity xLargeCity dem travelSpeed30 travelTime90 \
xLargeCityTime largeCityTime mediumCityTime smallCityTime \
landTravelSpeed30 \
travelTime developable residential_att futureLanduse

LEfiles=$(shell echo ${LEmaps} | sed -e 's/ /,/g')
Otherfiles=$(shell echo ${OtherMaps} | sed -e 's/ /,/g')
Indexfiles=$(shell echo ${IndexMaps} | sed -e 's/ /,/g')

default:
	@echo make LEmaps - to create LUC input maps
	@echo make indexmaps - to create LUCs index maps
	@echo make residential_att - to create LUCs residential attractor map
	@echo make residential_prob - to create LUCs residential probability map
	@echo make futureLanduse - to create a future landuse map
	@echo make clean - to remove temporary maps
	@echo make veryclean - to remove all maps generated by this makefile

LEmaps: ${LEmaps}

indexmaps: ${IndexMaps}

######################################################################
######################################################################
# Fetching the base maps
######################################################################
######################################################################

# Fetches a map of the locations of the small cities
smallCity:
	@echo '#########################'; echo $@ ; date
	g.region res=30
	r.mapcalc 'smallCity0=if(${SMALL_CITY} <= 0,null(), ${SMALL_CITY})' > /dev/null 2>&1
	r.clump smallCity0 out=smallCity1
	# - Find centroids
	r.mapcalc 'smallCity2=if(isnull(smallCity1),null(),1)' > /dev/null 2>&1
	r.volume data=smallCity2 clump=smallCity1 site=smallCityCentroids

	# - convert to cells
	s.to.rast smallCityCentroids out=smallCity3 size=3
	r.null smallCity3 setnull=0
	r.mapcalc 'smallCity=if(isnull(smallCity3),null(),1)' > /dev/null 2>&1
	g.remove smallCity0,smallCity1,smallCity2,smallCity3

# Fetches a map of the locations of the medium cities
mediumCity:
	@echo '#########################'; echo $@ ; date
	g.region res=30
	r.mapcalc 'mediumCity0=if(${MED_CITY} < 0,0, ${MED_CITY})' > /dev/null 2>&1
	r.clump mediumCity0 out=mediumCity1
	# - Find centroids
	r.mapcalc 'mediumCity2=if(isnull(mediumCity1),null(),1)' > /dev/null 2>&1
	r.volume data=mediumCity2 clump=mediumCity1 site=mediumCityCentroids

	# - convert to cells
	s.to.rast mediumCityCentroids out=mediumCity3 size=3
	r.null mediumCity3 setnull=0
	r.mapcalc 'mediumCity=if(isnull(mediumCity3),null(),1)' > /dev/null 2>&1
	g.remove mediumCity0,mediumCity1,mediumCity2,mediumCity3

# Fetches a map of the locations of the large cities
largeCity:
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.mapcalc 'largeCity0=if(${LARGE_CITY} < 0,0, ${LARGE_CITY})' > /dev/null 2>&1
	r.clump largeCity0 out=largeCity1
	# - Find centroids
	r.mapcalc 'largeCity2=if(isnull(largeCity1),null(),1)' > /dev/null 2>&1
	r.volume data=largeCity2 clump=largeCity1 site=largeCityCentroids

	# - convert to cells
	s.to.rast largeCityCentroids out=largeCity3 size=3
	r.null largeCity3 setnull=0
	r.mapcalc 'largeCity=if(isnull(largeCity3),null(),1)' > /dev/null 2>&1
	g.remove largeCity0,largeCity1,largeCity2,largeCity3

# Fetches a map of the locations of the xl cities
xLargeCity:
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.mapcalc 'xLargeCity0=if(${XL_CITY} < 0,0, ${XL_CITY})' > /dev/null 2>&1
	r.clump xLargeCity0 out=xLargeCity1
	# - Find centroids
	r.mapcalc 'xLargeCity2=if(isnull(xLargeCity1),null(),1)' > /dev/null 2>&1
	r.volume data=xLargeCity2 clump=xLargeCity1 site=xLargeCityCentroids

	# - convert to cells
	s.to.rast xLargeCityCentroids out=xLargeCity3 size=3
	r.null xLargeCity3 setnull=0
	r.mapcalc 'xLargeCity=if(isnull(xLargeCity3),null(),1)' > /dev/null 2>&1
	g.remove xLargeCity0,xLargeCity1,xLargeCity2,xLargeCity3

# Fetches a map of no growth areas.
# Should be replaced with fetching maps of various no growth areas that can be
# edited and then merged.
noGrowth:
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.mapcalc 'noGrowth=if(round(${NO_GROWTH}))' > /dev/null 2>&1
	r.null noGrowth null=0

# Fetches the digital elevation model
dem:
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.mapcalc 'dem=${DEM}' > /dev/null 2>&1

# Create a municipal boundary map based on the municipal boundary in PERM
municipalboundary:
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.buffer in=${MUNICIPAL_BOUNDARY} out=municipalboundary dist=1.5 unit=miles

# Fetch the study area boundary from PERM
boundary:
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.mapcalc 'boundary=if(isnull(${STUDY_AREA}) || ${STUDY_AREA} ==0,null(),1)' > /dev/null
2>&1

# Fetches the landcover file from PERM and masks it with the study area boundary
landcover: boundary
```

```
        @echo '##########################'; echo $@ ; date
        g.region res=30
        r.mapcalc landcover='if(isnull(boundary),null(), \
          if(isnull(${LANDCOVER}),0,${LANDCOVER}))' > /dev/null 2>&1
        r.null landcover setnull=0
        (echo 11 110 127 177 ; \
        echo 21 253 229 228 ; \
        echo 22 247 178 159 ; \
        echo 23 229 86 78 ; \
        echo 31 210 205 192 ; \
        echo 32 175 175 177 ; \
        echo 33 83 62 118 ; \
        echo 41 133 199 126 ; \
        echo 42 56 129 78 ; \
        echo 43 212 231 176 ; \
        echo 51 220 202 143 ; \
        echo 61 187 174 118 ; \
        echo 71 253 233 170 ; \
        echo 81 251 246 93 ; \
        echo 82 202 145 70 ; \
        echo 83 121 108 74 ; \
        echo 84 244 238 202 ; \
        echo 85 240 156 54 ; \
        echo 91 200 230 248 ; \
        echo 92 100 179 213 ; \
        echo end ; ) | \
        r.colors landcover color=rules

# Fetches the road files
otherroads: boundary
        @echo '##########################'; echo $@ ; date
        g.region res=30
        r.mapcalc $@='if(isnull(boundary),null(),if(${OTHERROADS}==0,null(),${OTHERROADS}))' >
/dev/null 2>&1
        r.null $@ null=0

interstates: boundary
        @echo '##########################'; echo $@ ; date
        g.region res=30
        r.mapcalc $@='if(isnull(boundary),null(),if(${INTERSTATES}==0,null(),1))' > /dev/null
2>&1
        r.null $@ null=0

cross: otherroads interstates
        r.mapcalc cross='if(interstates==1 && otherroads==6, 1, 0)'

# Force state highways to connect to interstates
#road: otherroads interstates
#       # Identify state highway cells that border interstates as ramps
#       r.mapcalc 'road=if(interstates>0,interstates,otherroads)'
#       r.null road null=0

##########################################################################
##########################################################################
# The city attractor map is prepared in stages. The basic approach is
# as follows:
# Fetch the small, medium, and large city footprint maps
# For each (small, medium, and large)
#   Create road travel time map at 30 meter resolution
#   Create overland travel time map starting from times set above
#   Merge the two maps
# Combine the small, medium, and large cumulative maps
##########################################################################
##########################################################################
```

```
../LEAM/awkscript:
        @mkdir -p ../LEAM
        @echo 'BEGIN { FS = " "; cat=0;tot=0;developed=0;developable=0;gotone=0 }' >
../LEAM/awkscript
        @echo '{' >> ../LEAM/awkscript
        @echo '  if ($$1 != cat)' >> ../LEAM/awkscript
        @echo '  {' >> ../LEAM/awkscript
        @echo '    if (cat != 0 && cat != "*")' >> ../LEAM/awkscript
        @echo '    {' >> ../LEAM/awkscript
        @echo '      if (developable != 0)' >> ../LEAM/awkscript
        @echo '      {' >> ../LEAM/awkscript
        @echo '        if (gotone !=0) printf(", ") ;' >> ../LEAM/awkscript
        @echo '        printf("%f,%f", cat*MULT/20, developed/developable) ;' >>
../LEAM/awkscript
        @echo '        gotone += 1 ;' >> ../LEAM/awkscript
        @echo '      }' >> ../LEAM/awkscript
        @echo '    }' >> ../LEAM/awkscript
        @echo '    developed = 0; developable = 0 ; cat = $$1' >> ../LEAM/awkscript
        @echo '  }' >> ../LEAM/awkscript
        @echo '  if ($$2 == 2 || $$2 == 1) developable += $$3;' >> ../LEAM/awkscript
        @echo '  if ($$2 == 2 ) developed = $$3;' >> ../LEAM/awkscript
        @echo '  tot += $$3;' >> ../LEAM/awkscript
        @echo '}' >> ../LEAM/awkscript
        @echo 'END {printf("\n")}' >> ../LEAM/awkscript

# Combine the travel-time maps to get to nearest small, medium, and large
# city into one map using an inverse-distance weighting algorithm using the
# population of each type as the base weight.
cities_att: smallCityTime mediumCityTime largeCityTime xLargeCityTime developable
        @echo '############################'; echo $@ ; date
        g.region res=30
        r.mapcalc $@_t1='round(if(developable, \
          float(${XLPOP})/if(xLargeCityTime==0,1,xLargeCityTime) + \
          float(${LARGEPOP})/if(largeCityTime==0,1,largeCityTime) + \
          float(${MEDIUMPOP})/if(mediumCityTime==0,1,mediumCityTime) + \
          float(${SMALLPOP})/if(smallCityTime==0,1,smallCityTime),0))' \
           > /dev/null 2>&1
# This forces a 4-byte file for LEAMluc
        r.mapcalc cities_att='if(row()==1 && col()==1,33554432,$@_t1)' \
           > /dev/null 2>&1
        g.remove $@_t1

roadBuffer: otherroads
        @echo '############################'; echo $@ ; date
        r.buffer -z otherroads out=$@ \
        distance=30,60,90,120,150,180,210,240,270,300,330,360,390,420,450,480,510,540,570,600,630
,660,690

smallCityTimeRoad: cross intTravelTime30 othTravelTime30 smallCity
        @echo '############################'; echo $@ ; date
        g.region res=30
        /data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
        out=$@ xover=cross start_rast=smallCity percent=100 \
        > /dev/null 2>&1

# Create a travel cost surface for travel to small cities
smallCityTime: smallCityTimeRoad overlandTravelTime90
        @echo '############################'; echo $@ ; date
        g.region res=30
        r.null smallCityTimeRoad null=0
        r.mapcalc '$@_t1 = max( \
          smallCityTimeRoad[-1,-1],\
          smallCityTimeRoad[-1,0],\
          smallCityTimeRoad[-1,1],\
          smallCityTimeRoad[ 0,-1],\
          smallCityTimeRoad      ,\
          smallCityTimeRoad[ 0,1],\
          smallCityTimeRoad[ 1,-1],\
          smallCityTimeRoad[ 1,0],\
          smallCityTimeRoad[ 1,1])' > /dev/null 2>&1
        r.null $@_t1 setnull=0
        r.null smallCityTimeRoad setnull=0
        g.region res=90
        r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
        g.region res=30
        r.mapcalc smallCityTime='if(isnull(smallCityTimeRoad),$@_t2,smallCityTimeRoad)'
        g.remove $@_t1,$@_t2

mediumCityTimeRoad: cross intTravelTime30 othTravelTime30 mediumCity
        @echo '###########################'; echo $@ ; date
        g.region res=30
        /data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
        out=$@ xover=cross start_rast=mediumCity percent=100 \
        > /dev/null 2>&1

# Create a travel cost surface for travel to medium cities
mediumCityTime: mediumCityTimeRoad overlandTravelTime90
        @echo '###########################'; echo $@ ; date
        g.region res=30
        r.null mediumCityTimeRoad null=0
        r.mapcalc '$@_t1 = max( \
          mediumCityTimeRoad[-1,-1],\
          mediumCityTimeRoad[-1,0],\
          mediumCityTimeRoad[-1,1],\
          mediumCityTimeRoad[ 0,-1],\
          mediumCityTimeRoad      ,\
          mediumCityTimeRoad[ 0,1],\
          mediumCityTimeRoad[ 1,-1],\
          mediumCityTimeRoad[ 1,0],\
          mediumCityTimeRoad[ 1,1])' > /dev/null 2>&1
        r.null $@_t1 setnull=0
        r.null mediumCityTimeRoad setnull=0
        g.region res=90
        r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
        g.region res=30
        r.mapcalc mediumCityTime='if(isnull(mediumCityTimeRoad),$@_t2,mediumCityTimeRoad)'
        g.remove $@_t1,$@_t2

largeCityTimeRoad: cross intTravelTime30 othTravelTime30 largeCity
        @echo '###########################'; echo $@ ; date
        g.region res=30
        /data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
        out=$@ xover=cross start_rast=largeCity percent=100 \
        > /dev/null 2>&1

# Create a travel cost surface for travel to large cities
largeCityTime: largeCityTimeRoad overlandTravelTime90
        @echo '###########################'; echo $@ ; date
        g.region res=30
        r.null largeCityTimeRoad null=0
        r.mapcalc '$@_t1 = max( \
          largeCityTimeRoad[-1,-1],\
          largeCityTimeRoad[-1,0],\
          largeCityTimeRoad[-1,1],\
          largeCityTimeRoad[ 0,-1],\
          largeCityTimeRoad      ,\
          largeCityTimeRoad[ 0,1],\
          largeCityTimeRoad[ 1,-1],\
          largeCityTimeRoad[ 1,0],\
          largeCityTimeRoad[ 1,1])' > /dev/null 2>&1
```

```
	r.null $@_t1 setnull=0
	r.null largeCityTimeRoad setnull=0
	g.region res=90
	r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
	g.region res=30
	r.mapcalc largeCityTime='if(isnull(largeCityTimeRoad),$@_t2,largeCityTimeRoad)'
	g.remove $@_t1,$@_t2

xLargeCityTimeRoad: cross intTravelTime30 othTravelTime30 xLargeCity
	@echo '##########################'; echo $@ ; date
	g.region res=30
	/data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
	out=$@ xover=cross start_rast=xLargeCity percent=100 \
	> /dev/null 2>&1

# Create a travel cost surface for travel to xLarge cities
xLargeCityTime: xLargeCityTimeRoad overlandTravelTime90
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.null xLargeCityTimeRoad null=0
	r.mapcalc '$@_t1 = max( \
	  xLargeCityTimeRoad[-1,-1],\
	  xLargeCityTimeRoad[-1,0],\
	  xLargeCityTimeRoad[-1,1],\
	  xLargeCityTimeRoad[ 0,-1],\
	  xLargeCityTimeRoad      ,\
	  xLargeCityTimeRoad[ 0,1],\
	  xLargeCityTimeRoad[ 1,-1],\
	  xLargeCityTimeRoad[ 1,0],\
	  xLargeCityTimeRoad[ 1,1])' > /dev/null 2>&1
	r.null $@_t1 setnull=0
	r.null xLargeCityTimeRoad setnull=0
	g.region res=90
	r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
	g.region res=30
	r.mapcalc xLargeCityTime='if(isnull(xLargeCityTimeRoad),$@_t2,xLargeCityTimeRoad)'
	g.remove $@_t1,$@_t2

# Creates a slope map (in degrees) from the dem. Assumes the dem z values are meters
slope_att: dem
	@echo '##########################'; echo $@ ; date
	g.region res=30
	r.slope.aspect dem slope=slope_att_tmp
	r.mapcalc 'slope_att=round(slope_att_tmp)' > /dev/null 2>&1
	g.remove slope_att_tmp

# Create a travel cost surface for travel to ramps (road cat 6)
rampTimeRoad: cross intTravelTime30 othTravelTime30 otherroads interstates
	@echo '##########################'; echo $@ ; date
	# Look for ramp (cat 6) cells that border limited access roads (cat 1)
	g.region res=30
	r.mapcalc $@_t1='if(otherroads==6 && (\
	   interstates[-1,-1]==1 || interstates[-1,0]==1 || interstates[-1,1]==1 || \
	   interstates[ 0,-1]==1 ||                       interstates[ 0,1]==1 || \
	   interstates[ 1,-1]==1 || interstates[ 1,0]==1 || interstates[ 1,1]==1),1,null())' >
/dev/null 2>&1
	/data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
	out=$@ xover=cross start_rast=$@_t1 percent=100 \
	> /dev/null 2>&1
	g.remove $@_t1

# Create a travel cost surface for travel to ramps
ramp_att: rampTimeRoad overlandTravelTime90
	@echo '##########################'; echo $@ ; date
	g.region res=30
```

```
        r.null rampTimeRoad null=0
        r.mapcalc '$@_t1 = max( \
          rampTimeRoad[-1,-1],\
          rampTimeRoad[-1,0],\
          rampTimeRoad[-1,1],\
          rampTimeRoad[ 0,-1],\
          rampTimeRoad      ,\
          rampTimeRoad[ 0,1],\
          rampTimeRoad[ 1,-1],\
          rampTimeRoad[ 1,0],\
          rampTimeRoad[ 1,1])' > /dev/null 2>&1
        r.null $@_t1 setnull=0
        r.null rampTimeRoad setnull=0
        g.region res=90
        r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
        g.region res=30
        r.mapcalc ramp_att='if(isnull(rampTimeRoad),$@_t2,rampTimeRoad)'
        g.remove $@_t1,$@_t2

# Create a travel cost surface for travel to state roads
stateTimeRoad: cross intTravelTime30 othTravelTime30 otherroads
        @echo '############################'; echo $@ ; date
        g.region res=30
        # Look for state roads
        r.mapcalc $@_t1='if(otherroads==3,1,null())' > /dev/null 2>&1
        /data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
        out=$@ xover=cross start_rast=$@_t1 percent=100 \
        > /dev/null 2>&1
        g.remove $@_t1

# Create a travel cost surface for travel to ramps
staterd_att: stateTimeRoad overlandTravelTime90
        @echo '############################'; echo $@ ; date
        g.region res=30
        r.null stateTimeRoad null=0
        r.mapcalc '$@_t1 = max( \
          stateTimeRoad[-1,-1],\
          stateTimeRoad[-1,0],\
          stateTimeRoad[-1,1],\
          stateTimeRoad[ 0,-1],\
          stateTimeRoad      ,\
          stateTimeRoad[ 0,1],\
          stateTimeRoad[ 1,-1],\
          stateTimeRoad[ 1,0],\
          stateTimeRoad[ 1,1])' > /dev/null 2>&1
        r.null $@_t1 setnull=0
        r.null stateTimeRoad setnull=0
        g.region res=90
        r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
        g.region res=30
        r.mapcalc staterd_att='if(isnull(stateTimeRoad),$@_t2,stateTimeRoad)'
        g.remove $@_t1,$@_t2

# Create a travel cost surface for travel to county roads
countyTimeRoad: cross intTravelTime30 othTravelTime30 otherroads
        @echo '############################'; echo $@ ; date
        g.region res=30
        # Look for county roads
        r.mapcalc $@_t1='if(otherroads==4,1,null())' > /dev/null 2>&1
        /data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
        out=$@_t2 xover=cross start_rast=$@_t1 percent=100 \
        > /dev/null 2>&1
        r.mapcalc '$@=if($@_t2==0,othTravelTime30,$@_t2)'
#       g.remove $@_t1,$@_t2
# Create a travel cost surface for travel to ramps
road_att: countyTimeRoad overlandTravelTime90
        @echo '##########################'; echo $@ ; date
        g.region res=30
        r.null countyTimeRoad null=0
        r.mapcalc '$@_t1 = max( \
          countyTimeRoad[-1,-1],\
          countyTimeRoad[-1,0],\
          countyTimeRoad[-1,1],\
          countyTimeRoad[ 0,-1],\
          countyTimeRoad      ,\
          countyTimeRoad[ 0,1],\
          countyTimeRoad[ 1,-1],\
          countyTimeRoad[ 1,0],\
          countyTimeRoad[ 1,1])' > /dev/null 2>&1
        r.null $@_t1 setnull=0
        r.null countyTimeRoad setnull=0
        g.region res=90
        r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
        g.region res=30
        r.mapcalc road_att='if(isnull(countyTimeRoad),$@_t2,countyTimeRoad)'
        g.remove $@_t1,$@_t2

# Create a travel cost surface for travel to intersections
intersectTimeRoad: cross intTravelTime30 othTravelTime30 otherroads intersection
        @echo '##########################'; echo $@ ; date
        g.region res=30
        /data/FF/bin/r.cost.dks in=othTravelTime30 m2=intTravelTime30 \
        out=$@ xover=cross start_rast=intersection percent=100 \
        > /dev/null 2>&1

# Create a travel cost surface for travel to intersections
intersect_att: intersectTimeRoad overlandTravelTime90
        @echo '##########################'; echo $@ ; date
        g.region res=30
        r.null intersectTimeRoad null=0
        r.mapcalc '$@_t1 = max( \
          intersectTimeRoad[-1,-1],\
          intersectTimeRoad[-1,0],\
          intersectTimeRoad[-1,1],\
          intersectTimeRoad[ 0,-1],\
          intersectTimeRoad      ,\
          intersectTimeRoad[ 0,1],\
          intersectTimeRoad[ 1,-1],\
          intersectTimeRoad[ 1,0],\
          intersectTimeRoad[ 1,1])' > /dev/null 2>&1
        r.null $@_t1 setnull=0
        r.null intersectTimeRoad setnull=0
        g.region res=90
        r.cost  -r start_rast=$@_t1 input=overlandTravelTime90 output=$@_t2 percent=100
        g.region res=30
        r.mapcalc intersect_att='if(isnull(intersectTimeRoad),$@_t2,intersectTimeRoad)'
        g.remove $@_t1,$@_t2

# Finds state (2) and county  (3) road  intersections. Create a map of
# just state and county roads. Expand these locations by one cell and
# then thin them down to one cell (to take care of any small map errors).
# Then find cells that are surrounded by more than two cells containing
# roads.
intersection: otherroads
        @echo '##########################'; echo $@ ; date
        g.region res=30
        r.mapcalc '$@_t1=if((otherroads==2) || (otherroads==3), 1, 0)' > /dev/null 2>&1
        r.thin $@_t1 output=$@_t2
        r.null $@_t2 null=0
        r.mapcalc 'intersection=if($@_t2==1,if(2< \
          ($@_t2[-1,-1] + $@_t2[-1,0] + $@_t2[-1,1] + \
           $@_t2[ 0,-1]             + $@_t2[ 0,1] + \
           $@_t2[ 1,-1] + $@_t2[ 1,0] + $@_t2[ 1,1]),1,0))' > /dev/null 2>&1
        r.null intersection setnull=0
        g.remove $@_t1,$@_t2

# Create the municipal buffer file for LEAM
cityBuff: municipalboundary boundary
        @echo '##########################'; echo $@ ; date
        g.region res=30
        r.mapcalc 'cityBuff=if(isnull(boundary),1,if(isnull(municipalboundary),0,1))' > /dev/null
2>&1

# Create a surface with an index of a cell's juxtaposition with residential neighbors
neighbor_att: landcover
        @echo '##########################'; echo $@ ; date
        ( \
        echo TITLE     Find dense residential areas ;\
        echo MATRIX    29 ;\
        echo  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0 ;\
        echo  0  0  0  0  0  0  0  0  1  1  1  2  2  2  2  2  2  2  1  1  1  0  0  0  0  0  0  0  0 ;\
        echo  0  0  0  0  0  0  1  1  2  2  2  3  3  3  3  3  3  3  2  2  2  1  1  0  0  0  0  0  0 ;\
        echo  0  0  0  0  0  1  1  2  2  3  3  4  4  4  4  4  4  4  3  3  2  2  1  1  0  0  0  0  0 ;\
        echo  0  0  0  0  1  2  2  3  3  4  4  5  5  5  5  5  5  5  4  4  3  3  2  2  1  0  0  0  0 ;\
        echo  0  0  0  1  2  2  3  4  4  5  5  6  6  6  6  6  6  6  5  5  4  4  3  2  2  1  0  0  0 ;\
        echo  0  0  1  1  2  3  4  4  5  6  6  6  7  7  7  7  7  6  6  6  5  4  4  3  2  1  1  0  0 ;\
        echo  0  0  1  2  3  4  4  5  6  6  7  7  8  8  8  8  8  7  7  6  6  5  4  4  3  2  1  0  0 ;\
        echo  0  1  2  2  3  4  5  6  7  7  8  8  9  9  9  9  9  8  8  7  7  6  5  4  3  2  2  1  0 ;\
        echo  0  1  2  3  4  5  6  6  7  8  9  9 10 10 10 10 10  9  9  8  7  6  6  5  4  3  2  1  0 ;\
        echo  0  1  2  3  4  5  6  7  8  9  9 10 11 11 11 11 11 10  9  9  8  7  6  5  4  3  2  1  0 ;\
        echo  1  2  3  4  5  6  6  7  8  9 10 11 11 12 12 12 11 11 10  9  8  7  6  6  5  4  3  2  1 ;\
        echo  1  2  3  4  5  6  7  8  9 10 11 11 12 13 13 13 12 11 11 10  9  8  7  6  5  4  3  2  1 ;\
        echo  1  2  3  4  5  6  7  8  9 10 11 12 13 14 14 14 13 12 11 10  9  8  7  6  5  4  3  2  1 ;\
        echo  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1 ;\
        echo  1  2  3  4  5  6  7  8  9 10 11 12 13 14 14 14 13 12 11 10  9  8  7  6  5  4  3  2  1 ;\
        echo  1  2  3  4  5  6  7  8  9 10 11 11 12 13 13 13 12 11 11 10  9  8  7  6  5  4  3  2  1 ;\
        echo  1  2  3  4  5  6  6  7  8  9 10 11 11 12 12 12 11 11 10  9  8  7  6  6  5  4  3  2  1 ;\
        echo  0  1  2  3  4  5  6  7  8  9  9 10 11 11 11 11 11 10  9  9  8  7  6  5  4  3  2  1  0 ;\
        echo  0  1  2  3  4  5  6  6  7  8  9  9 10 10 10 10 10  9  9  8  7  6  6  5  4  3  2  1  0 ;\
        echo  0  1  2  2  3  4  5  6  7  7  8  8  9  9  9  9  9  8  8  7  7  6  5  4  3  2  2  1  0 ;\
        echo  0  0  1  2  3  4  4  5  6  6  7  7  8  8  8  8  8  7  7  6  6  5  4  4  3  2  1  0  0 ;\
        echo  0  0  1  1  2  3  4  4  5  6  6  6  7  7  7  7  7  6  6  6  5  4  4  3  2  1  1  0  0 ;\
        echo  0  0  0  1  2  2  3  4  4  5  5  6  6  6  6  6  6  6  5  5  4  4  3  2  2  1  0  0  0 ;\
        echo  0  0  0  0  1  2  2  3  3  4  4  5  5  5  5  5  5  5  4  4  3  3  2  2  1  0  0  0  0 ;\
        echo  0  0  0  0  0  1  1  2  2  3  3  4  4  4  4  4  4  4  3  3  2  2  1  1  0  0  0  0  0 ;\
        echo  0  0  0  0  0  0  1  1  2  2  2  3  3  3  3  3  3  3  2  2  2  1  1  0  0  0  0  0  0 ;\
        echo  0  0  0  0  0  0  0  0  1  1  1  2  2  2  2  2  2  2  1  1  1  0  0  0  0  0  0  0  0 ;\
        echo  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0 ;\
        echo DIVISOR   1 ;\
        echo TYPE      P ;\
        ) > /tmp/filter$$

        g.region res=30
        # select urban residential areas
        r.mapcalc  $@_t1='if(landcover==21,1,if(landcover==22,2,0))'
        r.mfilter input=$@_t1 output=neighbor_att filter=/tmp/filter$$
        g.remove $@_t1
        rm /tmp/filter$$

# Create a distance surface to the nearest cell containing water.
water_att: landcover
        @echo '##########################'; echo $@ ; date
        g.region res=30
```

```
	r.mapcalc $@_t1='if(landcover==11 || landcover==91 || landcover==92,1,0)' > /dev/null
2>&1
	r.null $@_t1 setnull=0
	r.buffer $@_t1 output=$@_t2 distances=30,60,90,120
	r.mapcalc water_att='if(isnull($@_t2),127,($@_t2-1)*30)' > /dev/null 2>&1
	g.remove $@_t1,$@_t2

# Create a distance surface to the nearest cell containing forest -
forest_att: landcover
	@echo '##############################'; echo $@ ; date
	g.region res=30
	r.mapcalc $@_t1='if(landcover==41 || landcover==42 || \
	landcover==43 || landcover==91,1,0)' > /dev/null 2>&1
	r.null $@_t1 setnull=0
	r.buffer $@_t1 output=$@_t2 distances=30,60,90,120
	r.mapcalc forest_att='if(isnull($@_t2),127,($@_t2-1)*30)' > /dev/null 2>&1
	g.remove $@_t1,$@_t2

##########################################################################################
##########################################################################################
# Travel time maps generated below at the 30 and 90 meter resolution
##########################################################################################
##########################################################################################
# Creates a travel time across 90 meter cells by choosing the shortest time
# from among the 9 associated 30 meter cells. It preserves travel time along
# road cat 1 (limited access) and preserves the highway buffer
overlandTravelTime90: overlandTravelSpeed90
	@echo '##############################'; echo $@ ; date
	g.region res=90
	# 90m/cell * 1mile/1609meter * 60min/hr = 3.35538min.mile/cell.hr
	# 3.35538min.mile/cell.hr / x miles/hr = Y min/cell
	r.mapcalc 'overlandTravelTime90=if(overlandTravelSpeed90>0,3.35538/overlandTravelSpeed90,
null())' > /dev/null 2>&1

# Computes the time required to traverse a cell in the N-S or E-W axes
roadTime30: roadSpeed30
	@echo '##############################'; echo $@ ; date
	g.region res=30
	# 30m/cell * 1mile/1609meter * 60min/hr = 1.11846min.mile/cell.hr
	# 1.11846min.mile/cell.hr / x miles/hr = Y min/cell
	r.mapcalc 'roadTime30=if(roadSpeed30>0,1.11846/roadSpeed30, null())' > /dev/null 2>&1

#travelTime30: travelSpeed30
#	@echo '##############################'; echo $@ ; date
#	g.region res=30
#	# 30m/cell * 1mile/1609meter * 60min/hr = 1.11846min.mile/cell.hr
#	# 1.11846min.mile/cell.hr / x miles/hr = Y min/cell
#	r.mapcalc 'travelTime30=if(travelSpeed30>0,1.11846/travelSpeed30, null())' > /dev/null
2>&1

othTravelTime30: othTravelSpeed30
	@echo '##############################'; echo $@ ; date
	g.region res=30
	# 30m/cell * 1mile/1609meter * 60min/hr = 1.11846min.mile/cell.hr
	# 1.11846min.mile/cell.hr / x miles/hr = Y min/cell
	r.mapcalc 'othTravelTime30=if(othTravelSpeed30>0,1.11846/othTravelSpeed30, null())' >
/dev/null 2>&1

intTravelTime30: intTravelSpeed30
	@echo '##############################'; echo $@ ; date
	g.region res=30
	# 30m/cell * 1mile/1609meter * 60min/hr = 1.11846min.mile/cell.hr
	# 1.11846min.mile/cell.hr / x miles/hr = Y min/cell
	r.mapcalc 'intTravelTime30=if(intTravelSpeed30>0,1.11846/intTravelSpeed30, null())' >
/dev/null 2>&1
```

```
overlandTravelSpeed90: hwyBuff landTravelSpeed30
        @echo '###########################'; echo $@ ; date
        g.region res=30
        r.mapcalc overlandTravelSpeed90='eval( \
        highLand = max( \
          landTravelSpeed30[-1,-1],\
          landTravelSpeed30[-1,0],\
          landTravelSpeed30[-1,1],\
          landTravelSpeed30[ 0,-1],\
          landTravelSpeed30      ,\
          landTravelSpeed30[ 0,1],\
          landTravelSpeed30[ 1,-1],\
          landTravelSpeed30[ 1,0],\
          landTravelSpeed30[ 1,1]), \
        haveBuffer = if( \
          hwyBuff[-1,-1]==1 || hwyBuff[-1,0]==1 || hwyBuff[-1,1]==1 || \
          hwyBuff[ 0,-1]==1 || hwyBuff==1       || hwyBuff[ 0,1]==1 || \
          hwyBuff[ 1,-1]==1 || hwyBuff[ 1,0]==1 || hwyBuff[ 1,1]==1, 1, 0), \
          if(haveBuffer,0,highLand))' > /dev/null 2>&1

intTravelSpeed30: interstates
        @echo '###########################'; echo $@ ; date
        g.region res=30
        r.mapcalc intTravelSpeed30='\
          if(interstates==1,70,0)' > /dev/null 2>&1

othTravelSpeed30: otherroads
        @echo '###########################'; echo $@ ; date
        g.region res=30
        r.mapcalc othTravelSpeed30='\
          if(otherroads==1,70, \
          if(otherroads==2,50, \
          if(otherroads==3,40, \
          if(otherroads==4,30, \
          if(otherroads==5,30, \
          if(otherroads==6,30, \
          if(otherroads==7,10, \
          if(otherroads==8,20,0))))))))' > /dev/null 2>&1

landTravelSpeed30: landcover noGrowth
        @echo '###########################'; echo $@ ; date
        g.region res=30
        r.mapcalc landTravelSpeed30='\
          if(noGrowth==1, 0, \
          if(landcover==21 || \
             landcover==22 || \
             landcover==23 ,.5, \
          if(landcover==11 || \
             landcover==12 || \
             landcover==91 || \
             landcover==92, 0, 1)))' > /dev/null 2>&1
#       r.mapcalc landTravelSpeed30='\
#         if(landcover==21,10, \
#         if(landcover==22,10, \
#         if(landcover==23,10,null())))' \ > /dev/null 2>&1
#
#travelableLandSpeed: landcover
#       r.mapcalc travelableLandSpeed=' \
#         if(noGrowth!=1 && \
#            landcover!=11 && \
#            landcover!=12 && \
#            landcover!=91 && \
#            landcover!=92, 1, 0)' > /dev/null 2>&1
# Creates a map of 1-cell wide buffers along limited access highways
#  except where other roads intersect. Problems:
#  - No distinction between bridges and true intersections
#  - Parallel roads in the next cell are considered access points

hwyBuff: interstates otherroads
        @echo '############################'; echo $@ ; date
        g.region res=30
        r.mapcalc hwyBuff='\
        if(interstates == 1 || otherroads == 6, 0,\
        if(interstates[-1,-1]==1 || interstates[-1,0]==1 || interstates[-1,1]==1 || \
        interstates[ 0,-1]==1 ||                  interstates[ 0,1]==1 || \
        interstates[ 1,-1]==1 || interstates[ 1,0]==1 || interstates[ 1,1]==1, 1, 0))' >
/dev/null 2>&1

##############################################################################
##############################################################################
# The following instructions generate index (0-1) maps for each
# attractiveness factor using the following approach:
# 1 - cross-list the attractiveness level to starting residential patterns
#     and potential developable areas.
# 2 - associate attractiveness level with percent developed
# 3 - convert attractiveness map to percent developed (0-1 index)
#
# This is accomplished with these steps:
# 1 - Find high value in attractiveness map (r.describe -r)
# 2 - Divide the map by this value; multiply by 10 and round to create
#     10 categories (r.mapcalc)
# 3 - Cross-reference the categories with the number of residential and
#     developable cells (r.coin)
# 4 - Process the results to create a “graph” (awk)
# 5 - Apply the graph to the attractiveness map to create the index map
##############################################################################
##############################################################################

cities_att2: cities_att
        r.mapcalc cities_att2='log(if(row()==1 && col()==1,0,cities_att))'

developable: landcover noGrowth hwyBuff
        @echo '############################'; echo $@ ; date
        r.mapcalc developable= \
          'if(hwyBuff || noGrowth || landcover==11 || landcover==24 || \
          landcover==91 || landcover==92 || landcover==85, 0, \
          if(landcover==21 || landcover==22 || landcover==24, 2, 1))' \
          > /dev/null 2>&1

../LEAM/XXcitiesGraph: ../LEAM/awkscript developable cities_att2
        @echo '############################'; echo $@ ; date
        @ rm -f ../LEAM/XXcitiesHigh
        @ mkdir -p ../LEAM
        ## Normalize the map
        # find highest value in map
        r.describe cities_att2 -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXcitiesHigh
        # divide map by highest value - multiply by some number (e.g., 20)
        r.mapcalc XXcitiesGraph_t1='round((20 * cities_att2)/'`cat ../LEAM/XXcitiesHigh`')' >
/dev/null 2>&1
        # generate awkscript for processing r.stats output
        sed -e s:MULT:`cat ../LEAM/XXcitiesHigh`: ../LEAM/awkscript > ../LEAM/awk-cities
        # apply graph to attractiveness map to create index map
        r.stats -c XXcitiesGraph_t1,developable | awk -f ../LEAM/awk-cities >
../LEAM/XXcitiesGraph
        g.remove XXcitiesGraph_t1

cities_prob: cities_att2 ../LEAM/XXcitiesGraph
```

```
        r.mapcalc 'cities_prob=graph(float(cities_att2), '`cat ../LEAM/XXcitiesGraph`')' >
/dev/null 2>&1
        (echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
        echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXroadGraph: ../LEAM/awkscript developable road_att2
        @echo '##########################'; echo $@ ; date
        @ rm -f ../LEAM/XXroadHigh
        @ mkdir -p ../LEAM
        ## Normalize the map
        # find highest value in map
        r.describe road_att2 -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXroadHigh
        # divide map by highest value - multiply by some number (e.g., 20)
        r.mapcalc XXroadGraph_t1='round((20 * road_att2)/'`cat ../LEAM/XXroadHigh`')' > /dev/null
2>&1
        # generate awkscript for processing r.stats output
        sed -e s:MULT:`cat ../LEAM/XXroadHigh`: ../LEAM/awkscript > ../LEAM/awk-road
        # apply graph to attractiveness map to create index map
        r.stats -c XXroadGraph_t1,developable | awk -f ../LEAM/awk-road > ../LEAM/XXroadGraph
        g.remove XXroadGraph_t1

road_att2: road_att
        r.mapcalc 'road_att2=log(10*road_att)'

road_prob: road_att2 developable ../LEAM/XXroadGraph
        r.mapcalc 'road_prob=graph(float(road_att2), '`cat ../LEAM/XXroadGraph`')' > /dev/null
2>&1
        (echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
        echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXrampGraph: ../LEAM/awkscript developable ramp_att2
        @echo '##########################'; echo $@ ; date
        @ rm -f ../LEAM/XXrampHigh
        @ mkdir -p ../LEAM
        ## Normalize the map
        # find highest value in map
        r.describe ramp_att2 -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXrampHigh
        # divide map by highest value - multiply by some number (e.g., 20)
        r.mapcalc XXrampGraph_t1='round((20 * ramp_att2)/'`cat ../LEAM/XXrampHigh`')' > /dev/null
2>&1
        # generate awkscript for processing r.stats output
        sed -e s:MULT:`cat ../LEAM/XXrampHigh`: ../LEAM/awkscript > ../LEAM/awk-ramp
        # apply graph to attractiveness map to create index map
        r.stats -c XXrampGraph_t1,developable | awk -f ../LEAM/awk-ramp > ../LEAM/XXrampGraph
        g.remove XXrampGraph_t1

ramp_att2: ramp_att
        r.mapcalc 'ramp_att2=log(10*ramp_att)'

ramp_prob: ramp_att2 developable ../LEAM/XXrampGraph
        r.mapcalc 'ramp_prob=graph(float(ramp_att2), '`cat ../LEAM/XXrampGraph`')' > /dev/null
2>&1
        (echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
        echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXintersectGraph: ../LEAM/awkscript developable intersect_att2
        @echo '##########################'; echo $@ ; date
        @ rm -f ../LEAM/XXintersectHigh
        @ mkdir -p ../LEAM
        ## Normalize the map
        # find highest value in map
        r.describe intersect_att2 -r -i -q | grep thru | sed -e 's/.* //' >
../LEAM/XXintersectHigh
        # divide map by highest value - multiply by some number (e.g., 20)
```

```
	r.mapcalc XXintersectGraph_t1='round((20 * intersect_att2)/'`cat
../LEAM/XXintersectHigh`')' > /dev/null 2>&1
	# generate awkscript for processing r.stats output
	sed -e s:MULT:`cat ../LEAM/XXintersectHigh`: ../LEAM/awkscript > ../LEAM/awk-intersect
	# apply graph to attractiveness map to create index map
	r.stats -c XXintersectGraph_t1,developable | awk -f ../LEAM/awk-intersect >
../LEAM/XXintersectGraph
	g.remove XXintersectGraph_t1

intersect_att2: intersect_att
	r.mapcalc intersect_att2='log(10*intersect_att)' ;

intersect_prob: intersect_att2 developable ../LEAM/XXintersectGraph
	r.mapcalc 'intersect_prob=graph(float(intersect_att2), '`cat ../LEAM/XXintersectGraph`')'
> /dev/null 2>&1
	(echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
	echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXforestGraph: ../LEAM/awkscript developable forest_att
	@echo '##########################'; echo XXforestGraph ; date
	@ rm -f ../LEAM/XXforestHigh
	@ mkdir -p ../LEAM
	## Normalize the map
	# find highest value in map
	r.describe forest_att -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXforestHigh
	# divide map by highest value - multiply by some number (e.g., 20)
	r.mapcalc XXforestGraph_t1='round((20 * forest_att)/'`cat ../LEAM/XXforestHigh`')' >
/dev/null 2>&1
	# generate awkscript for processing r.stats output
	sed -e s:MULT:`cat ../LEAM/XXforestHigh`: ../LEAM/awkscript > ../LEAM/awk-forest
	# apply graph to attractiveness map to create index map
	r.stats -c XXforestGraph_t1,developable | awk -f ../LEAM/awk-forest >
../LEAM/XXforestGraph
	g.remove XXforestGraph_t1

forest_prob: forest_att developable ../LEAM/XXforestGraph
	r.mapcalc 'forest_prob=graph(float(forest_att), '`cat ../LEAM/XXforestGraph`')' >
/dev/null 2>&1
	(echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
	echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXwaterGraph: ../LEAM/awkscript developable water_att
	@echo '##########################'; echo XXwaterGraph ; date
	@ rm -f ../LEAM/XXwaterHigh
	@ mkdir -p ../LEAM
	## Normalize the map
	# find highest value in map
	r.describe water_att -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXwaterHigh
	# divide map by highest value - multiply by some number (e.g., 20)
	r.mapcalc XXwaterGraph_t1='round((20 * water_att)/'`cat ../LEAM/XXwaterHigh`')' >
/dev/null 2>&1
	# generate awkscript for processing r.stats output
	sed -e s:MULT:`cat ../LEAM/XXwaterHigh`: ../LEAM/awkscript > ../LEAM/awk-water
	# apply graph to attractiveness map to create index map
	r.stats -c XXwaterGraph_t1,developable | awk -f ../LEAM/awk-water > ../LEAM/XXwaterGraph
	g.remove XXwaterGraph_t1

water_prob: water_att developable ../LEAM/XXwaterGraph
	r.mapcalc 'water_prob=graph(float(water_att), '`cat ../LEAM/XXwaterGraph`')' > /dev/null
2>&1
	(echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
	echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXneighborGraph: ../LEAM/awkscript developable neighbor_att
	@echo '##########################'; echo XXneighborGraph ; date
```

```
        @ rm -f ../LEAM/XXneighborHigh
        @ mkdir -p ../LEAM
        ## Normalize the map
        # find highest value in map
        r.describe neighbor_att -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXneighborHigh
        # divide map by highest value - multiply by some number (e.g., 20)
        r.mapcalc XXneighborGraph_t1='round((20 * neighbor_att)/'`cat ../LEAM/XXneighborHigh`')'
> /dev/null 2>&1
        # generate awkscript for processing r.stats output
        sed -e s:MULT:`cat ../LEAM/XXneighborHigh`: ../LEAM/awkscript > ../LEAM/awk-neighbor
        # apply graph to attractiveness map to create index map
        r.stats -c XXneighborGraph_t1,developable | awk -f ../LEAM/awk-neighbor >
../LEAM/XXneighborGraph
        g.remove XXneighborGraph_t1

neighbor_prob: neighbor_att developable ../LEAM/XXneighborGraph
        r.mapcalc 'neighbor_prob=graph(float(neighbor_att), '`cat ../LEAM/XXneighborGraph`')' >
/dev/null 2>&1
        (echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
        echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXslopeGraph: ../LEAM/awkscript developable slope_att
        @echo '##############################'; echo $@ ; date
        @ rm -f ../LEAM/XXslopeHigh
        @ mkdir -p ../LEAM
        ## Normalize the map
        # find highest value in map
        r.describe slope_att -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXslopeHigh
        # divide map by highest value - multiply by some number (e.g., 20)
        r.mapcalc XXslopeGraph_t1='round((20 * slope_att)/'`cat ../LEAM/XXslopeHigh`')' >
/dev/null 2>&1
        # generate awkscript for processing r.stats output
        sed -e s:MULT:`cat ../LEAM/XXslopeHigh`: ../LEAM/awkscript > ../LEAM/awk-slope
        # apply graph to attractiveness map to create index map
        r.stats -c XXslopeGraph_t1,developable | awk -f ../LEAM/awk-slope > ../LEAM/XXslopeGraph
        g.remove XXslopeGraph_t1

slope_prob: slope_att developable ../LEAM/XXslopeGraph
        r.mapcalc 'slope_prob=graph(float(slope_att), '`cat ../LEAM/XXslopeGraph`')' > /dev/null
2>&1
        (echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
        echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

../LEAM/XXstaterdGraph: ../LEAM/awkscript developable staterd_att2
        @echo '##############################'; echo $@ ; date
        @ rm -f ../LEAM/XXstaterdHigh
        @ mkdir -p ../LEAM
        ## Normalize the map
        # find highest value in map
        r.describe staterd_att2 -r -i -q | grep thru | sed -e 's/.* //' > ../LEAM/XXstaterdHigh
        # divide map by highest value - multiply by some number (e.g., 20)
        r.mapcalc XXstateRoadGraph_t1='round((20 * staterd_att2)/'`cat ../LEAM/XXstaterdHigh`')'
> /dev/null 2>&1
        # generate awkscript for processing r.stats output
        sed -e s:MULT:`cat ../LEAM/XXstaterdHigh`: ../LEAM/awkscript > ../LEAM/awk-staterd
        # apply graph to attractiveness map to create index map
        r.stats -c XXstateRoadGraph_t1,developable | awk -f ../LEAM/awk-staterd >
../LEAM/XXstaterdGraph
        g.remove XXstateRoadGraph_t1

staterd_att2: staterd_att
        r.mapcalc staterd_att2='log(10*staterd_att)'

staterd_prob: staterd_att2 ../LEAM/XXstaterdGraph
```

```
	r.mapcalc 'staterd_prob=graph(float(staterd_att2), '`cat ../LEAM/XXstaterdGraph`')' >
/dev/null 2>&1
	(echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
	echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

######################################################################
# This combines residential attractors
# into an overall residential attractor map
# residential_att_mult: developable road_prob slope_prob ramp_prob \
#	staterd_prob forest_prob water_prob neighbor_prob\
#	cities_prob intersect_prob noGrowth landcover boundary
#	@echo '##########################'; echo $@ ; date
#	r.mapcalc residential_att_mult=eval'( \
#	  developable=if(developable!=0,1,0), \
#	  developable * road_prob * slope_prob * intersect_prob * \
#	  ramp_prob * staterd_prob * forest_prob * water_prob * \
#	  neighbor_prob * cities_prob )' > /dev/null 2>&1

residential_att: developable road_prob slope_prob ramp_prob \
	staterd_prob forest_prob water_prob neighbor_prob\
	cities_prob intersect_prob noGrowth landcover boundary
	@echo '##########################'; echo $@ ; date
	r.mapcalc residential_att=eval'( \
	  developable=if(developable>0,1,0), \
	  developable * (road_prob * ${road_prob_MULT} + \
	  slope_prob * ${slope_prob_MULT} + \
	  intersect_prob * ${intersect_prob_MULT} + \
	  ramp_prob * ${ramp_prob_MULT} + \
	  staterd_prob * ${staterd_prob_MULT} + \
	  forest_prob * ${forest_prob_MULT} + \
	  water_prob * ${water_prob_MULT} + \
	  cities_prob * ${cities_prob_MULT}) / \
	  (${intersect_prob_MULT} + ${road_prob_MULT} + \
	  ${slope_prob_MULT} + \
	  ${ramp_prob_MULT} + ${staterd_prob_MULT} + \
	  ${forest_prob_MULT} + ${water_prob_MULT} + \
	  ${cities_prob_MULT}))' > /dev/null 2>&1

../LEAM/XXresidentialGraph: ../LEAM/awkscript developable residential_att
	@echo '##########################'; echo $@ ; date
	@ rm -f ../LEAM/XXattract_res_High
	@ mkdir -p ../LEAM
	## Normalize the map
	# find highest value in map
	r.describe residential_att -r -q | sed -e 's/.*-//' -e 's/ *$$//' >
../LEAM/XXattract_res_High
	# divide map by highest value - multiply by some number (e.g., 20)
	r.mapcalc ResidentialGraph_t1='round((20 * residential_att)/'`cat
../LEAM/XXattract_res_High`')' > /dev/null 2>&1
	# generate awkscript for processing r.stats output
	sed -e s:MULT:`cat ../LEAM/XXattract_res_High`: ../LEAM/awkscript > ../LEAM/awk-
residential
	# apply graph to attractiveness map to create index map
	r.stats -c ResidentialGraph_t1,developable | awk -f ../LEAM/awk-residential >
../LEAM/XXresidentialGraph
	g.remove ResidentialGraph_t1

residential_prob: residential_att ../LEAM/XXresidentialGraph
	r.mapcalc 'residential_prob=graph(float(residential_att), '`cat
../LEAM/XXresidentialGraph`')' > /dev/null 2>&1
	(echo 0.0 white; echo .01 255 240 150; echo .05 yellow; echo .4 green; \
	echo 1.0 0 100 0;echo end) | r.colors $@ color=rules -q

# Evolve Landscape
futureLanduse: residential_prob landcover
```

```
        r.mapcalc newLanduse=if'((residential_prob > rand(0.0,'`r.describe -r residential_prob |
sed -e 's/.*-//'`')) && (50>rand(0,100)),21,null())' > /dev/null 2>&1
        r.mapcalc futureLanduse='if(landcover==21 || landcover==22,1,if(newLanduse==21,2,0))' >
/dev/null 2>&1
        (echo 0 white; echo 1 grey; echo 2 red; echo end) | \
        r.colors $@ color=rules -q

noiseTolerance: futureLanduse
        r.mapcalc $@_t1='if(isnull(futureLanduse),0,1)' > /dev/null 2>&1
        r.mapcalc $@_t2=' \
        $@_t1[-2,-2]+$@_t1[-2,-1]+$@_t1[-2,0]+$@_t1[-2,1]+$@_t1[-2,2]+ \
        $@_t1[-1,-2]+$@_t1[-1,-1]+$@_t1[-1,0]+$@_t1[-1,1]+$@_t1[-1,2]+ \
        $@_t1[ 0,-2]+$@_t1[ 0,-1]+$@_t1[ 0,0]+$@_t1[ 0,1]+$@_t1[ 0,2]+ \
        $@_t1[ 1,-2]+$@_t1[ 1,-1]+$@_t1[ 1,0]+$@_t1[ 1,1]+$@_t1[ 1,2]+ \
        $@_t1[ 2,-2]+$@_t1[ 2,-1]+$@_t1[ 2,0]+$@_t1[ 2,1]+$@_t1[ 2,2]' > /dev/null 2>&1
        g.region res=150
        r.decay $@_t2 out=noiseTolerance dist=1000 comp=.01 per=100
        g.region res=30

altfutureLanduse: residential_att
#       x=$(shell r.stats -c residential_att | sed -e 's/-[^ ]*/*/' -e 's/$$/+\\/' -e
's/^\*.*/0/' | bc ); \
#       echo $$x ; \
        r.mapcalc futureLanduse=if'(residential_att * ATTRACTOR_RES >\
        rand(0.0,1.0),21,null())' > /dev/null 2>&1

# Removes all the temporary files used in the creation of the LEAM inputs
clean:
        g.remove rast=${Otherfiles}

# Removes all the temporary and LEAM input files
veryclean: clean
        g.remove rast=${LEfiles}
        g.remove rast=${Indexfiles}
```

Appendix B: Acquiring and Processing Required GIS Data

The following provides a step-by-step cookbook for acquiring and processing required GIS data. Remember, however, that it is the end result that is important, not the steps to get there. These steps are provided as an approached that has worked, but—due to changes in available data, differences among GIS software, your hardware, and available skills—different and perhaps better approaches will be used.

The steps described below are partly accomplished with ESRI GIS software and partly with the public domain GRASS software.

ESRI Software Steps

Acquire land use data for the area of interest

Acquire GIS data layers from available Internet available sources.

• Land Cover:	USGS NLCD maps
• Digital Elevation:	USGS DEM
• Road Network:	Census Bureau Tiger files
• Property Ownership:	USGS, DoD, BLM
• Floodplain Areas:	FEMA
• County.	

Re-project all Data into the Same Equal Area Projection

While GRASS can process vector and raster GIS data in virtually any projection and coordinate system, it cannot simultaneously work with maps that are in different projections. Therefore ArcGIS is used to re-project all data into an equal-area coordinate system – preferably Universal Transverse Mercator (UTM). Also, choose a minimal bounding box to be your study area into which the maps will be re-projected.

Process the Road Data

Because GRASS generally works with single category values (rather than a table of attributes), vector data imported into GRASS must be associated with useful category values. The CFCC2 category codes associated with roads are string values such as “A1”, “A2”, “A3”, etc. These must be transformed to integer values by dropping the “A.” To accomplish this, edit the attributes table and add a new field that you will call CFCC3 as a short integer. In the attribute table editor for CFCC3, click on the String radio button. Then enter the text in the box: “Right (CFCC2, 1)” and run this request. That should create the road classes.

Create the “No Growth” map.

Use the Federal lands in the study area to generate the no growth Shapefile. Values of 0 represent areas where growth is allowed; a value of 1 is used for areas where growth is restricted. These lands are designated no growth because the government owns them and residential growth will not occur there.

Prepare for Copying to Your GRASS Environment

At this point, you should have the following maps all in the same UTM coordinate system:

Code	Name	Description
roads.shp	Roads	Vector shape file
nogrowth.shp	No Growth	Vector shape file
county.shp	Counties in study	Vector shape file
mnbnd	Municipal areas	City boundary shape file
landcover	Landcover	Raster grids with NLCD categories 0-99
DEM	DEM	Raster grids with elevations in meters

GRASS Software Steps

The Shapefiles and Rasters must be copied to the LINUX environment for processing by GRASS. For those using GRASS under Linux this will require copying the maps from a Windows machine to the LINUX machine. (For those running GRASS under CYGWIN that is operating under Windows, you may be able to access the same files directly without copying.) Visit your local system administrator for help using your systems, network, and software.

ESRI vector Shapefiles should be copied over to UNIX as a set of six files. These files will have the same name but different extensions (.dbf, .prj, .sbn, .sbx, .shp, .shx).

Raster grids should be copied over to UNIX as folders.

Create New GRASS Location

Start GRASS and choose to create a new location if this is the first time for this particular study area. For a location, choose a name without spaces (e.g., FortBragg), and choose the mapset called “PERMANENT.” Every GRASS location has a mapset called PERMANENT, which is reserved to contain any final maps. You will be asked to supply information such as the coordinate system, the projection, the cell resolution, and coordinates for the north, south, east, and west extent of the study area. You can find this information back in ArcCatalog by looking at the spatial attributes of your area into which you re-projected your maps (e.g., DEM or NLCD).

Enter the information when prompted: *Nad83, Data Transformation Parameter: 6, Zone __, N*, then enter extents a little big for the area**

It is very important to set the resolution to 30: meters and ensure that the difference between the N-S and E-W need to be equally divisible by 30. For example, if the N is 36,000 and the S is 30,000. So 36,000 - 30,000 = 6,000. Also, 6,000 / 30 = 200. Since it is a whole number, the resolution is 30.

Accept the region when prompted and then exit GRASS. You have now successfully set up a new GRASS location!

Import the ESRI Data

Start GRASS again and select the location you just created and select/create a new mapset that you must name “leamBase.” By convention, this mapset name will contain all of the base maps used for the land use change analysis. Also, by convention all map names will be specifically assigned to allow for various scripts to work properly. Change directory to where your data is and import the files you brought in using *r.in.shape* for Shapefiles and *r.in.gdal* for rasters as follows:

```
v.in.shape in=interstates.shp out=interstates
v.in.shape in=otherroads.shp out=otherroads cat=CFCC3
r.in.shape in=nogrowth.shp out=nogrowth
r.in.shape in=county.shp out=county
v.in.shape in=boundary.shp out=boundary
r.in.shape in=boundary.shp out=boundary
r.in.shape in=mnbnd out=mnbnd
r.in.gdal in=landcover out=landcover
r.in.gdal in=dem out=dem
```

These files will be stored in the leamBase mapset folder for the current location in GRASS. The files can be listed using “g.list rast” and “g.list vect”, assuming “leamBase” is in your mapset list (see “g.mapsets help”). Use the GRASS display commands (e.g., d.mon, d.rast, d.what.rast, d.vect, and d.what.vect) to check your results.

Create the “City” Files

Required analysis input maps include four “city” maps, each identifying attractor areas to urban development based on “city size.” These attractors are points of locally highest residential density based on the NLCD land-cover maps and discovered with the following GIS analysis script.

```
# This creates a surface of “connectedness to local residential”
# The filter looks up to 15 cells away and assigns an importance of residential depending
# on distance and type (e.g., the 1 or 2 value from above). It multiplies the two values
# for each cell together and adds the results for all neighboring cells.
r.mfilter input=t1 output=t2 filter=neighborhoodFilter

#### Next, find the center of the higher attracting areas
# First, find the highest values
r.mfilter input=t2 output=t3 filter=peakFilter
r.mapcalc t4='if(t2>(t3+400) && t2>0)'

# Then, find the center of each area.
# - Make clumps
r.clump t4 out=t5

# - Find centroids
r.volume data=t3 clump=t5 site=centroids

# - convert to cells
s.to.rast centroids out=t6 size=3

# Get original mfilter output numbers for each centroid
r.mapcalc t7='if(t6,t2,0)'

# Divide into 4 sets
r.mapcalc t8=round\(4\*t7/`r.describe -r t7 | grep thru | sed -e 's/.* //'`\)

# set to “small, medium, large, and x-large”
r.mapcalc small_city='if(t8<2,1,null())'
r.mapcalc medium_city='if(t8==2,1,null())'
r.mapcalc large_city='if(t8==3,1,null())'
r.mapcalc xl_city='if(t8==4,1,null())'
```

The two filters used in the script follow.
The first is “neighborhoodFilter,” which is used to perform a distance-weighted analysis of all residential areas around each cell using the r.mfilter program.

```
TITLE     Find dense urban areas
MATRIX    29
  0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 1 1 1 2 2 2 2 2 2 2 1 1 1 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 1 1 2 2 2 3 3 3 3 3 3 3 2 2 2 1 1 0 0 0 0 0 0
  0 0 0 0 0 1 1 2 2 3 3 4 4 4 4 4 4 4 3 3 2 2 1 1 0 0 0 0 0
  0 0 0 0 1 2 2 3 3 4 4 5 5 5 5 5 5 5 4 4 3 3 2 2 1 0 0 0 0
  0 0 0 1 2 2 3 4 4 5 5 6 6 6 6 6 6 6 5 5 4 4 3 2 2 1 0 0 0
  0 0 1 1 2 3 4 4 5 6 6 6 7 7 7 7 7 6 6 6 5 4 4 3 2 1 1 0 0
  0  0  1  2  3  4  4  5  6  6  7  7  8  8  8  8  8  7  7  6  6  5  4  4  3  2  1  0  0
  0  1  2  2  3  4  5  6  7  7  8  8  9  9  9  9  9  8  8  7  7  6  5  4  3  2  2  1  0
  0  1  2  3  4  5  6  6  7  8  9  9 10 10 10 10 10  9  9  8  7  6  6  5  4  3  2  1  0
  0  1  2  3  4  5  6  7  8  9  9 10 11 11 11 11 11 10  9  9  8  7  6  5  4  3  2  1  0
  1  2  3  4  5  6  6  7  8  9 10 11 11 12 12 12 11 11 10  9  8  7  6  6  5  4  3  2  1
  1  2  3  4  5  6  7  8  9 10 11 11 12 13 13 13 12 11 11 10  9  8  7  6  5  4  3  2  1
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 14 14 13 12 11 10  9  8  7  6  5  4  3  2  1
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 14 14 13 12 11 10  9  8  7  6  5  4  3  2  1
  1  2  3  4  5  6  7  8  9 10 11 11 12 13 13 13 12 11 11 10  9  8  7  6  5  4  3  2  1
  1  2  3  4  5  6  6  7  8  9 10 11 11 12 12 12 11 11 10  9  8  7  6  6  5  4  3  2  1
  0  1  2  3  4  5  6  7  8  9  9 10 11 11 11 11 11 10  9  9  8  7  6  5  4  3  2  1  0
  0  1  2  3  4  5  6  6  7  8  9  9 10 10 10 10 10  9  9  8  7  6  6  5  4  3  2  1  0
  0  1  2  2  3  4  5  6  7  7  8  8  9  9  9  9  9  8  8  7  7  6  5  4  3  2  2  1  0
  0  0  1  2  3  4  4  5  6  6  7  7  8  8  8  8  8  7  7  6  6  5  4  4  3  2  1  0  0
  0  0  1  1  2  3  4  4  5  6  6  6  7  7  7  7  7  6  6  6  5  4  4  3  2  1  1  0  0
  0  0  0  1  2  2  3  4  4  5  5  6  6  6  6  6  6  6  5  5  4  4  3  2  2  1  0  0  0
  0  0  0  0  1  2  2  3  3  4  4  5  5  5  5  5  5  5  4  4  3  3  2  2  1  0  0  0  0
  0  0  0  0  0  1  1  2  2  3  3  4  4  4  4  4  4  4  3  3  2  2  1  1  0  0  0  0  0
  0  0  0  0  0  0  1  1  2  2  2  3  3  3  3  3  3  3  2  2  2  1  1  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  1  1  1  2  2  2  2  2  2  2  1  1  1  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0
DIVISOR   1
TYPE      P
```

The second is peakFilter, which finds the average value in a neighborhood surrounding each cell.

```
TITLE     Find average in neighborhood
MATRIX  29
 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
DIVISOR  665
TYPE   P
```

Process the Roads Shape File

Roads are used in the LEAM processing to compute travel times and to provide context information in output display maps. To accommodate the roads file must be imported into GRASS as two vector files, interstates and other roads (non-interstates). These are then converted to raster files for later travel-time processing with the following commands:

```
# List the categories associated with the roads file
v.in.shape interstates.shp -d
v.in.shape otherroads.shp –d

# Imports the roads file into GRASS
v.in.shape interstates.shp out=interstates att -o
# Creates vector support files for interstates
v.support interstates

v.in.shape otherroads.shp out=otherroads att=CFCC3 -o
# Creates vector support files for otherroads
v.support otherroads

# Converts both to raster
v.to.rast interstates out=interstates
v.to.rast otherroads out=otherroads
```

Status

At this point the following maps should exist as part of the leamBase mapset:

Map name	Type	Description
small_city	Raster	Small City
medium_city	Raster	Medium City
large_city	Raster	Large City
xl_city	Raster	Extra Large City
nogrowth	Raster	No Growth
dem	Raster	DEM
mnbnd	Raster	Municipal Boundaries
county	Raster	Study Area / Counties
landcover	Raster	NLCD
boundary	Raster	Boundary of some area of interest
otherroads	Raster	Non-interstates
interstates	Raster	Interstate highways
boundary	Vector	Boundary of some area of interest
otherroads	Vector	Non-interstates
interstates	Vector	Interstate highways

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE (DD-MM-YYYY) 16-09-2006	**2. REPORT TYPE** Final	**3. DATES COVERED (From - To)**
4. TITLE AND SUBTITLE LEAMram™: Land use Evolution and impact Assessment Model Residential Attractiveness Model		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) James D. Westervelt and Joseph Rank		**5d. PROJECT NUMBER** SERDP
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER CS-1257
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Engineer Research and Development Center (ERDC) Construction Engineering Research Laboratory (CERL) PO Box 9005, Champaign, IL 61826-9005		**8. PERFORMING ORGANIZATION REPORT NUMBER** ERDC/CERL TR-06-28
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) SERDP Program Office 901 North Stuart Street, Suite 303 Arlington, VA 22203	Commander, Fort Bragg Fort Bragg, NC 28303	**10. SPONSOR/MONITOR'S ACRONYM(S)** SERDP
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This document describes a GIS approach that can quickly and inexpensively identify areas around military installations that are attractive to residential development – based on proposed regional investments and policies. Analysis results can then be used to inform regional planning and to help identify long-term impacts of residential development on an installation's future opportunity to train and test. Proposed state and local investment in regional planning policies and projects can alter the expected patterns of growth and should be evaluated with respect to their short and long-term impacts on the ability to conduct training and testing at the installation. For demonstration purposes, the approach is demonstrated for a multi-county area around Fort Bragg, NC.

15. SUBJECT TERMS

land use planning	Ft. Bragg, NC	GIS
urbanization	modeling	GRASS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	80	**19b. TELEPHONE NUMBER (include area code)**