

GIS Day @ University of Kansas
Nov. 16th, 2011



GeoComputational Intelligence and High-performance Geospatial Computing

Qingfeng (Gene) Guan, Ph.D
Center for Advanced Land Management Information Technologies
School of Natural Resources
University of Nebraska - Lincoln

Contents

1. Computational Science and GeoComputation

2. GeoComputational Intelligence

- ANN-based Urban-CA model

3. High-performance Geospatial Computing

- Parallel Geostatistical Areal Interpolation
- pRPL and pSLEUTH

4. Conclusion

Introduction – Computational Science

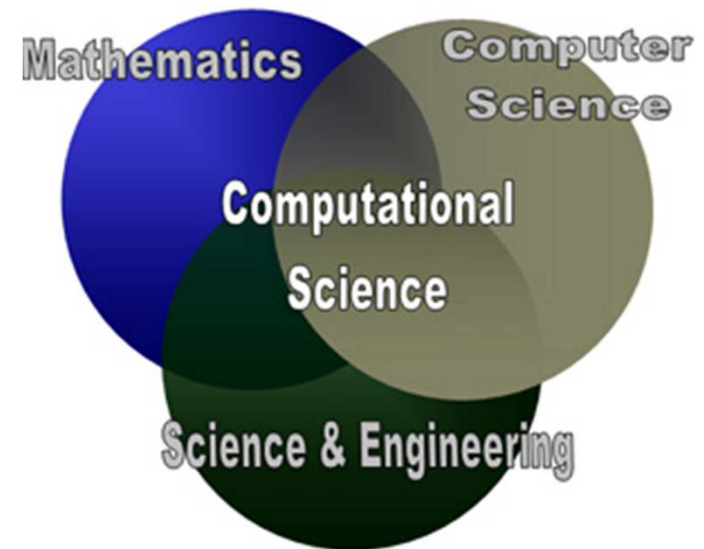
❖ Definition

- *“the field of study concerned with constructing mathematical models and numerical solution techniques and using computers to analyze and solve scientific, social scientific and engineering problems.” (wikipedia)*

❖ Domains include:

- Numerical simulations
- Model fitting and data analysis

❖ Massive computational intensity



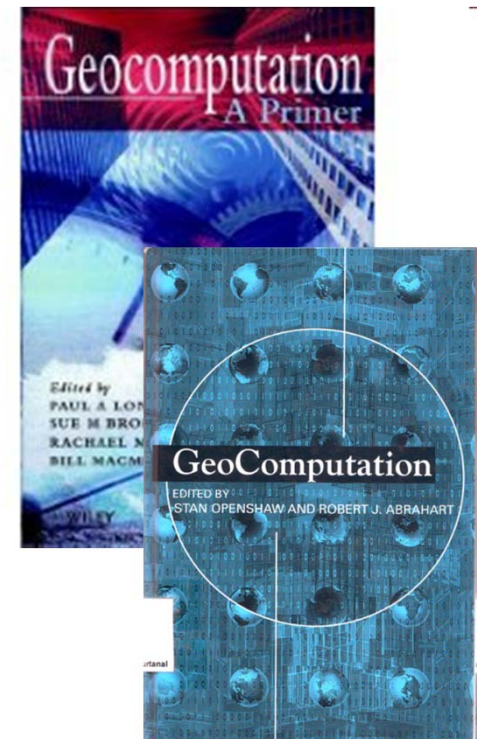
<http://www.it.uu.se/edu/masters/CompSc/>

Introduction - GeoComputation

❖ Definition

- Couclelis (1998) identified the “core GeoComputation” as the innovative (or derived from other disciplines) computer-based geospatial modeling and analysis
 - Contrasted against the traditional computer-supported spatial data analysis and geospatial modeling
- Openshaw (2000) also emphasized Computational Science as the origin of GeoComputation (the Computation part) and the essential concerns about geographical and earth systems (the Geo part)

❖ The capital G and C



Introduction – GeoComputation (cont.)

❖ Methodology

- A wide array of computer-based models and techniques, many of them derived from the field of Artificial Intelligence (AI) and the more recently defined area of Computational Intelligence (CI) (Couclelis, 1998)
 - Expert Systems, Cellular Automata, Neural Networks, Fuzzy Sets, Genetic Algorithms, Fractal Modelling, Visualization and Multimedia, Exploratory Data Analysis and Data Mining, etc.

❖ High-performance geospatial computing



ANN-Urban-CA: an urban growth model

❖ Overview

- Combination of a Cellular Automata (CA) model, an Artificial Neural Network (ANN), and a macro-scale socio-economic model
- Integration of Geography, Natural Resource Science, Social Science, and Economics in a GeoComputation framework



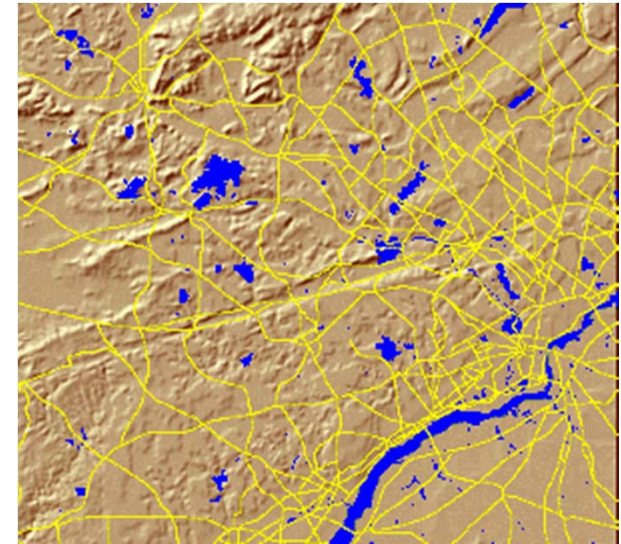
Geospatial Cellular Automata

❖ Bottom-Up structure

- Simple local rules to simulate complex global spatio-temporal dynamics

❖ Widely used in geospatial modeling

- Land-use/Land-cover Change
- Wildfire Propagation
- Flood Spreading
- Freeway Traffic Flow
- More and More Coming up...



Prediction of urban development to the year 2050 over southeastern Pennsylvania and part of Delaware using the SLEUTH model
http://www.essc.psu.edu/~dajr/chester/animation/movie_small.htm

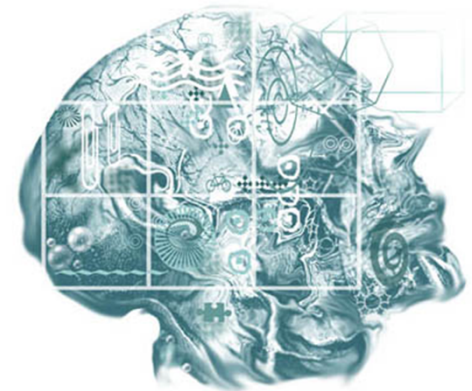
Issues of Geospatial CA

❖ Hard to set proper transition rules and parameters

- How to produce realistic simulations?
- Brute-force calibration
 - Generate results using all possible parameter values
 - Find the “best-match” combination
 - Highly computationally intensive

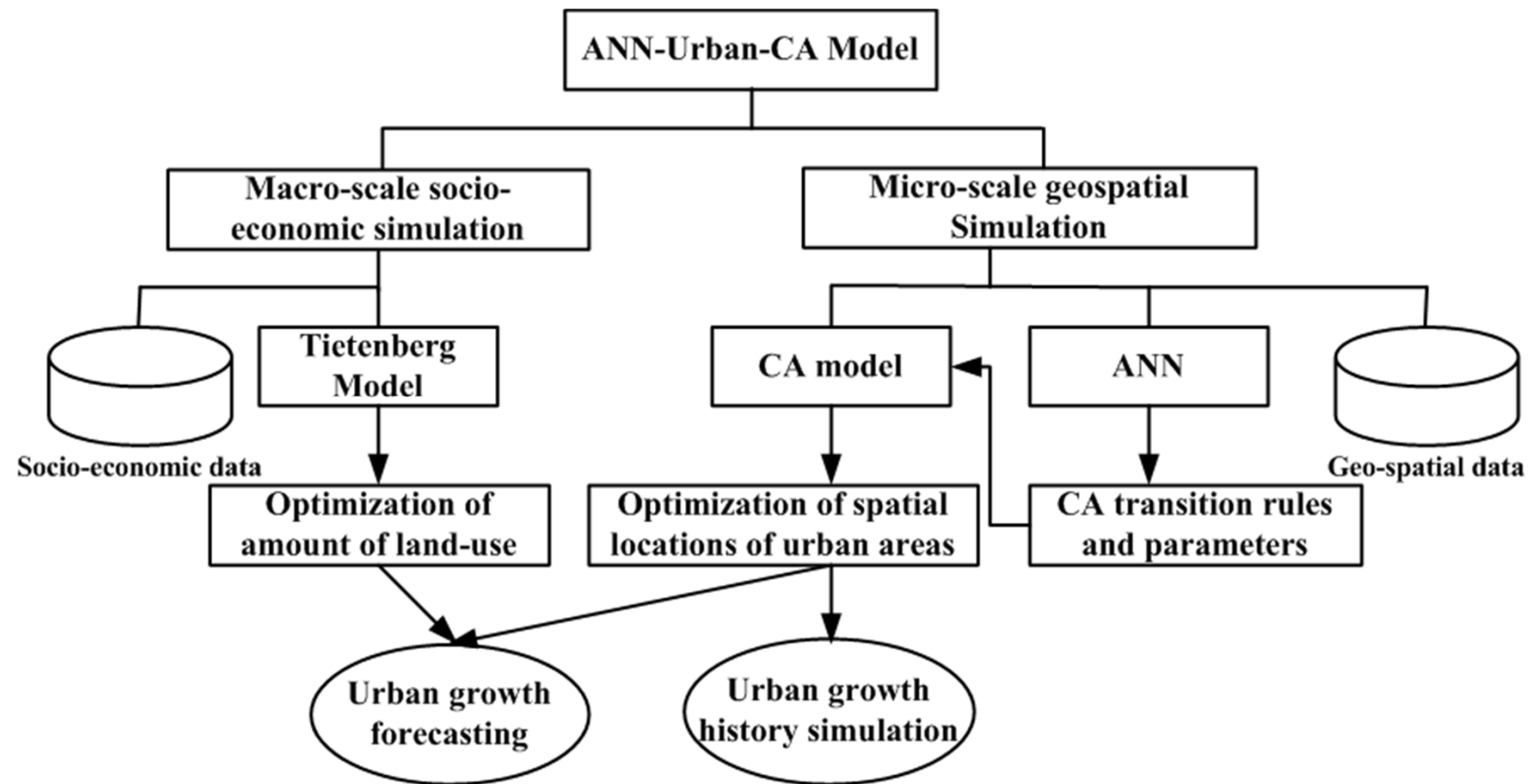
❖ Lack of global control

- Bottom-up structure
- Evolve without constraints



ANN-Urban-CA: Structure

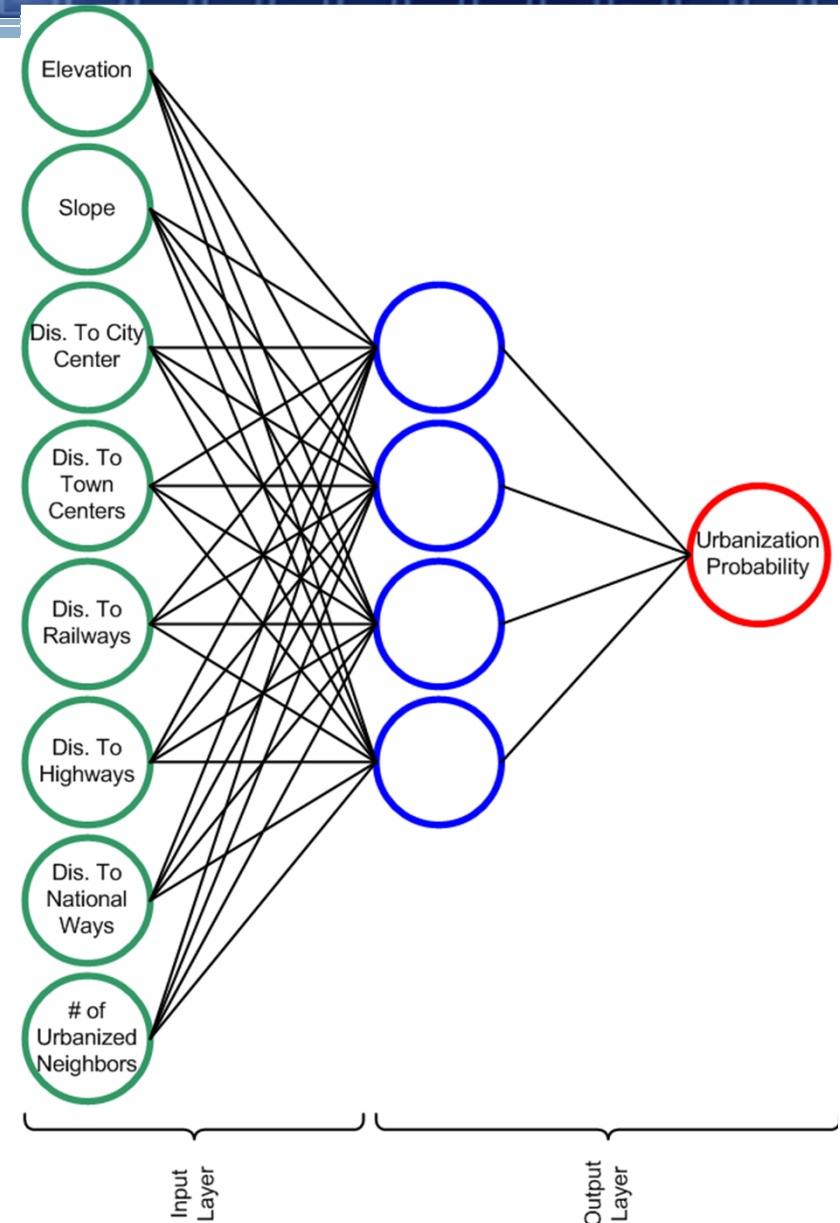
- ❖ **An Artificial-Neural-Network-Based, constrained, Cellular Automata model for urban growth simulation**



ANN-Urban-CA: ANN

❖ Artificial Neural Network

- ANN is suited for dealing with complex nonlinear relationships, e.g., the impacts of driving factors to urban growth
- ANN can learn from available data, and deal with redundancy, inaccuracy, and noise
- Knowledge and experience can be easily learned and stored for further simulation



ANN-Urban-CA: Macro Constrain

❖ Macro-scale Socio-economic model

- The Tietenberg model is used to generate the proper demand for urban space in each period (e. g. year) in the future.
 - A Resource Economic model, which usually is used to solve the problem of *“how to consume resources in the future according to the principle of sustainable development”*
 - Lands are treated as non-regenerative resources, and the urbanization process is treated as land source consumption
 - Population increase as the driving force of land consumption

$$\frac{a - bq_t / P_{ta} - c}{(1+r)^{t-1}} - \lambda = 0$$

$$t = (1, 2, \dots, n)$$

$$Q - \sum_{t=1}^n q_t = 0$$

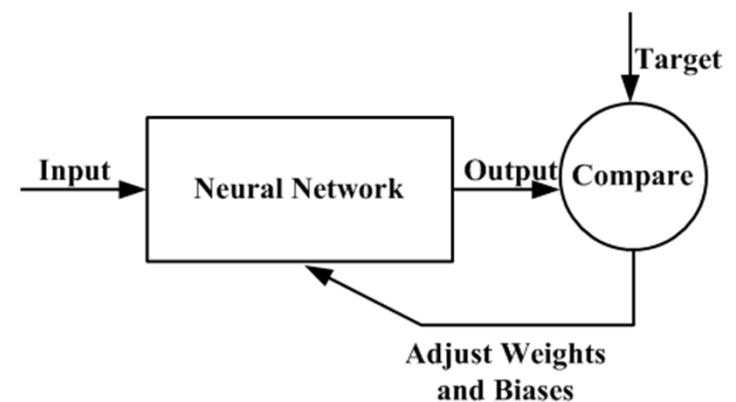
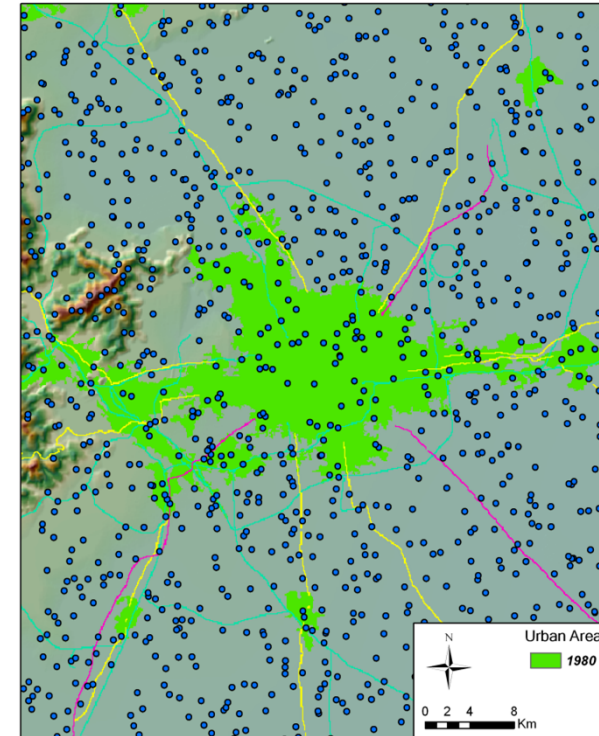
ANN-Urban-CA: Training

❖ Purpose

- ANN adjusts the weight values
- Determine the best-fit transition rules and parameters of the CA

❖ Method

- Back-Propagation (BP) training algorithm
- Input: Driving factors
- Output: Urbanization probability
- Target: Historical urban data



ANN-Urban-CA: Results

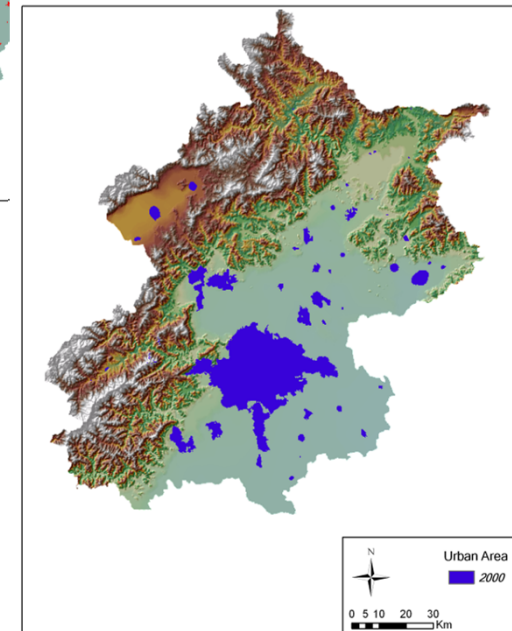
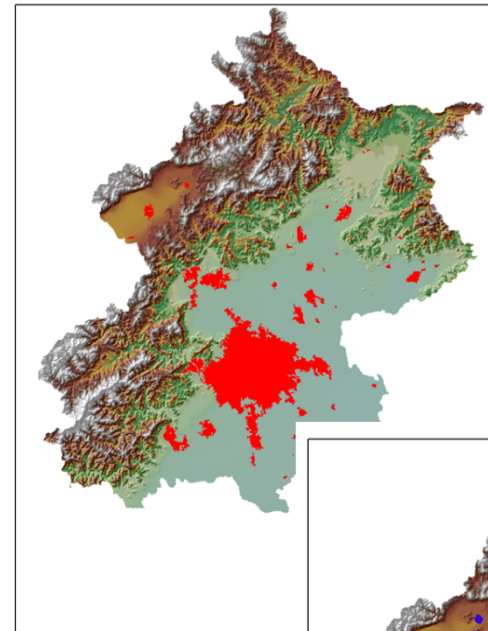
❖ History Simulation

- Trained using samples of Beijing urban maps of 1980, 1995, and 2000
- Simulate urban growth in Beijing 1995 - 2000

$$Lee - Sallee = \frac{A_{real} \cap A_{sim}}{A_{real} \cup A_{sim}} = 0.8318$$

$$correlation = \frac{\sum_k^n c_{ij}}{\sqrt{\sum_k^n (z_i - \bar{z}_i)^2} \times \sqrt{\sum_k^n (z_j - \bar{z}_j)^2}} = 0.9018$$

Real Beijing urban, 2000

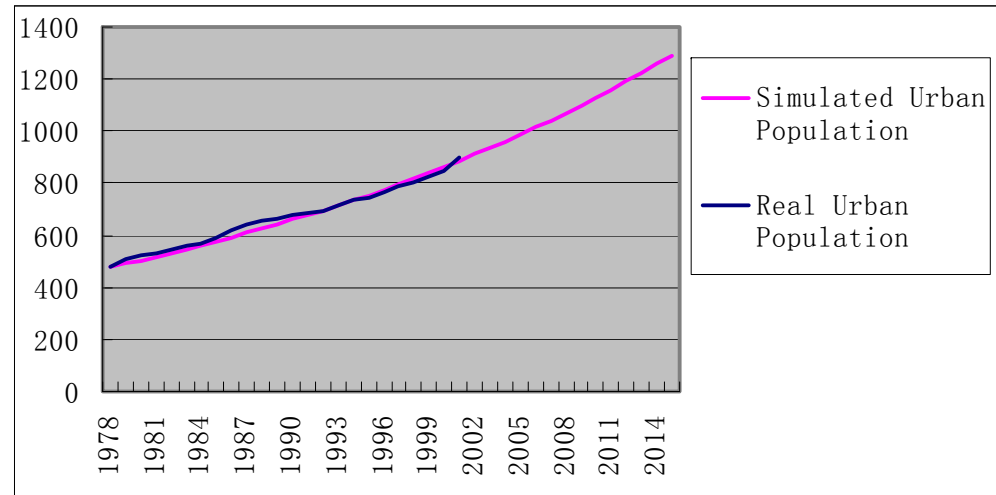


Simulated Beijing urban, 2000

ANN-Urban-CA: Results

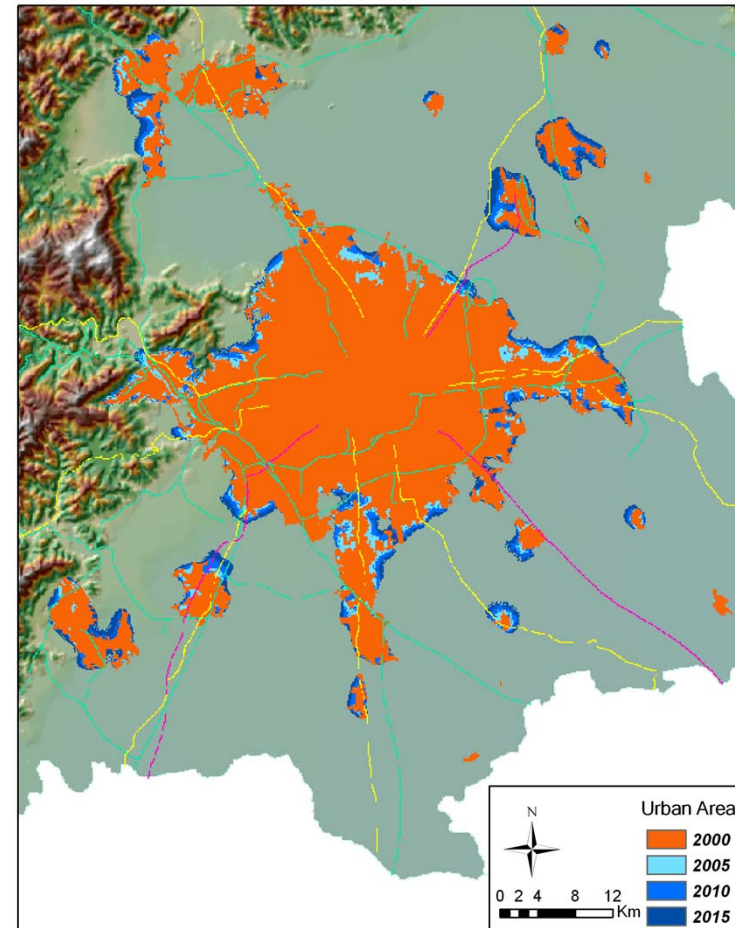
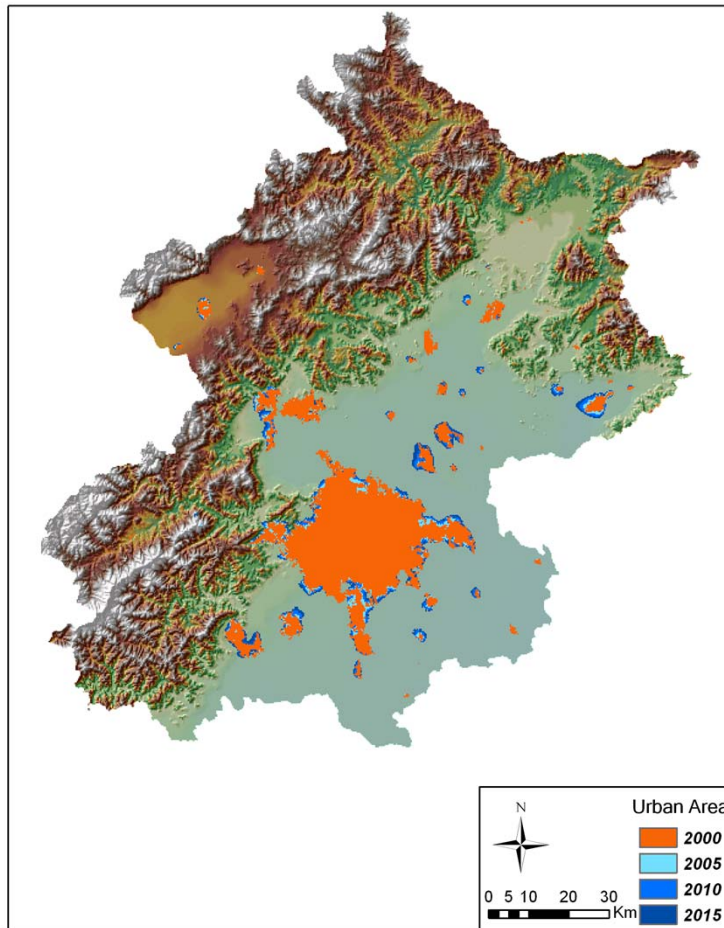
❖ Future Forecast

- Increased populations of Beijing 2001- 2015
- By using the Tietenberg Model, 6 scenarios of urbanization were derived



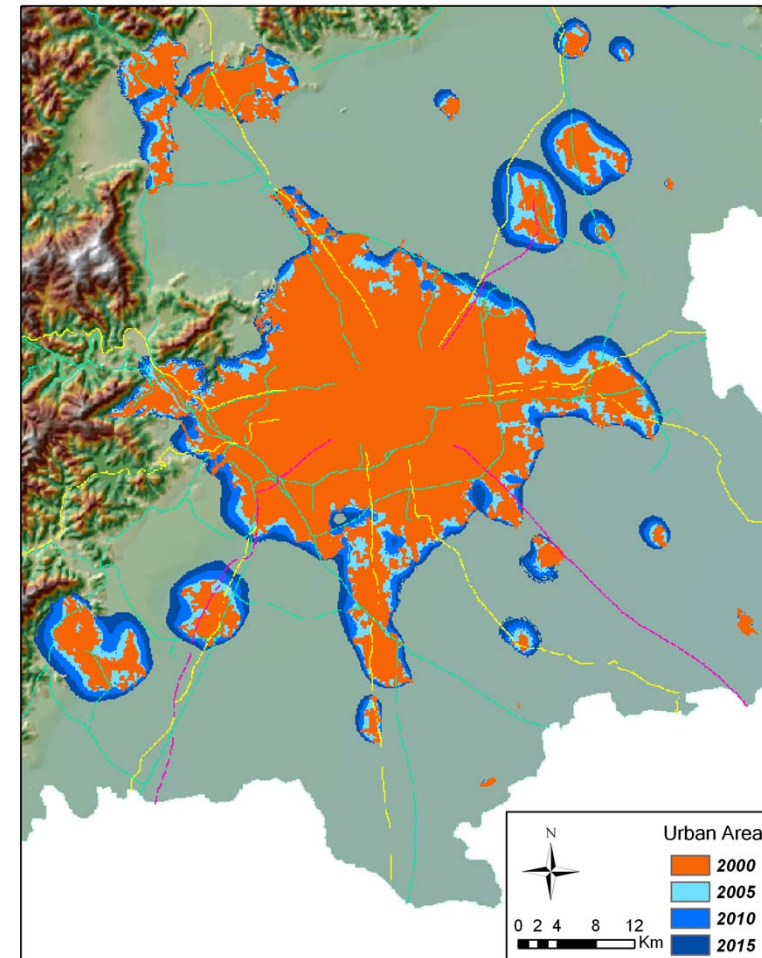
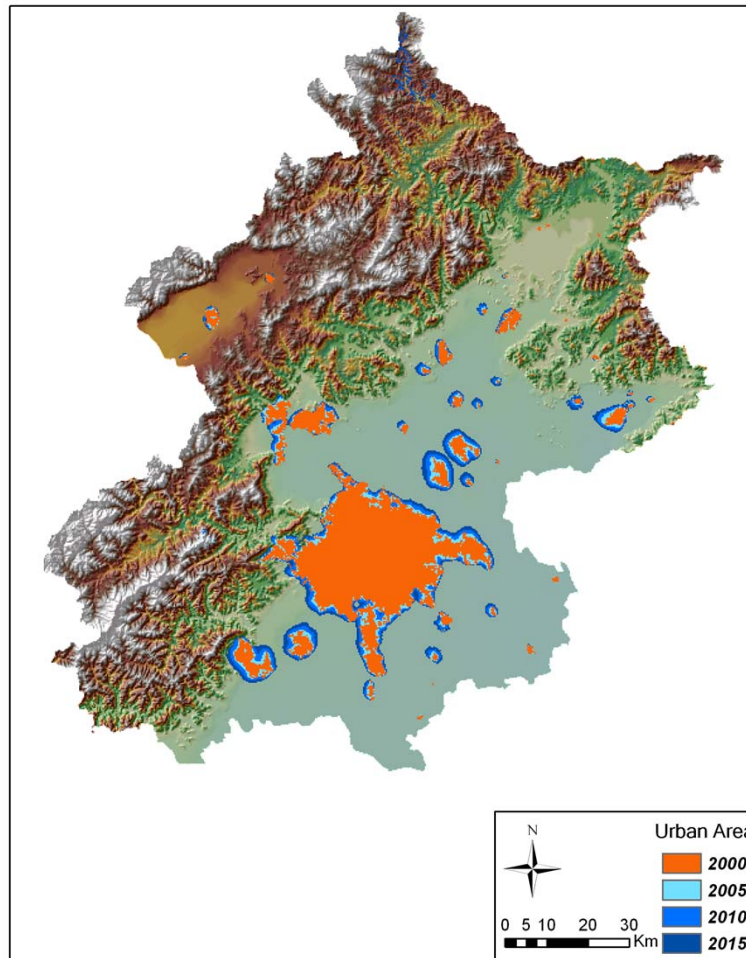
Years	Increased Pop (10,000)	Total Scenario 1 (hm ²)		Total Scenario 2 (hm ²)		Total Scenario 3 (hm ²)	
		r=0	r=0.01	r=0	r=0.01	r=0	r=0.01
2001~2005	123.956	7522.9	8201.1	15186.8	15781.6	17741.5	18308.5
2006~2010	141.766	8687.9	8758.7	17538.6	17600.8	20488.8	20548.1
2011~2015	162.134	10033.2	9284.2	20254.6	19597.6	23661.7	23035.4

ANN-Urban-CA: Results



Urban Growth in Beijing 2000 – 2015 (Scenarios 1)

ANN-Urban-CA: Results



Urban Growth in Beijing 2000 – 2015 (Scenario 4)

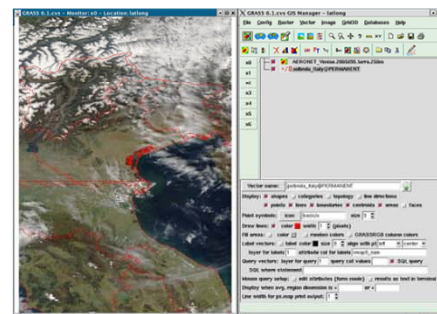
Sub-Conclusion on ANN-Urban-CA

- ❖ **ANN's capability of dealing with nonlinear complex systems**
 - Calibrated without heavy computing overhead and subjective human interference
- ❖ **Optimal quantity allocation + optimal spatial allocation**
 - Providing an ideal pattern of sustainable urban development, useful in urban planning
- ❖ **Highly flexible structure and modeling approach**
 - Easily generalized to model other kinds of spatio-temporal dynamics for various purposes, e.g., spread of invasive species and vegetative epidemics, movement of toxic pollutants in water systems, and land-cover change caused by climate change
 - Open to any possible/available datasets, e.g., numerous remotely sensed data and other natural resource and environmental data

High-performance Geospatial Computing

❖ Why high-performance computing?

- GeoComputation implies HPC
- Increasing demand for computational power in geospatial research and applications
 - Sophisticated and complicated analytical algorithms and simulation models
 - High-resolution and large-volume datasets
 - Rapid processing and real-time response



Integrated E&P GIS Data

O&G Well Permits

O&G Well Headers, IP/Test Completion and Production

O&G Field Outlines

O&G Pipelines and Facilities

BLM, MMS & State Leases, Units and Agreements

Land Ownership - Private, State and Federal

Culture - Boundaries, Water and Towns

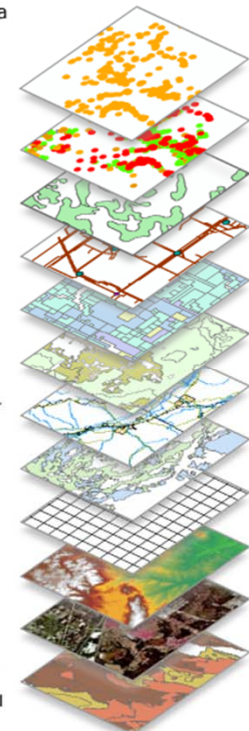
Features

Grids

Models

to

Geology, Seismicity, and Thermal



High-performance Computing

❖ Definition

- Usually refers to parallel computing
 - The use of multiple computing units (e.g., computers, processors/CPU cores, or processes) working together on a common task in a concurrent manner in order to achieve higher performance
 - In contrast to sequential computing that usually has only one computing unit
- Performance is usually measured with computing time



❖ Emerging Cyberinfrastructure

- Grid Computing
- Cloud Computing

A massive parallel computing system
(<http://ctbp.ucsd.edu/pc/html/intro4.html>)

Areal Interpolation – Introduction

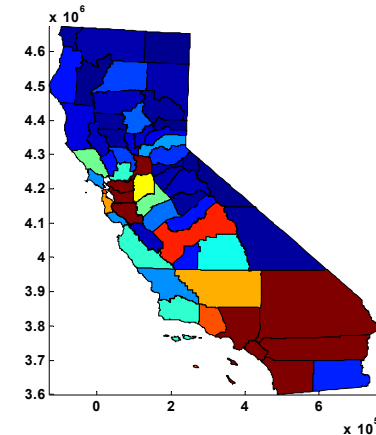
❖ Definition

- Predicts the unknown (target) attribute values at the required partition (target zones or supports) from a set of known (source) attribute data available on a different partition (source zones or supports)

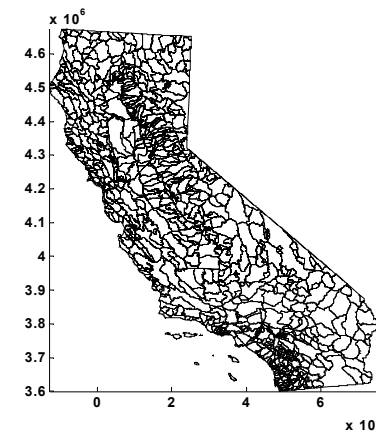
❖ Two main approaches

- Cartographical methods
 - Use cartographical properties of supports, e.g., area, as the basis
 - Simple and widely used
- Geostatistical methods
 - Use variants of Kriging
 - Accounts for spatial autocorrelation
 - Measure the reliability of prediction
 - Mass-preserving target prediction

Population of counties



Population of watersheds = ?

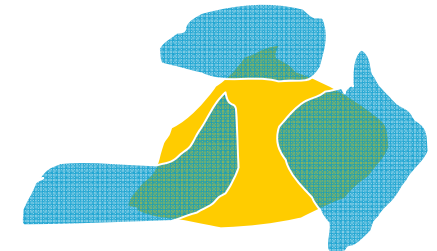
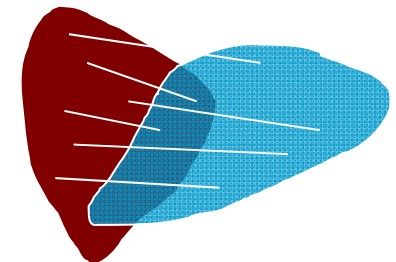
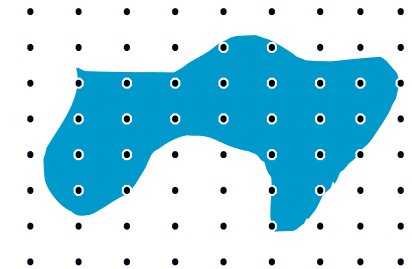


An areal interpolation problem

Geostatistical Areal Interpolation

❖ Steps

- Discretization of source and target supports with a regular raster (point values not known, just location)
- Computation of support-to-support covariances as integrals from a given point covariance model
 - Between all source supports
 - Between all source and target supports
- Use of Kriging system with computed covariances to derive weights for interpolation
- Interpolated values computed as linear combinations of the Kriging weights and the source data



FFT-based Areal Interpolation

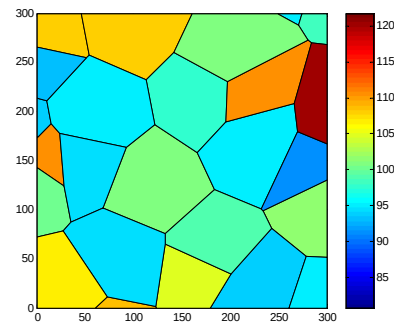
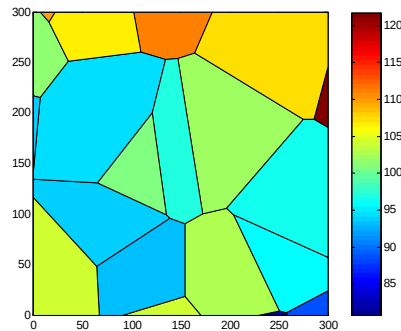
❖ Traditional geostatistical interpolation

- Highly computationally intensive
 - Massive memory space
 - Long computing time

❖ FFT-based spectral method

- Use Fast Fourier Transform (FFT) to compute support-to-support covariances

$$\sigma_z(s_k, s_{k'}) = \text{Cov} \{Z(s_k), Z(s_{k'})\} = \text{FFT}(\overline{g_k}) \otimes \text{FFT}(\overline{C(1,:)}) \otimes \text{FFT}(\overline{g_{k'}})$$



Execution times (for a 360x360 discretization grid)

Traditional method: ~4,000 sec

FFT-based Spectral method: ~50 sec

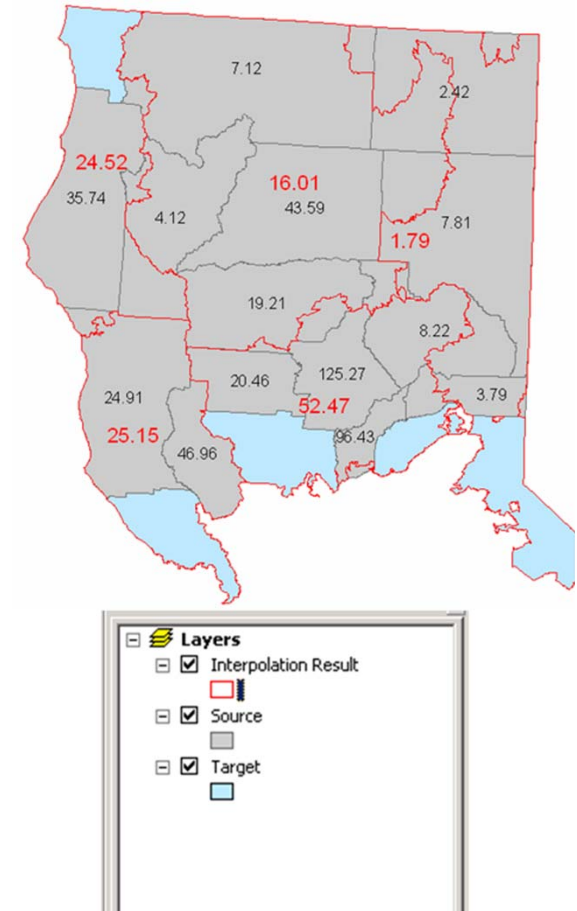
FFT-based Areal Interpolation

❖ FFT-based method is **STILL** computationally intensive

- When it comes to real-world applications
 - Population density from counties to 3-digit zipcode regions in Northern California
 - 500X500 discretization grid
 - Matlab program
 - Penium4 3.2GHz PC with 2GB RAM
 - 900 seconds

❖ **Solution**

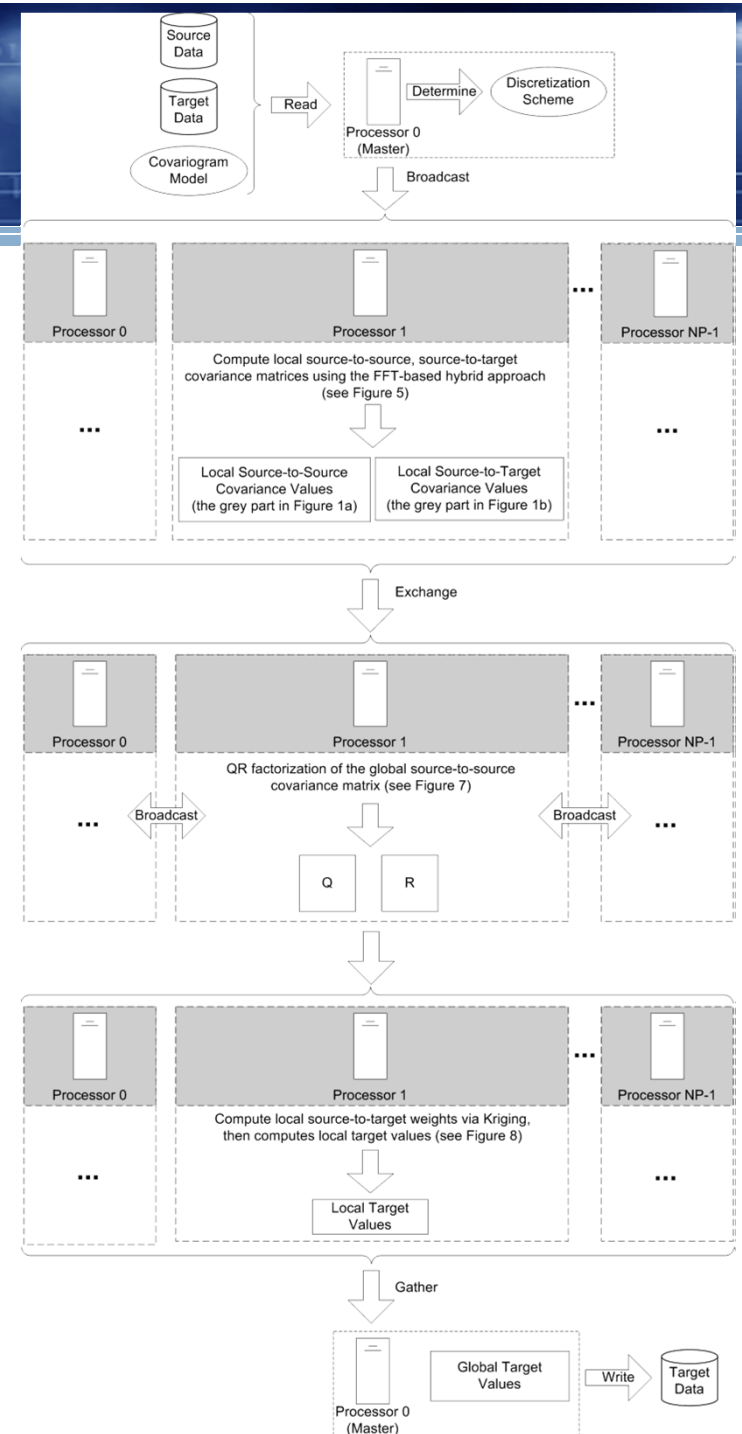
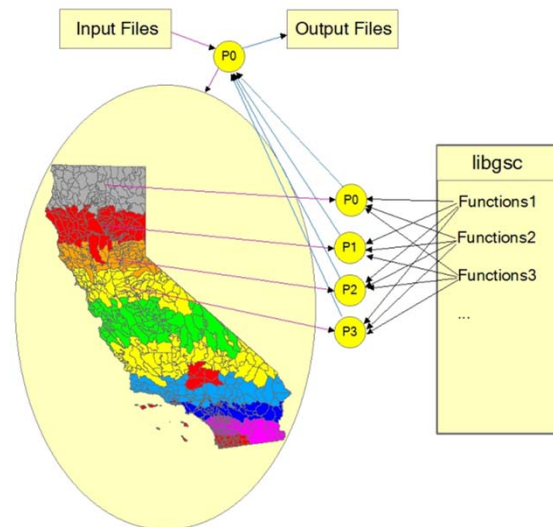
- Parallel computing
 - The computation of covariance between a pair of supports is independent from that of other pairs
 - The computation of prediction for a target support is independent from that for other target supports



pAI: Algorithm Overview

❖ Three parallel processes

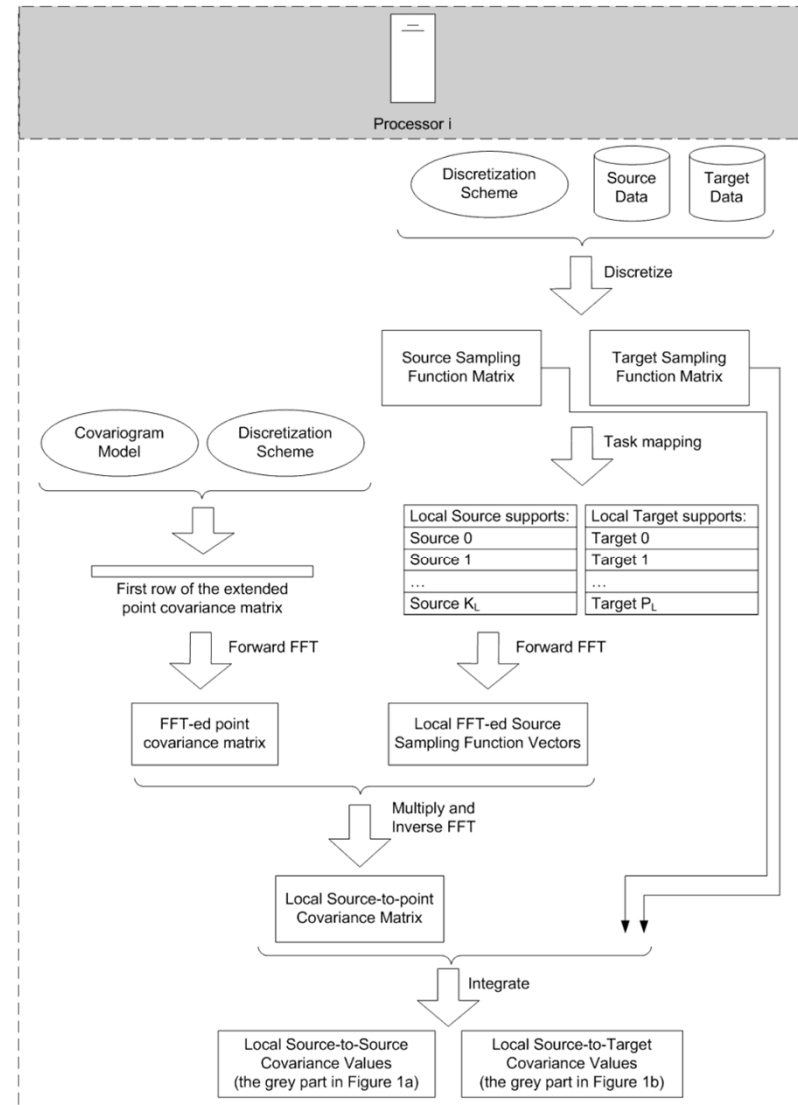
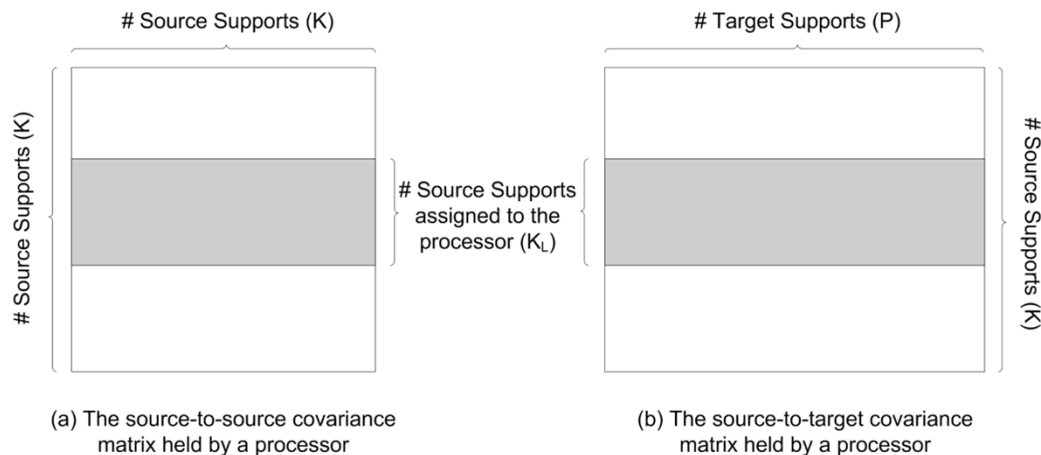
- Source-to-source, and source-to-target covariance matrices by means of FFT
- QR factorization of the source-to-source covariance matrix
- Source-to-target weights via Kriging, and predicted attribute values for target supports



pAI: 1st Parallel Process

❖ Support-to-support covariance

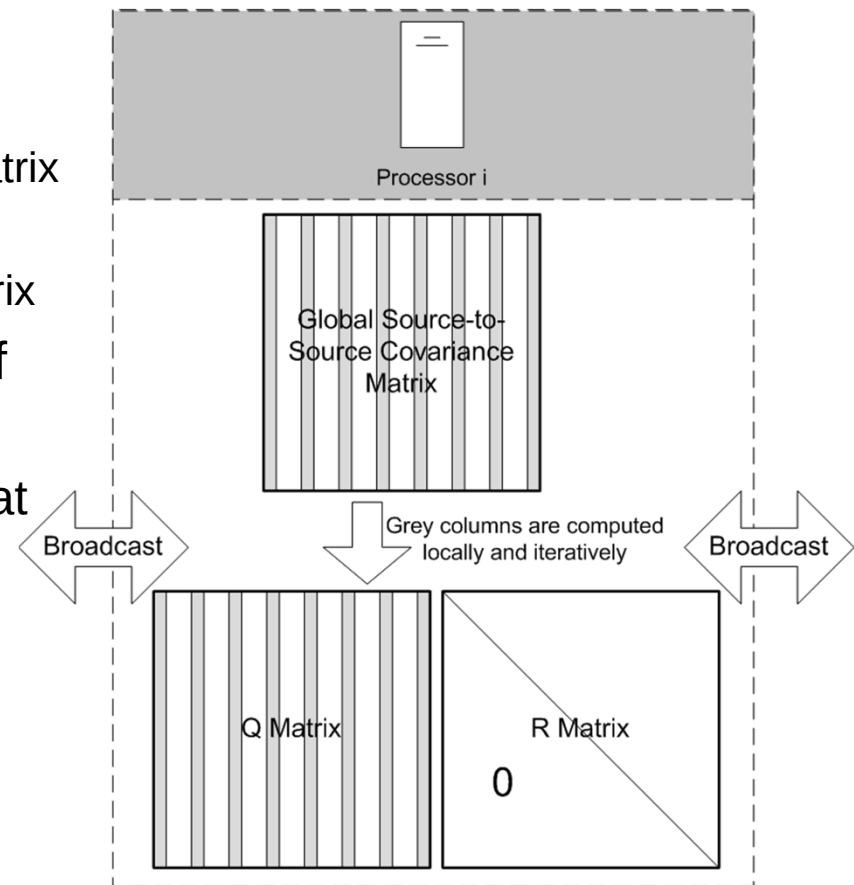
- Parallel over source supports
 - Each processor handles a subset of source supports



pAI: 2nd Parallel Process

❖ QR factorization

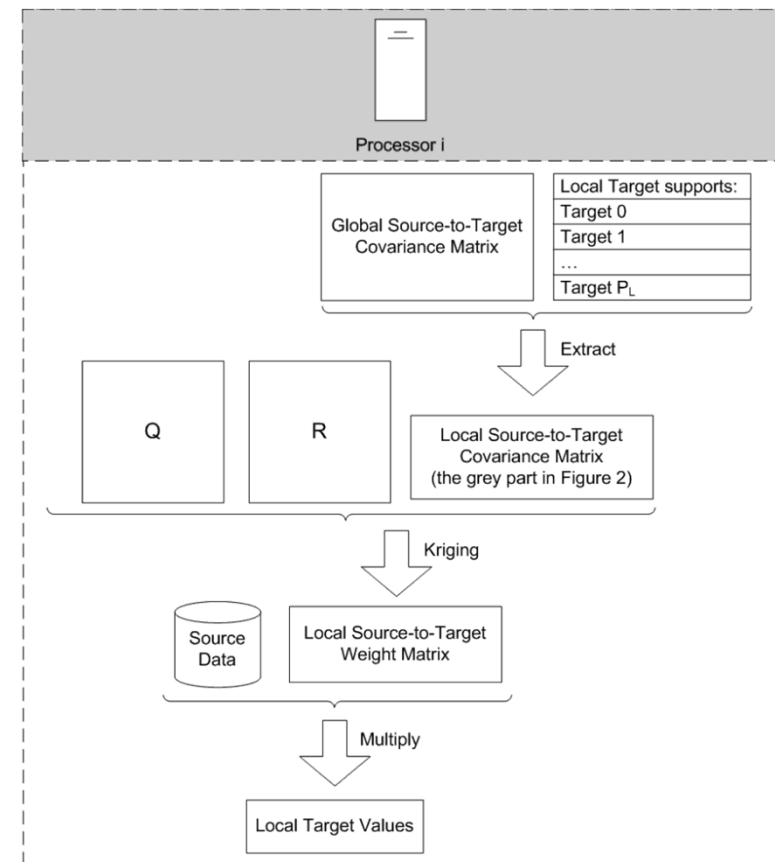
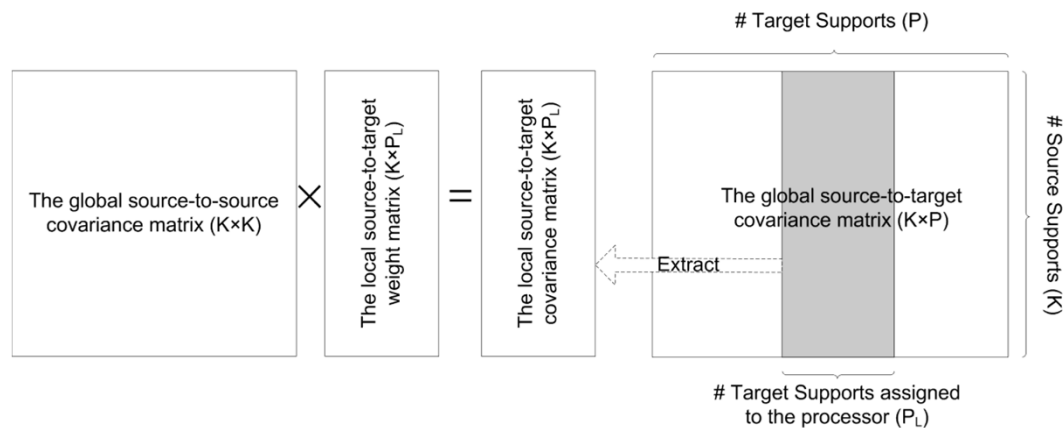
- Needed to solve a $Ax=b$ problem
 - In a Kriging system
 - A: source-to-source covariance matrix
 - x: source-to-target weight matrix
 - b: source-to-target covariance matrix
- Each processor handles a subset of columns in the Q matrix
 - Data exchange among processors at each iteration
 - Non-blocking communication technique
 - Overlap computation and data exchange



pAI: 3rd Parallel Process

❖ Source-to-target weights and target predictions

- Parallel over target supports
 - Each processor handles a subset of target supports



pAI: Implementation

❖ Stand-alone program

- Written in C++
- Based on Message Passing Interface (MPI)
- Utilizes public-domain libraries
 - FFTW (www.fftw.org)
 - GsTL(<http://pangea.stanford.edu/~nremy/GTL/>)
 - Shapefile C Library (<http://shapelib.maptools.org/>) for data I/O
 - Template Numerical Toolkit (TNT, <http://math.nist.gov/tnt/index.html>)

❖ User-specified options

- Shapefiles of source and target supports
- Discretization density (point spacing distance)
- Covariogram model
- Task mapping scheme
- Simple Kriging or Ordinary Kriging

pAI: Experiment Settings

❖ Two datasets

- Eastern Time Zone dataset
 - Source: population densities of counties in 2000 (2248 polygons)
 - Target: population densities of watersheds (1633 polygons)
- Continental U.S. dataset
 - Source: population densities of counties in 2000 (4703 polygons)
 - Target: population densities of watersheds (2848 polygons)

❖ Discretization scheme

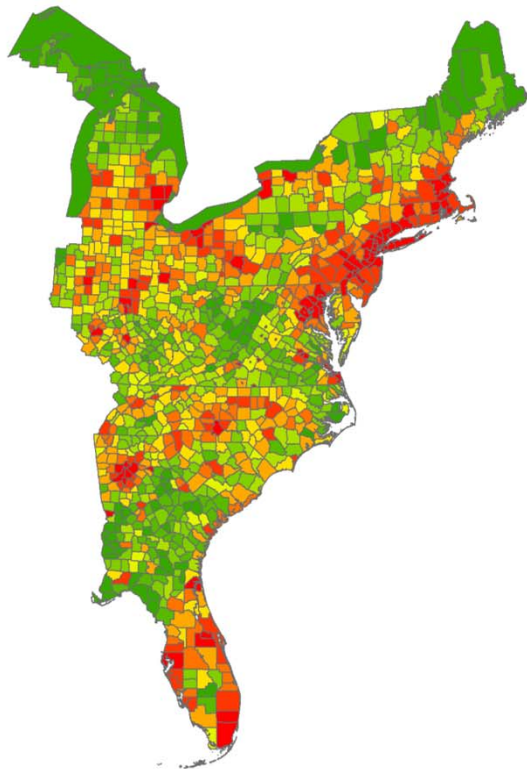
- 2000-meter point spacing
- Eastern Time Zone – 1333X917 grid (1.2 million points)
- Continental U.S. - 1452X2348 grid (3.4 million points)

❖ Computer cluster

- 280 AMD quad core nodes (2.2 GHz, 8 GB RAM per node)
- 871 Opteron two-dual-core nodes (2.8 GHz, 8GB RAM per node)
- 800 MB/second InfiniBand

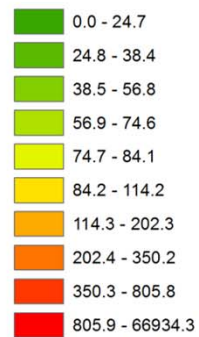
pAI: Results

Population Densities of Counties

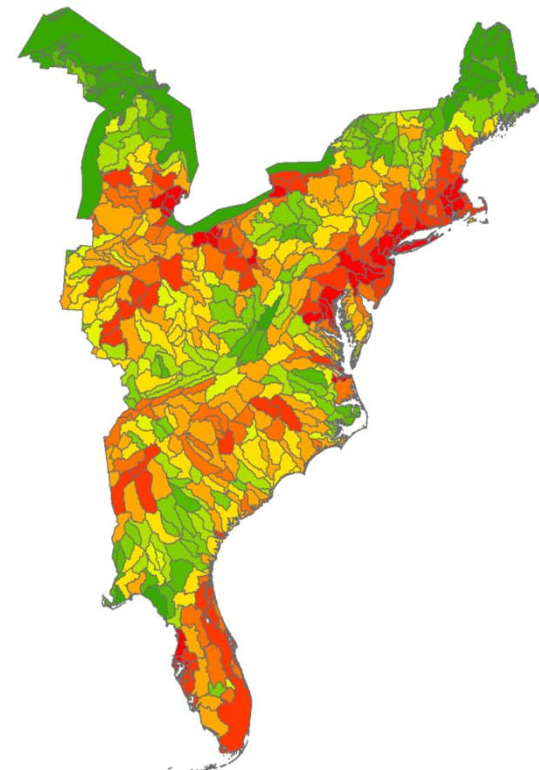


Eastern Time Zone

Population Density
(People/SquareMile)



Population Densities of Watersheds



0 400,000 800,000 1,600,000 Meters

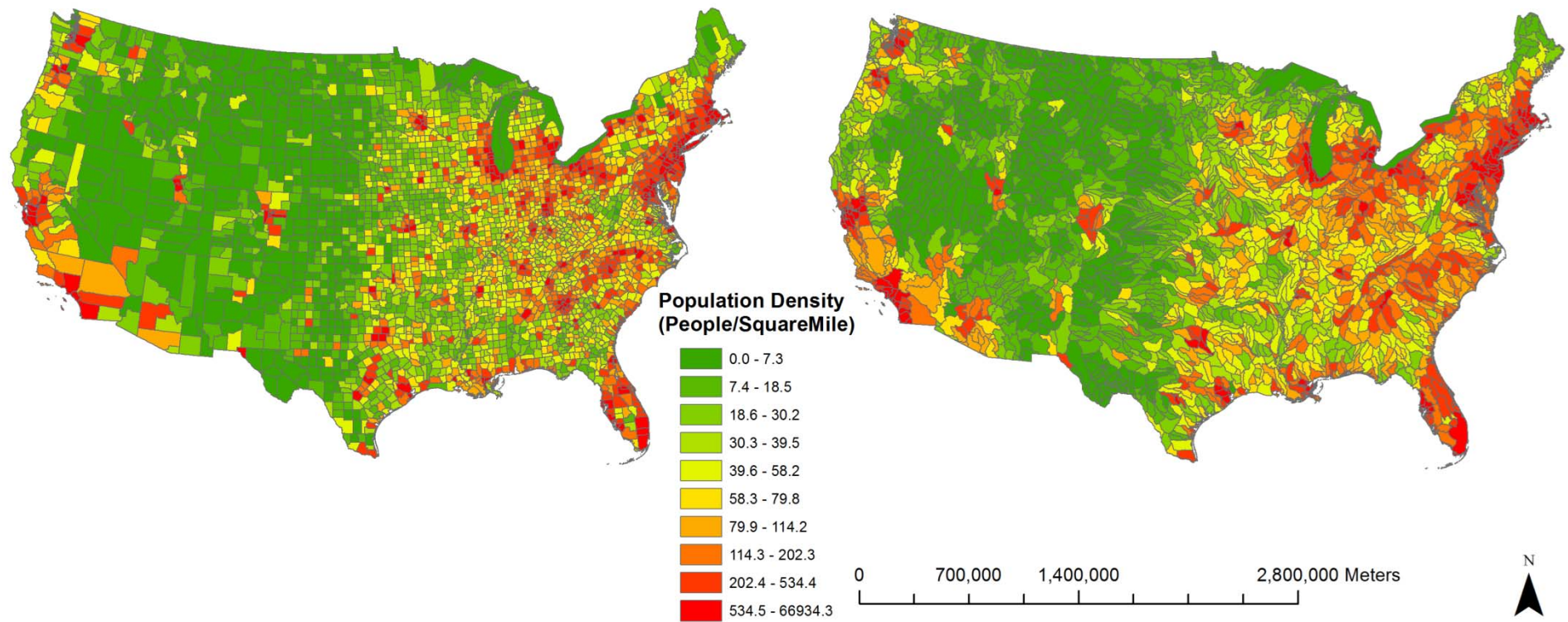


pAI: Results

Population Densities of Counties

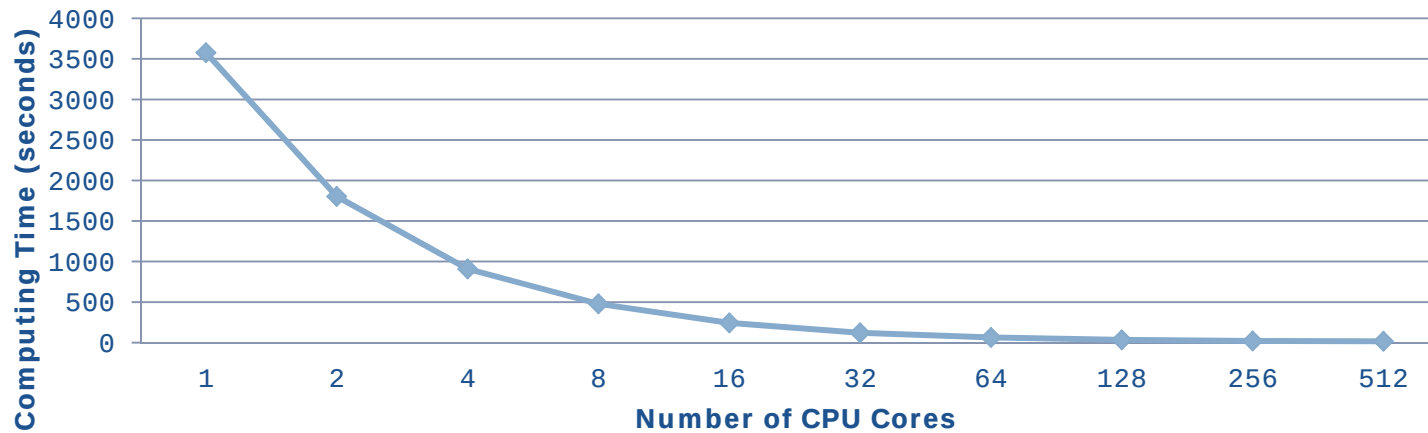
Continental U.S.

Population Densities of Watersheds

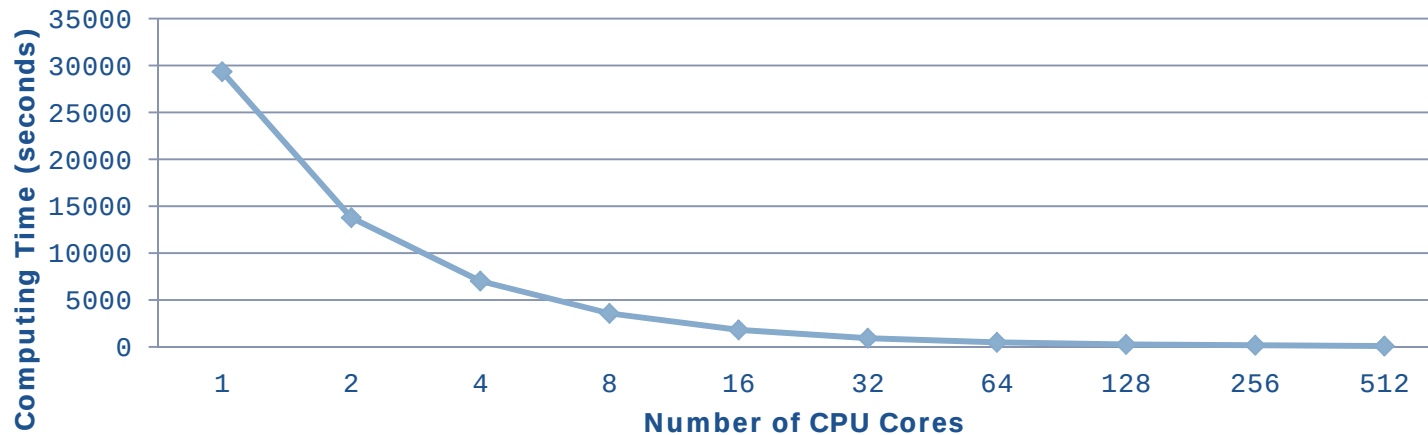


pAI: Computing time

(a) Eastern Time Zone Dataset



(b) Continental U.S. Dataset



Sub-Conclusion on pAI

❖ This parallel algorithm

- Drastically reduced the computing time
- Achieved fairly high speed-ups and efficiencies
- Scaled reasonably well as the number of processors increased and as the problem size increased

❖ Based on global Kriging

- All source supports are used for the prediction for each target
- Can be used for local Kriging
 - Neighbor search

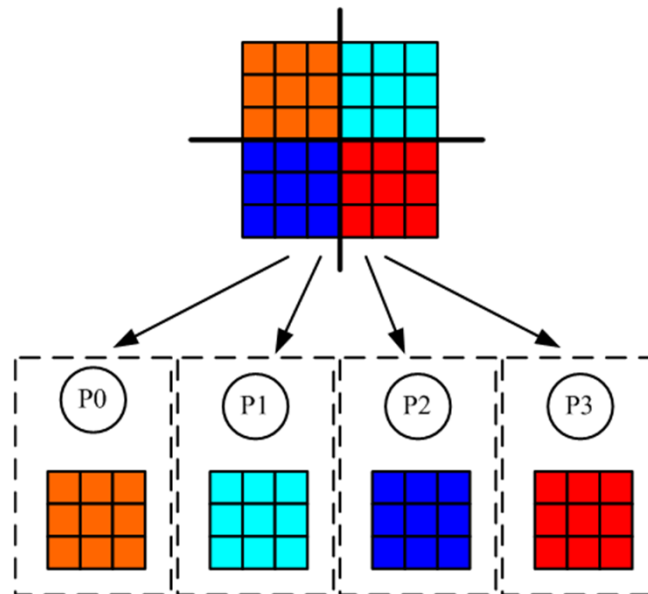
❖ Regular discretization grids

- FFT-based technology requires regular grids
- If irregular discretization is to be used, computational complexity for support-to-support covariance is not uniform anymore
 - Adaptive task mapping methods will help

pRPL: parallel Raster Processing Library

❖ Raster is born to be parallelized

- A raster dataset essentially is a matrix of values, each of which represents the attribute of the corresponding cell of the field
- A matrix can be easily partitioned into sub-matrices and assigned onto multiple processors so that the sub-matrices can be processed simultaneously



pRPL: Introduction

- ❖ **An open-source general-purpose parallel Raster Processing programming Library**
- ❖ **Encapsulates complex parallel computing utilities and routines specifically for raster processing**
 - Enables the implementation of parallel raster-processing algorithms with minimal knowledge of parallel computing and programming
 - Greatly reduces the development complexity
- ❖ **Possible usage**
 - Massive-volume geographic raster processing
 - Image (including remote sensing imagery) processing
 - Large-scale Cellular Automata (CA) and Agent-based Modeling
- ❖ **Free downloadable and open source**
 - <http://sourceforge.net/projects/prpl/>



pRPL: Features

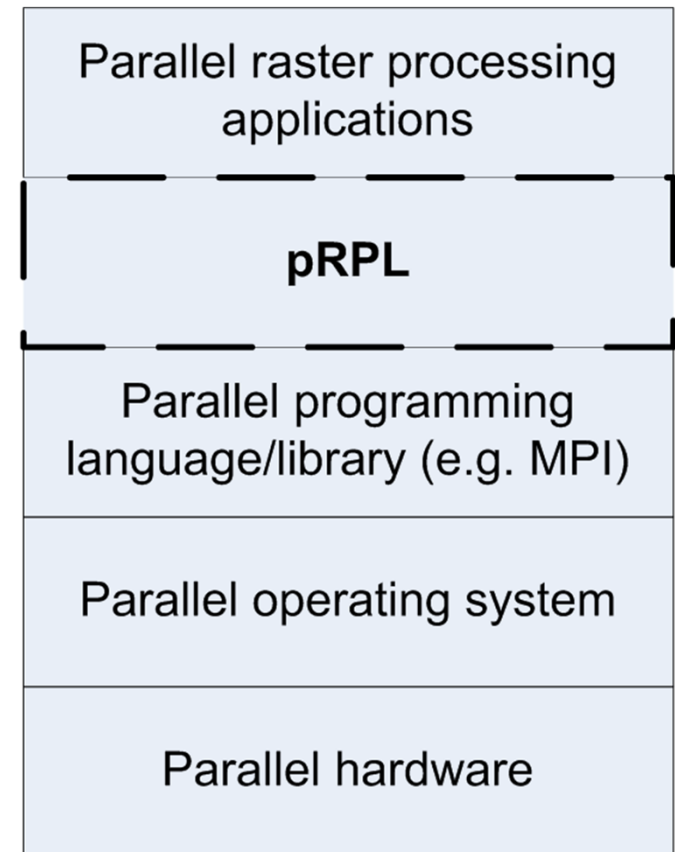
❖ Object-Oriented programming style

- Written in C++
- Built upon the Message Passing Interface (MPI)

❖ Class templates support arbitrary data types

- e.g. integer, char, double precision floating point number, even user-defined types

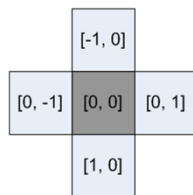
❖ Transparent Parallelism



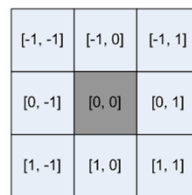
pRPL: Features (cont.)

❖ Spatially Flexible

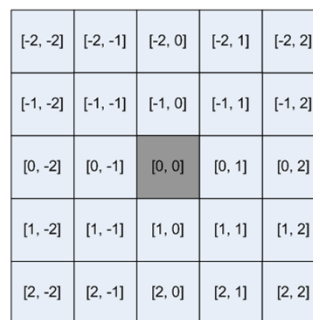
- Supports any arbitrary neighborhood configuration
- Supports centralized and non-centralized algorithms



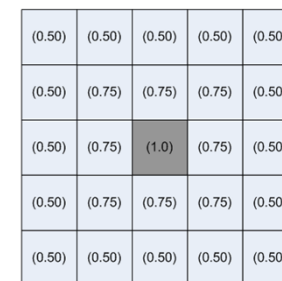
(a) Von Neumann Neighborhood



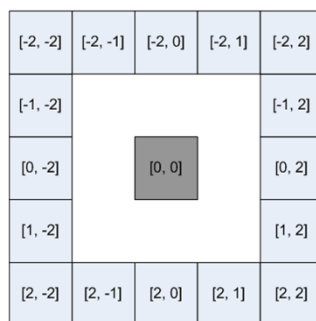
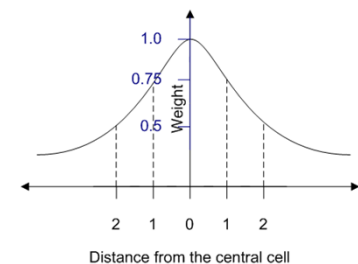
(b) Moore Neighborhood



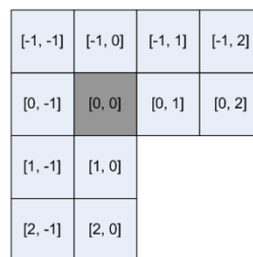
(c) Extended Moore Neighborhood



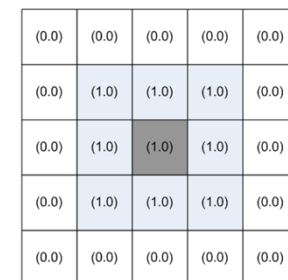
(a) A weighting scheme for kernel algorithms



(d) A discontinuous Neighborhood



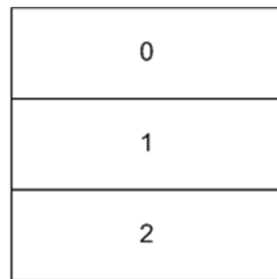
(e) An Asymmetric Neighborhood



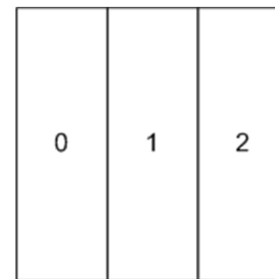
(b) Different weighting schemes for virtually different spatial configurations

pRPL: Features (cont.)

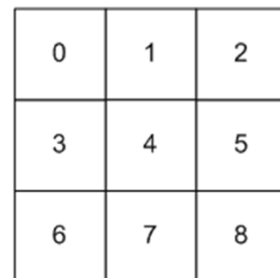
❖ Regular and irregular decomposition



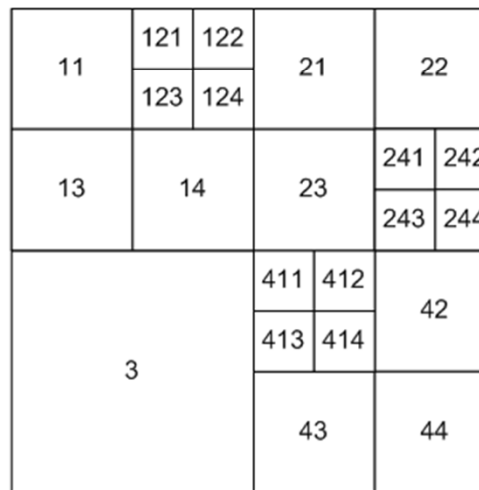
(a) Row-wise decomposition



(b) Column-wise decomposition



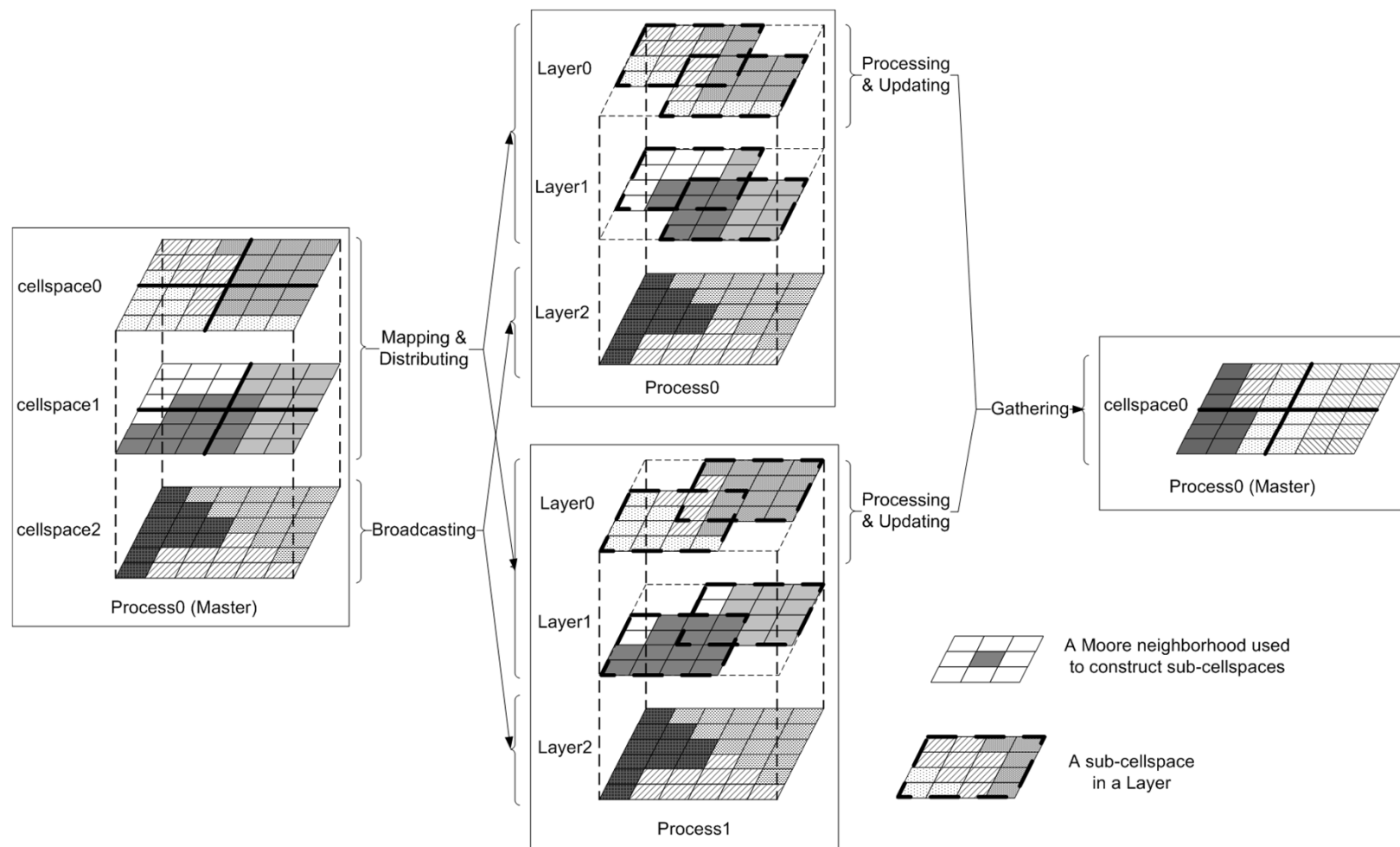
(c) Block-wise decomposition



(d) Quad-tree decomposition

pRPL: Features (cont.)

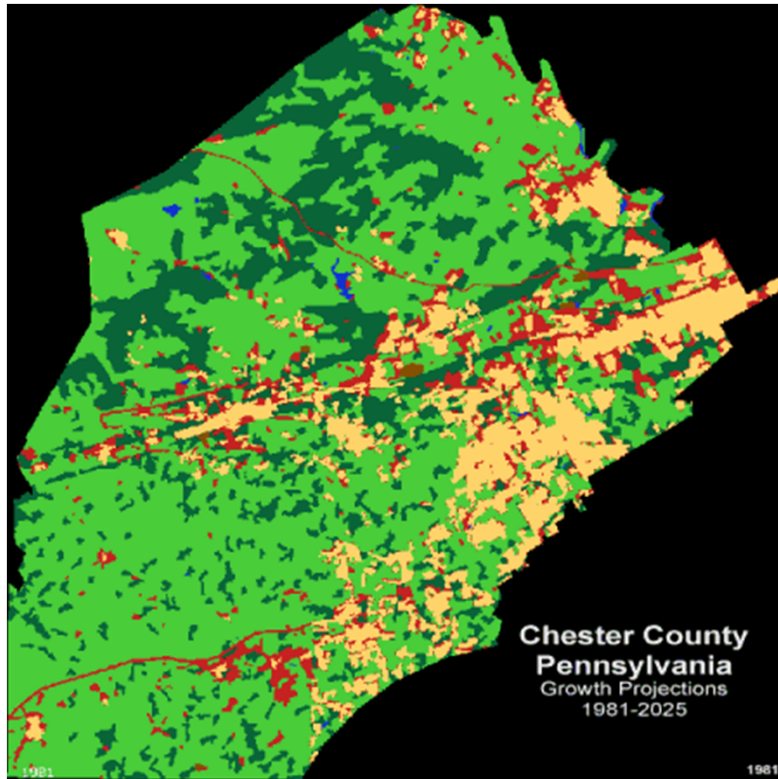
❖ Multi-layer processing



pRPL: Programming

```
int main(int argc, char *argv[]) {  
    PRProcess myPrc(MPI_COMM_WORLD); /* Raster processor */  
    myPrc.init(argc, argv);  
  
    Layer<int> lyr2update(myPrc, "Int_Layer");  
    ... /* load the data on the master processor */  
  
    lyr2update.smplDcmpDstrbt(SMPL_ROW, 2*myPrc.nPrCs()); /* Decompose  
distribute */  
    MyTransition myTrans; /* Customized Transition object */  
    lyr2update.update(myTrans) ; /* Apply the Transition on the Layer */  
    lyr2update.gatherCellSpace(); /* Gather the whole cellspace and store it  
on the master processor */  
  
    myPrc.finalize();  
    return 0;  
}
```


pSLEUTH: Introduction



Urban growth in Chester County, Pennsylvania,
1981 – 2025
http://mcmcweb.er.usgs.gov/de_river_basin/phil/modeling.html

❖ SLEUTH model

- Uses a **modified CA** to model the spread of urbanization across a landscape (Clarke et al., 1996, 1997)
- Its name comes from the GIS data layers that are incorporated into the model: **S**lope, **L**anduse, **E**xclusion layer, **U**rban, **T**ransportation, and **H**illshade

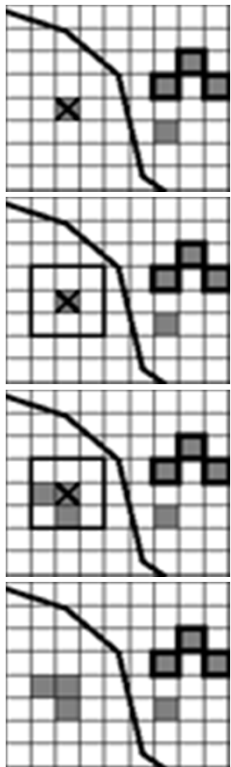
pSLEUTH: Introduction (cont.)

❖ Coefficients

- Dispersion
- Breed
- Spread
- Slope
- Road Gravity

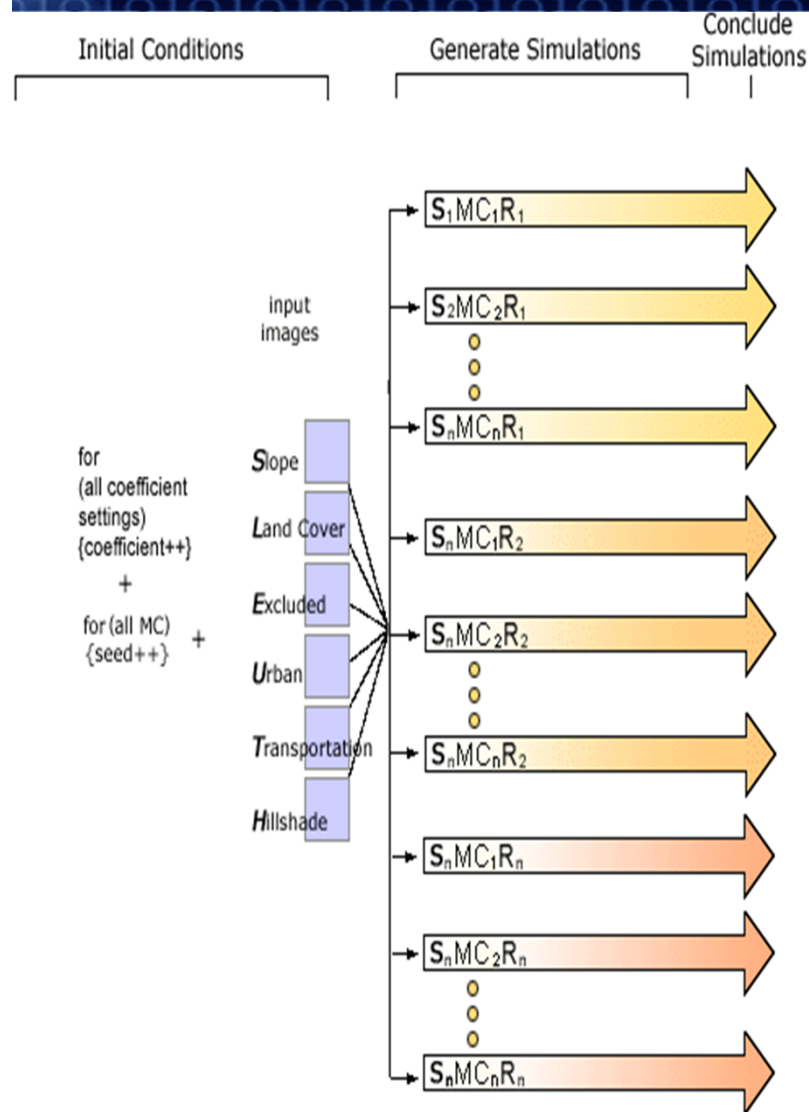
❖ Rules

- Spontaneous Growth Rule (centralized)
- New Spreading Centers Rule (non-centralized)
- Edge Growth Rule (non-centralized)
- Road-Influenced Growth Rule (non-centralized)



→ For more info. about SLEUTH: <http://www.ncgia.ucsb.edu/projects/gig/>

pSLEUTH: Introduction (cont.)



❖ Calibration

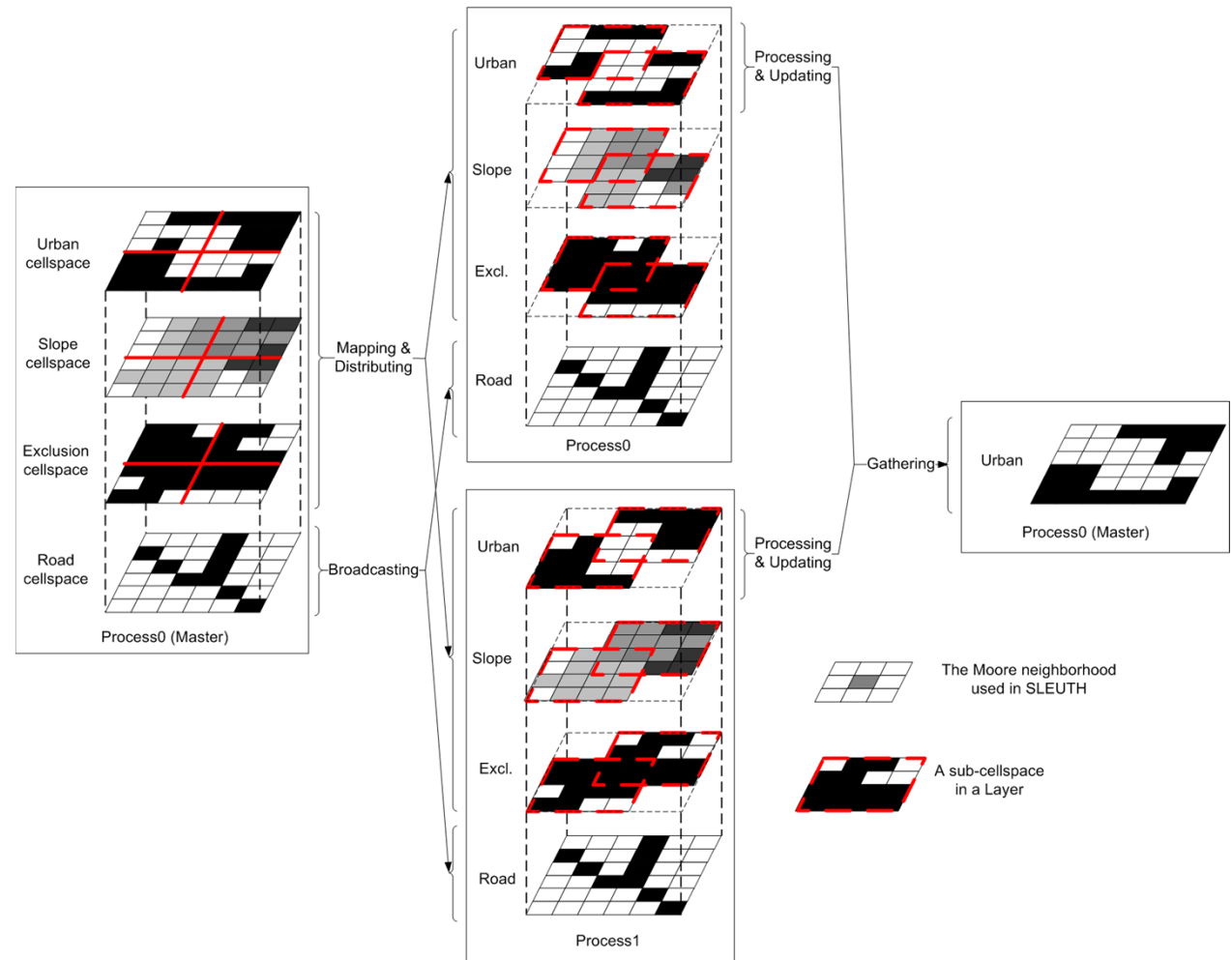
- Determine best parameter values to produce realistic results
- Brute-force calibration
 - Produce results using all possible combinations of parameter values
 - Compare with historical data to determine the best-match combination
 - A large number of combinations of parameters (10^{15} coefficient sets) → Extremely intensive computing overhead

Calibration of SLUETH

<http://www.ncgia.ucsb.edu/projects/gig/>

pSLEUTH: Parallelization

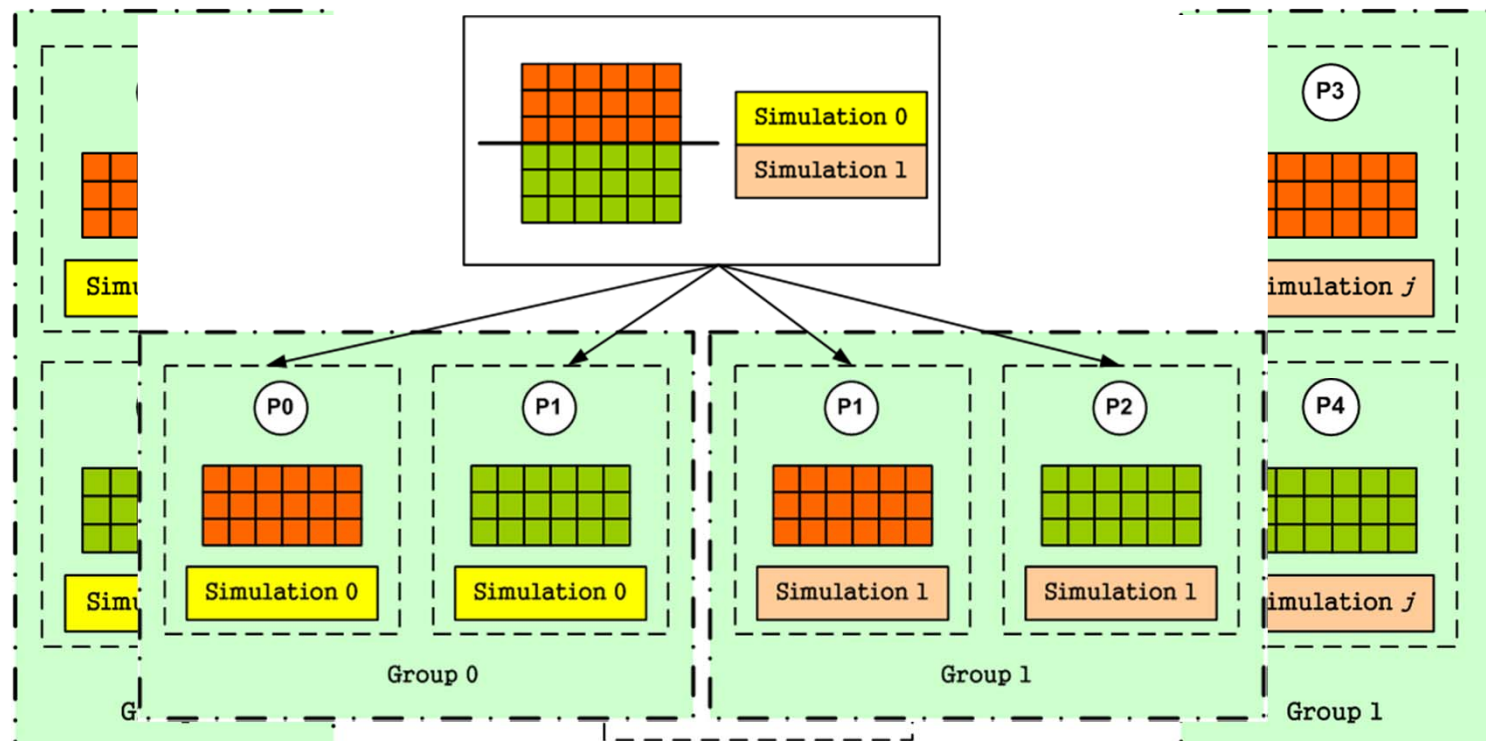
- ❖ The four growth rules in the SLEUTH model, were implemented using pRPL



pSLEUTH: Parallelization (cont.)

❖ Processor Grouping

- With pRPL, pSLEUTH is able to organize the processors in groups. Data parallelism within a group, task parallelism among groups.
- Static tasking and dynamic tasking



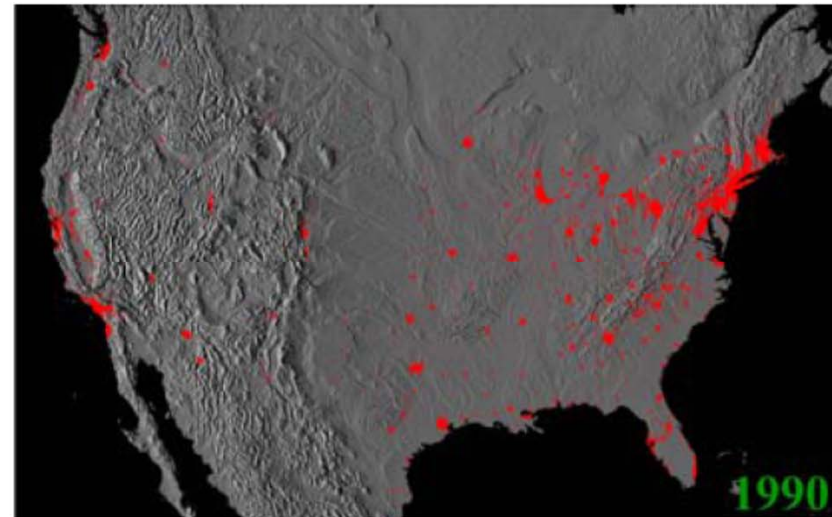
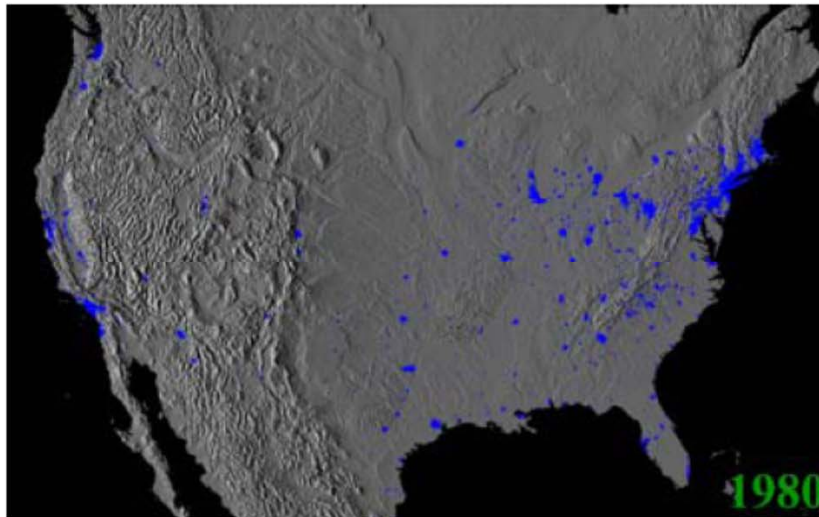
pSLEUTH: Experiments Settings

❖ Data

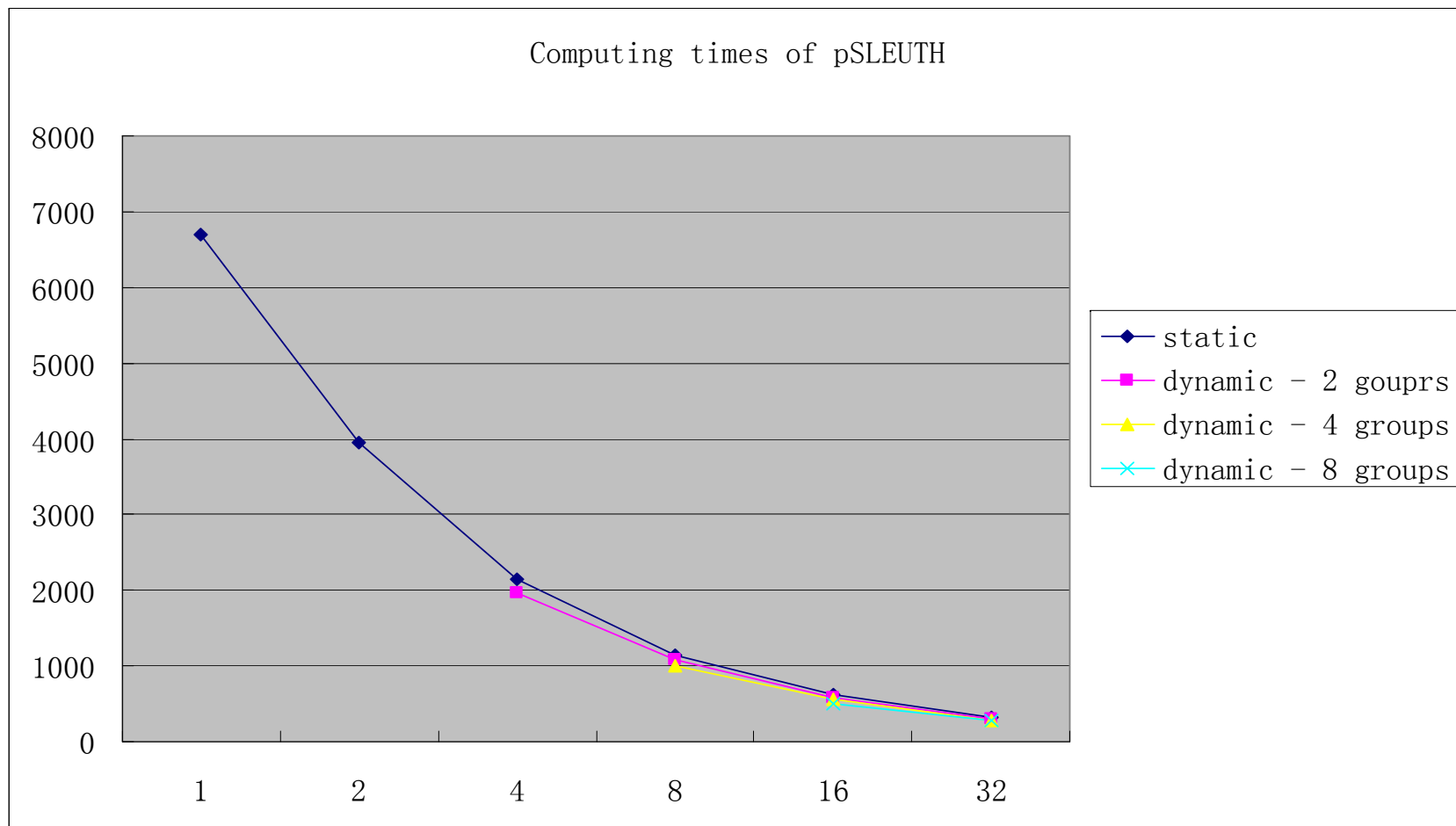
- Urban areas of the continental US (1980 and 1990, 4948×3108)

❖ Calibration Settings

- Only three values (0, 50, 100) will be evaluated for each coefficient
- The total number of simulations is 243 ($= 3^5$)
- Each simulation includes 11 ($= 1990-1980+1$) years

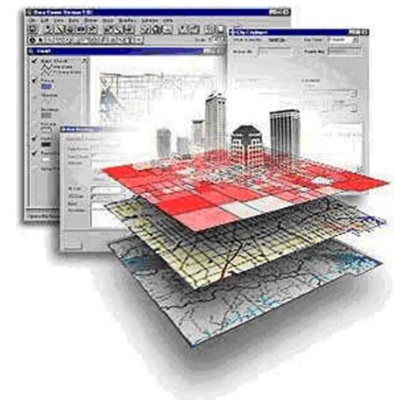


pSLEUTH – Performance



Sub-Conclusions on pRPL and pSLEUTH

- ❖ pRPL greatly reduces the development complexity of implementing a parallel raster processing algorithm
- ❖ pRPL can be used for many types of raster-based processing, including Cellular Automata (CA) and Agent-based Modeling
- ❖ pRPL largely reduces the computing time, and enables extremely complex geospatial analysis and modeling



Conclusions

❖ **GeoComputation is the application of Computational Science in geospatial studies**

- A large variety of computer-based statistical and mathematical methods for the analysis and modeling of complex geospatial phenomena
- A natural next step of GIS
- High-performance computing enables extremely complex geospatial analysis and modeling using massive-volume data

❖ **GeoComputation can be used as an exploratory-analysis tool, a simulation method, a problem-solving environment, a decision-making-support and planning tool, a theory test-bed, and a theory-discovery approach**





Comments and Questions?

Thank You!