Identification of Cellular Automata:

theoretical remarks

Alessandra Laghi Giovanni A. Rabino

Politecnico di Milano
Dipartimento di Architettura e Pianificazione
Via Bonardi 3, 20133 Milano

giovanni.rabino@polimi.it

ABSTRACT

Modeling nonlinear system has come on the fore in the last decades; some results are now available, even if precision and simplicity of application characteristics of linear problems do not hold anymore.

One of the most important issues to face in structuring a nonlinear model is the identification of the rules that govern its evolution. This paper collects some of the results achieved for solving this problem, particularly analyzed from the point of view of Cellular Automata.

Introduction

The purpose of this paper is to present the problem of identification of Cellular Automata, providing tools to solve it derived from nonlinear systems theory.

The first chapter introduces some definitions and fundamental aspects of nonlinear systems: state variables, dynamic of the system, possible classification of nonlinear systems, different approaches available for an observer, concepts of reachability and observability.

The second chapter presents Cellular Automata as nonlinear systems; the concepts explained in the previous chapter are applied to this kind of models.

Chapters 3 and 4 face up respectively to direct and inverse problem issues. They are structured in the same way: at the outset the problem's characteristics are discussed as well as the difficulties that arise. After, different methods to solve these problems are explained, sometimes using ideas drawn from other disciplines.

In the end, in the fifth chapter some remarks are presented about applying the various methods discussed to real available data.

1. Nonlinear systems: preliminary definitions

A dynamic system is a model of any part of the real world that evolves.
While the concept of dynamic system may be easy to figure, the definition of what evolves and how might not be so immediate to explain, because the same part of the world can be modeled using different techniques, depending on the aspect we are interested in.
The matter of "what evolves" is concerned with the *state of the system*, that is the set of variables that condense information of the system itself necessary to understand the evolution. You have to take into account that the state of a system may not be ever measurable directly by an observer; often, the observer can measure other variables, called *outputs*, which are linked to the state variables by a function.

The matter of “how it evolves” is concerned with dynamic itself, and it has two different aspects: on one hand, there are the rules which determine the evolution of the state of the system, on the other hand there are the relationships between state variables and outputs.

Moreover, the observer can interact with the system‘s evolution through *inputs* that perturb the system and make it react.

These concepts are mathematically synthesized in a couple of equations and an initial condition more:

$$\begin{aligned} &\dot{x} = \mathrm{f}(x(t),u(t),t) \\ &y(t) = \mathrm{h}(x(t),t) \quad , \quad x(0) = x_0 \end{aligned}$$

where *x* is the state of the system, *y* is the output, x_0 is the initial state, f is the transition function and h is the function that transforms state variables in outputs.

A basic classification of systems is made on the kind of functions f and h:

- *linear*: f and h are both a sum of quantities, in each one appearing only one state variable raised to the first power;
- *nonlinear*: in the quantities mentioned above appear products of variables, or variables raised to the second power or bigger, or any other mathematical operator that lays out the previous definition.

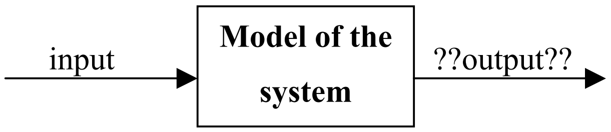
Linear systems theory is well developed because the superposition principle holds: a linear system can be divided into parts that are independent and as a consequence can be studied separately, the effects on the whole system being the sum of the single ones. Because of the superposition principle, such problems can often be broken into simpler pieces that can be solved individually, and then the results can be added together.

The superposition principle does not hold for nonlinear systems, so many difficulties arise which prevent us from using tools developed in linear theory.

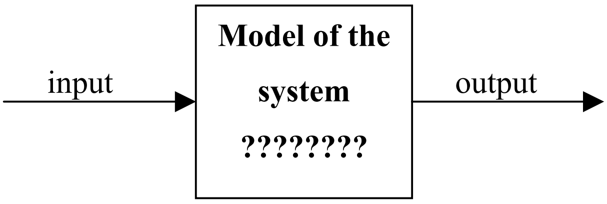
Further distinctions are made on state variables (continuous/discrete), functions varying with time (stationary/non stationary), stochasticity of state variables and/or parameters.

Two different approaches to dynamic systems are available:

- *direct*: the observer already knows the evolution rules of the system, so he tests the system using different inputs and recording the corresponding outputs:

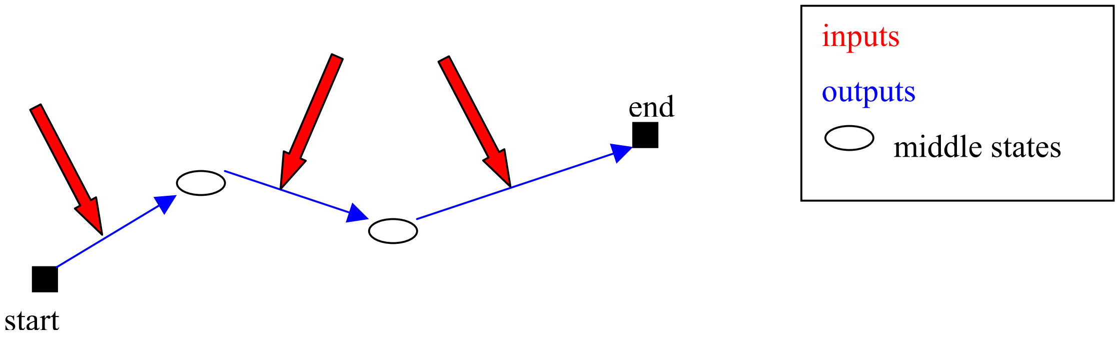


- *inverse*: the observer has a series of inputs and corresponding outputs measured during experiments and wants to reconstruct how the system works. This purpose includes both the functions f and h and the parameters that may are included. Usually the observer already has an idea of the kind of structure, thanks to his experience and through comparisons with other systems better known; he tries to fit the function to the case study, using available data:

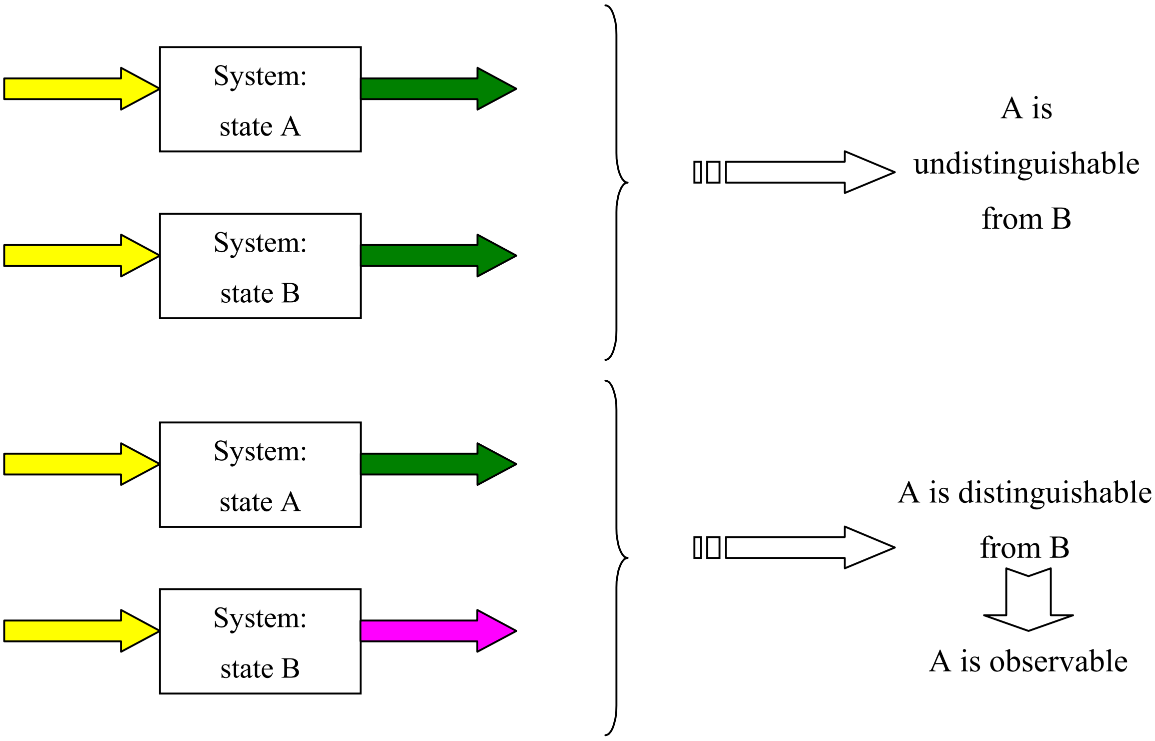


The latter approach is the one that happens more frequently and is called *identifiability problem* of a dynamic system. We may introduce more specific definitions depending on the available data: if they are noise-free (ideal case) it is called deterministic identifiability, otherwise (real case) it is a numeric identifiability. If functions and parameters can be estimated uniquely, the model is globally identifiable from the experiment; if any of a finite number of alternative estimates for them model parameters fits the data, the model is locally identifiable; if any of an infinite number of estimates fits the data, the model is unidentifiable from the experiment.
Two important properties of state variables are to be outlined, reachability and observability, which both concern the interaction of the model variables with the external world (inputs and outputs).

Reachable states of a dynamic system are those you can move the system to, selecting the right inputs. This property implies the possibility to control the system, especially artificial ones (whose dynamic is obviously known) that must be governed:



Observable states are distinguishable observing their output in correspondence of the same input. As a consequence, if the system state is not directly measurable and the only way to detect the system evolution is through outputs, only those states which "stamp" the output with their hallmark can be recognize; all the other ones are told undistinguishable:



2. Cellular Automata and nonlinear systems

Cellular Automata are actually nonlinear systems. This statement will be proved in the next paragraphs through the definitions explained in chapter 1.

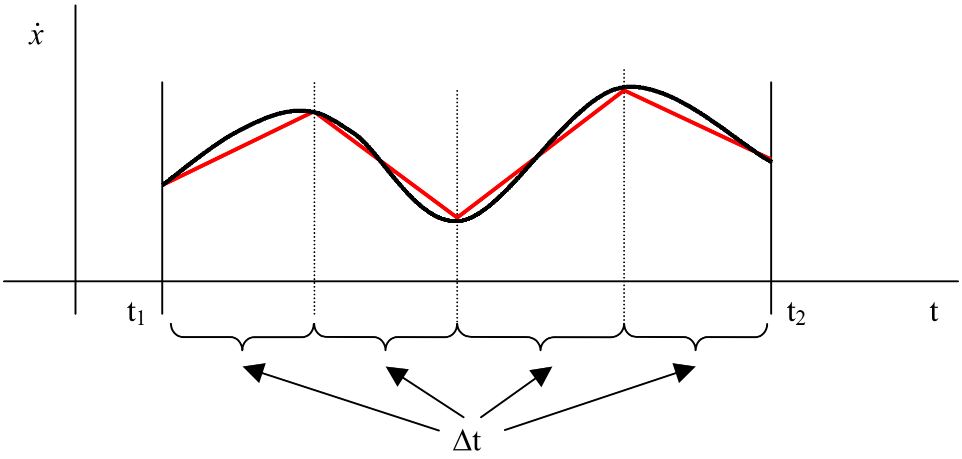
As a consequence of being made of cells, a Cellular Automaton is a system discrete in space; n-dimensional vectors describe its state, where n is the number of cells. Often in urban applications land use is chosen as state variable, within a finite set of values; as a consequence, state variables are measurable and you do not need to introduce a function that links it to outputs.

As far as inputs are concerned, the observer may change cells state.

Transition rules are a kind of "if….then" structure, where "if" introduces the neighborhood and "then" the cell state in the next step. This kind of rules is pretty far from linearity and this prevent us to use the superposition principle; moreover, spatial proximity concept itself, as a basis of this kind of models (if necessary broadened with long distance actions), prevents us from dividing the Automaton's space into independent zones and taking the sum of partial effects as a result for the global effect. This operation would create unnatural breaks between cells interactions damaging the system's evolution.

The equations we have just introduced to explain system's dynamic are ordinary differential equations of the first kind: in fact there is the first derivative of state variables that depend only on time. As we have learnt during mathematical analysis classes, this kind of equations is solved on an interval of time [t_1, t_2]; physically speaking, this means studying the evolutions of state variables during [t_1, t_2]. While equations explain how the system evolves now and forever, their solutions link the system to a determinate time period, giving us values for state variables.

Operatively speaking, solving differential equations means to simulate the behavior of the system in [t_1, t_2]; in order to do this kind of computations using a computer, it is necessary to divide [t_1, t_2] in smaller time intervals of the same width, Δt, used to approximate the derivative with the corresponding incremental ratio (in the next picture the state of the system is a one dimensional continuous variable to facilitate the representation):

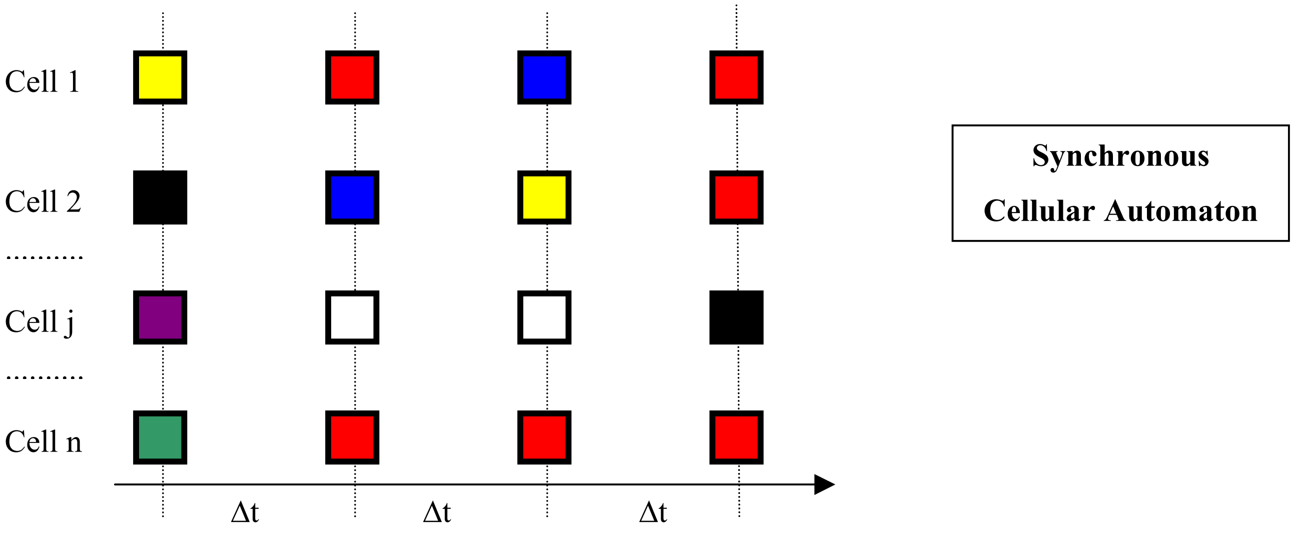


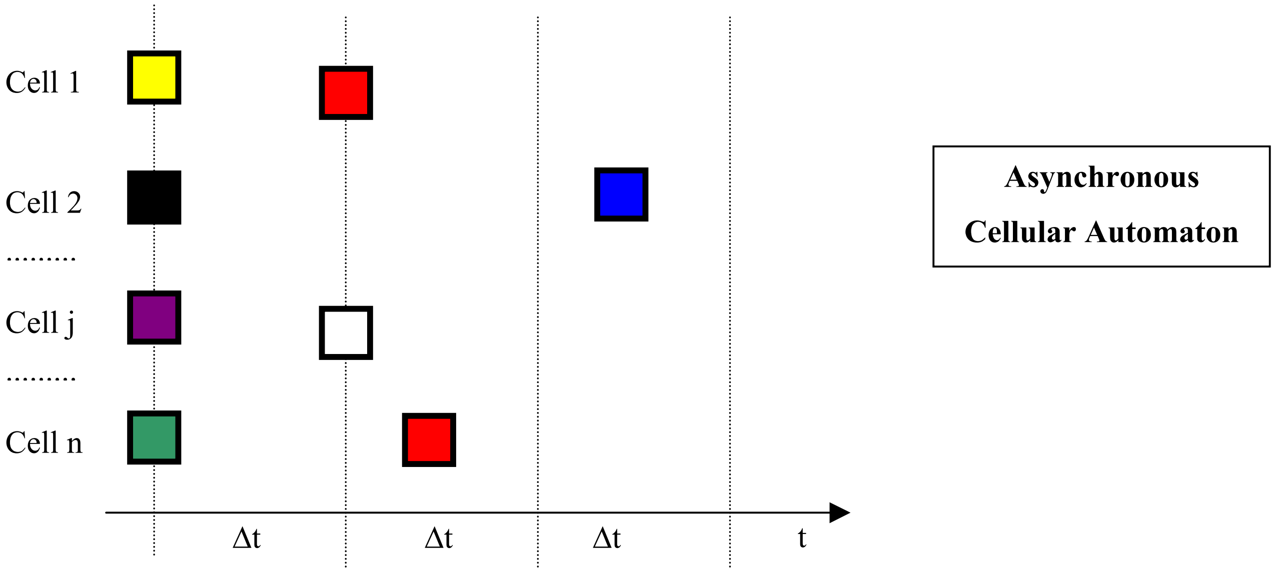
Solving differential equations that rule the dynamic of a Cellular Automaton implies some consequences on Automaton's characteristics.

Cellular Automata are discrete systems because their state variables take values in a finite set. The existence of a regular lattice, also, creates a regular grid where every cell state is computed as a mean value among all the existing values; in fact, this kind of discretization does not suit perfectly the land use distribution. As far as time is concerned, there is a clock that paces transitions.

As a matter of fact, in a real city transitions do not last the same time period: some are faster than others. On one hand computational issues force us to use a regular time interval to pace transitions, on the other hand the need to keep our model as close as possible to the real system requires irregular time intervals. A way to manage these opposite issues is modifying Automaton's characteristics in order to take into account the differences explained above.

First, Automata can be split into two subclasses, *synchronous* and *asynchronous*: the first kind develops transitions simultaneously, the latter admits that some transitions are slower than others. If Δt is smaller than the maximum time interval required to develop a transition, there is at least on transition that is late.





Pictures in the previous page show the characteristics of both these classes: above is a synchronous Automaton where three transitions develop at the same time for every cell; below there is only one transition that takes different time intervals to develop: while the first and the third cells are synchronized with clock pace, the fourth is one step late and the second two steps late. If we would have chosen Δt' = 3Δt, all the four transitions would belong to the same time step.

As a consequence an asynchronous Cellular Automaton succeeds in managing different time intervals required by transitions to develop; rules are applied when a transition is over, not simply when time step is over.

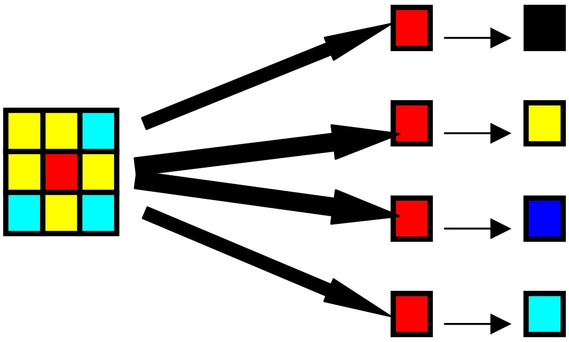
A second issue of time discretization is the *memory* of an Automaton.

In fact, every rule "if" introduces the neighborhood calculated in the current time step and "then" introduces the cell state in the next time step. The fact that transitions are not simultaneous implies that neighborhoods are difficult to define in every single time step because of late transitions, so it is necessary to keep memory of neighborhoods belonging to past time steps in order to reconstruct the right sequence of transitions. The deepness of memory depends on the case study.

In the end we can briefly explain *stationariness* and *stochasticity*, keeping next chapters to develop the problem of identification of Cellular Automata, distinguishing direct and inverse approach.

A Cellular Automaton is stationary if transition rules are always the same; if not, the differences might affect both the neighborhood and the transitions.

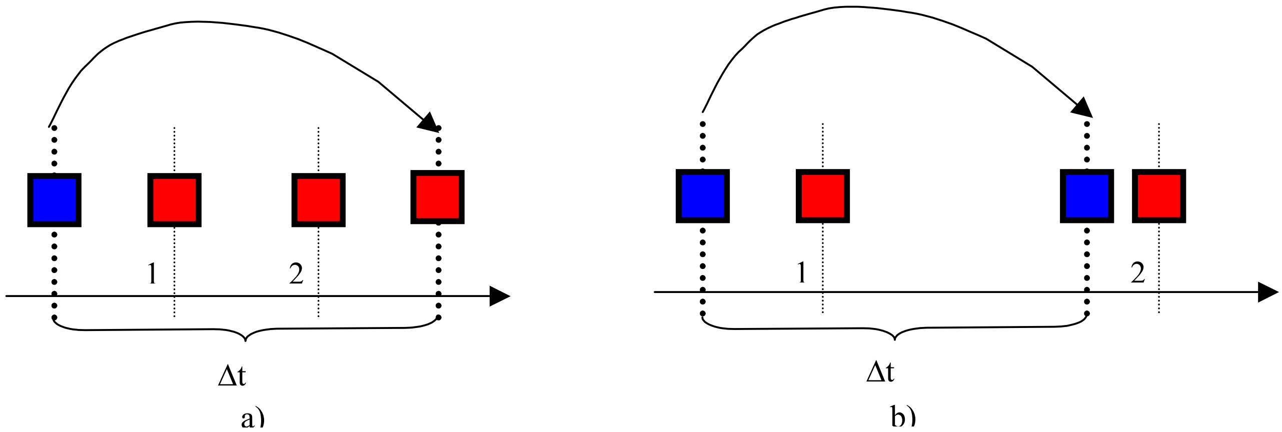
Stochasticity may affect both rules and delays of transitions. A transition rule is stochastic if the same neighborhood produces different transitions:



Every transition has its own probability to happen, and these probabilities are summed to 1.

A far as transitions' delays are concerned, stochasticity has important consequences only if transition is shifted so forward that it belongs to the next time step; if not, you may neglect it.

Next pictures show an asynchronous transition that can happen in two different times, 1 and 2.



In picture a), 1 and 2 are equal because in both cases the transition belongs to the same time step: the observer sees a red cell anyway.

In picture b), on the contrary, if transition happens in 1 the observer sees a red cell, if it happens in 2 the observer sees a blue cell, because the transition to red belongs to the next time step.

3. Direct problem

If the observer knows the system dynamic he has to face up the so-called *direct problem*, whose purpose is perturbing the system through different inputs and analyzing the corresponding outputs.

If we are dealing with a deterministic Cellular Automaton, there is only one way in which the system can evolve starting from a given configuration, once the observer has chosen the input; the observer whether decides to analyze the system's evolution without perturbing it or tries to interact changing one or more cell states and recording how these changes may impact on evolution.

This kind of one-way evolution does not hold if we are dealing with a stochastic Cellular Automaton.

In a direct approach, the observer already knows the probability distribution of every rule and wants to find the path of evolution most likely to happen.

A useful tool for this kind of problems comes from stochastic processes theory: *Markov Chains*. A stochastic process is a set of random variables depending on a set of parameters. In our case we may focus on stochastic processes dependent on one parameter, time; as far as it is paced discretely, we call them temporal series.

Markov processes are characterized by this property: the probability distribution of a variable at time t+1 depends only on the probability distribution at time t; we speak of chains when the state variable is discrete, as in our case.

Mathematically:

$$\pi_{\mathrm{k}}(t+1) = \mathrm{g}_{\mathrm{k}}(\pi_{\mathrm{k}}(t))$$

Markov chains are frequently used to describe the evolution of stochastic systems, which have a probability distribution of the state in every time as the system may be in state x_1 with probability π_1, in state x_2 with probability π_2, and so on.

The state of an Urban Cellular Automaton made of n cells is a n-dimensional vector

$$\pi_1(t) = \left[\pi_{11}(t) \quad \pi_{12}(t) \quad ..... \quad \pi_{1j}(t) \quad ..... \quad \pi_{1m}(t)\right]^T$$
$$\pi_2(t) = \left[\pi_{21}(t) \quad \pi_{22}(t) \quad ..... \quad \pi_{2j}(t) \quad ..... \quad \pi_{2m}(t)\right]^T$$
$$.....$$
$$\pi_n(t) = \left[\pi_{n1}(t) \quad \pi_{n2}(t) \quad ..... \quad \pi_{nj}(t) \quad ..... \quad \pi_{nm}(t)\right]^T$$

π_{ij} = probability that cell i is in state j at time t

whose elements take values in a finite set of m land uses; every cell has its own m-dimensional probability vector, $\pi_i(t)$:

As a result, this kind of tool is very useful for us to improve our comprehension of system's evolution; in fact, in our case as well, every neighborhood configuration has a number of possible transitions so that the state of every cell at time t is far from being uniquely determined from the previous time and depends on the probability distribution of the transition rules.

Markov chains theory says that function g, which determines the evolution of the probability vector of every cell, depends on $p_{ij}(t)$, that is the probability that a cell changes its state from i to j in the transition from time t to time t+1.

In our case these probabilities are the same as transition rules probabilities, which are known as far as we are concerning a direct problem. Probabilities dynamics is

$$\pi_{\mathrm{k}}(t+1) = \mathrm{P_k}^{\mathrm{T}}(\mathrm{t}) * \pi_{\mathrm{k}}(\mathrm{t})$$

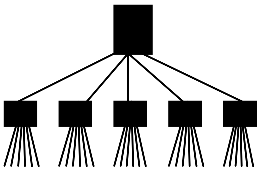
where $P^T(t)$ is the transpose matrix whose elements are $p_{ij}(t)$:

$$\mathrm{P(t)} = \begin{bmatrix} p_{\mathrm{k11}}(t) & p_{\mathrm{k12}}(t) & \ldots\ldots & p_{\mathrm{k1m}}(t) \\ p_{\mathrm{k21}}(t) & p_{\mathrm{k22}}(t) & \ldots\ldots & p_{\mathrm{k2m}}(t) \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ p_{\mathrm{km1}}(t) & p_{\mathrm{km2}}(t) & \ldots\ldots & p_{\mathrm{kmm}}(t) \end{bmatrix}$$

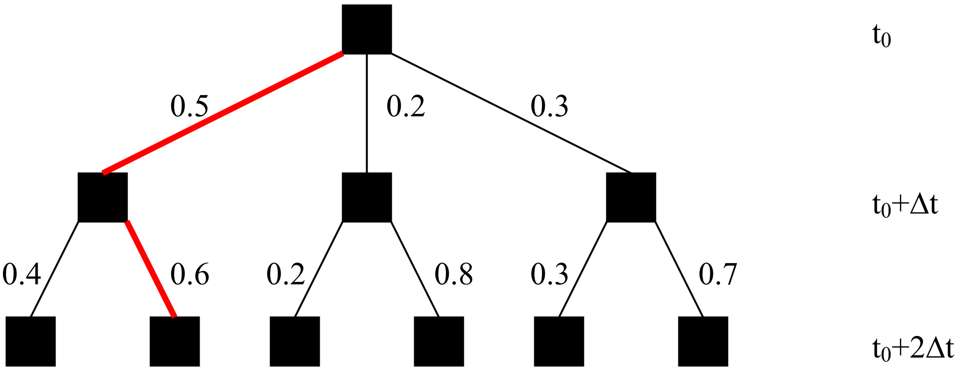
$p_{kij}(t)$=probability that cell k changes its state from i to j in the transition from time t to time t+1

This result allows us to admit that, even if the system is stochastic, the dynamic of probabilities is deterministic.

Using these tools the observer can develop the tree of possible configurations that is connected to a given starting configuration, because its probability vectors $\pi_1(0)$, $\pi_2(0)$,....., $\pi_n(0)$ are known: their elements are all null except one, with 1 as value, corresponding to the actual state of the cell. So the evolution of the system can be represented as a tree whose root is an available map and branches are all the possible paths; every branch has a weight that is the probability of the corresponding configuration, that is the probability that all cells make that particular transition.



If the observer wants to simulate the system 's evolution in a certain time period without perturbing the system, the problem is to evaluate the right simulation span according to the validity of the rules he possesses; in fact, while time goes on rules change because the system itself changes.
The level on the tree corresponding to last moment of the simulation is made of maps called leaves, as tree theory prescribes.
This kind of representation is useful to consider every possible evolution and detect the path most likely to happen:



If the observer wants to understand how changes made in urban land uses affect the evolution, Markov chains and trees also provide useful solving techniques.

This aspect is connected to the wide issue of system control through external actions, briefly explained in the first chapter while we were speaking about reachable states.

Controlling a system means trying to determine its evolution using appropriate inputs to modify transition probabilities.

The first step is constructing a function of state variables or other variables connected to them; this function works as an indicator of system performance and it is not unique: you may choose different *performance functions* depending on what you are interested in.

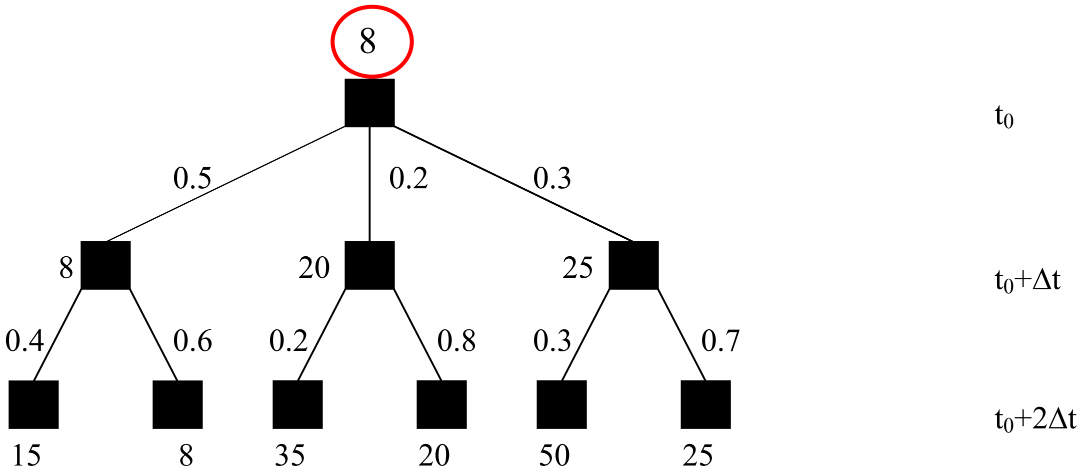
Later, after developing the tree without external interaction, the observer has to label every leaf with the corresponding value of performance.

In the end, going backward to the root, he has to label all nodes according to the theory of trees; the global performance is the one of the root.

For instance, suppose that the observer is interested in total public green surface in the city; the performance function is n_V, that is the number of cells in public green land use, and the purpose is:

$$\max n_V$$

The following picture depicts the previous tree added with performances of leaves and their fathers, obtained taking the performance of the child-node connected to the branch with the biggest weight:



The global performance of the system is 8, but the purpose of the observer is to maximize this performance; if he changes the weight of the branches he might succeeds in obtaining better performances.

The maximization is in respect of weights because they are the only things that the observer may affect through external actions, namely changing cells state. In fact, in a direct approach transition rules and their probabilities are already known (and given for this kind of systems), so the observer can only try to promote those transitions that imply public green land use as a consequence.

As far as stochastic Cellular Automata are concerned, we might compare this kind of systems with another one: *chess matches*, which can be structured in the same way.

First, let us analyze what aspects are in common:

- cells in Cellular Automaton space are the same as chess on a chessboard; the state of every piece is its position. Even if it would seem more natural comparing cells with chessboard squares, these ones do not change, they only host different pieces without being involved in any kind of changing. On the contrary pieces, moving on the chessboard, change their state.
- Time is paced by moves; every move is the same as a transition.
- Rules for moving every piece are precisely stated, they are deterministic, but the player has to move only one piece every time, so he has to choose the best move depending on the configuration on the chessboard.

Speaking about differences:

- While chess player can move only one piece in every move, more than one cell can change its state in a transition.

- Going on in a chess match pieces gradually become less, cells do not disappear.
- Chess player must move one piece in every move, but in a transition all cell can keep their state unchanged.

These differences affect this comparison only partially, so we may structure the two problems in the same manner. The observer (O) of the Cellular Automaton is like a chess player (P): they both have a starting configuration, from which develops a tree of possible configurations that depends on choices (for P) or transitions (for O); they both have a purpose (to win/ to maximize public green surface).

The system "match" does not have its own dynamic (pieces do not move themselves!), but every player depends on the other player's choices, so anyway there is a part of the system that a single player cannot control; he has to accept the way it develops, trying to interact with it. The probability distribution of every move he makes depends on the other player 's choices; every player tries to determine the evolution of the match towards his own purpose, which is to win.

The system "stochastic Cellular Automaton" has its own dynamic that the observer cannot modify (transition rules with their own probability) but with which he may interact trying to route it in the path he is interested in. Every cell can change its state according to a probability distribution that the observer cannot affect because it depends on complex mechanisms that involve economy, society,… Instead he can modify the state of some cells to promote the more probable transitions.

4. Inverse problem

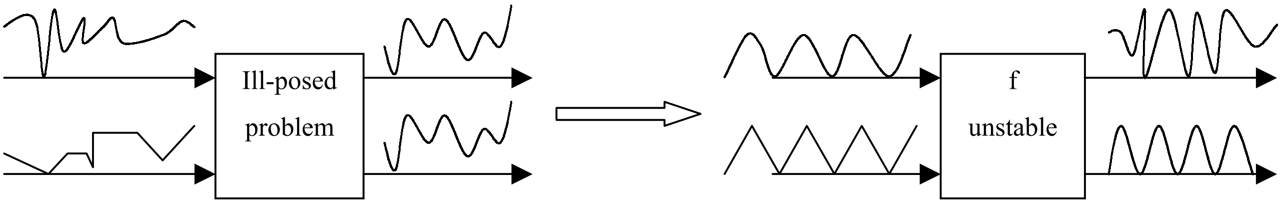
The most of the times the observer does not know the rules of the system's dynamic, namely when he is dealing with natural systems (that are not man-made). He tries to understand how it works, recording outputs without interacting with it and/or analyzing how outputs change if he perturbs the system.

The adjective "inverse" means that the order of cause and effect is reverse: the observer knows effects instead of causes and tries to deduce causes going backward; the unknown quantity of this problem is a function, that is the set of rules that determine the evolution of the system.

We begin with analyzing the most important issues connected with inverse problems.

Solving this kind of problems is strictly dependent on the quality of the available data, because it affects the uniqueness of solutions. A good general rule is that, in order to determine a function of n variables, one should collect data that also depend on n variables.

Often inverse problems solving is difficult because of the problem itself is *ill-posed*. In an ill-posed problem small changes in the observations may correspond to big changes in the phenomenon being observed. Ill-posed problems are difficult to deal with because the algorithms to solve them tend to be *unstable*, that means they are likely to produce very different solutions if inputs are changed just a little:



When the problem is ill-posed accurate observations are not enough; it is important to have prior information about the unknown system, derived from oneself experience. As a beginning, you may consider a very simple dynamic, even if it introduces strong simplifications, and then make corrections to make the model fit reality.

When you deal with well-posed problems (which are characterized by existence, uniqueness and stable solutions) more data available can improve the reconstruction; but more data in solving an unstable problem can be a double-edge sword. In fact, new data might generate a solution very different from the previous, posing the observer in front of a choice; this is the problem of *data consistency*.

We go on explaining some methods for solving inverse problems.

The first step in the case the system is continue is to make it discrete, dividing it into regions through a regular grid; in fact, the function we seek is supposed to be evaluated at every point of our system, in theory infinitely many points, and no computer has been made that can store an infinite amount of data. So every region will be treated as if it were a point.

Then, you might use the *optimization method*: you start with assigning a value at unknown quantities, as an attempt, and after calculating the correspondent values of outputs (y_i); you need some way to compare the simulated measurements with the real ones (Y_i).

You can do this, for example, by adding up the sums of the squares of the differences between the measurements:

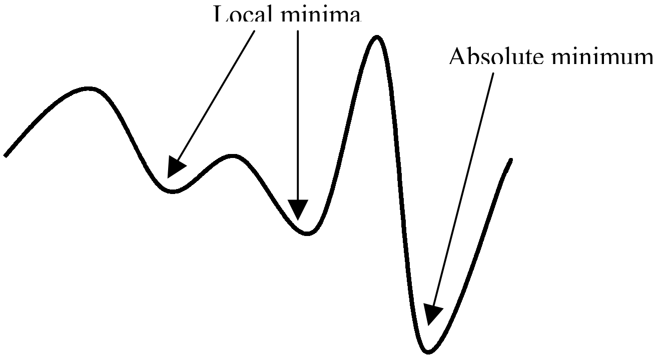
$$\min_{y} \sum_{i=1}^{n} (y_i - Y_i)^2$$

The distance function gives us a quantitative measure of the error in our discrete model, and the idea of the optimization method is to look for the values of the variables that make the error as small as possible.

Optimization methods are commonly used. If one needs to answer to a real inverse problem, optimization is a good technique to try. The ideas apply to almost every possible problem, and the methods can generally be stabilized. They produce a reconstruction that is reasonably consistent with the data.

consistenti con i dati.

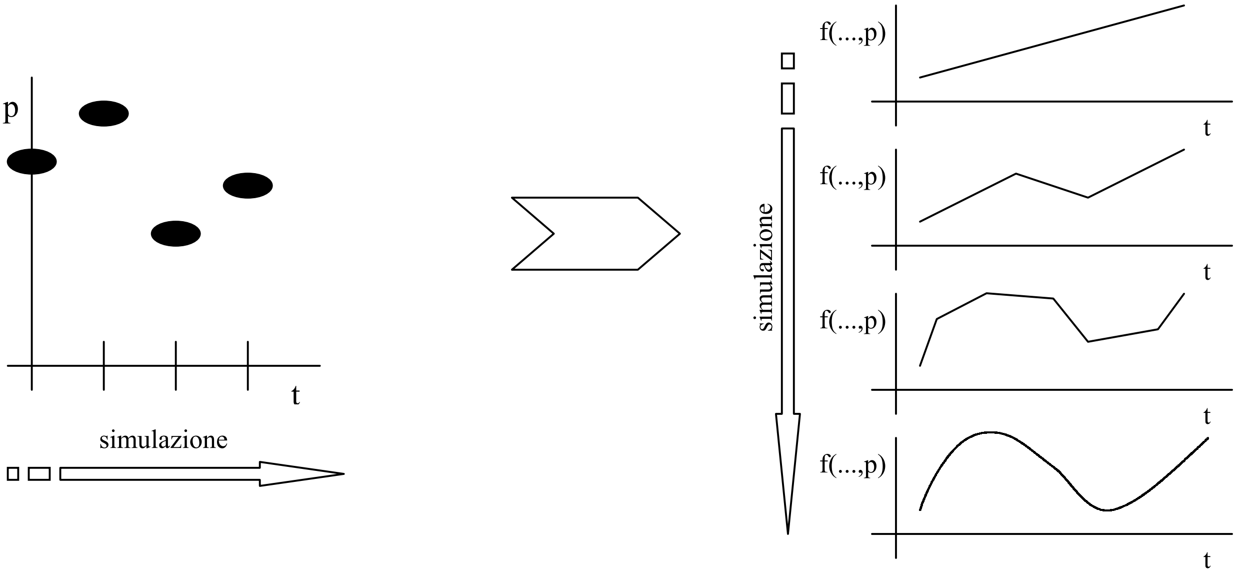
However, optimization methods tend to require a great deal of computation, and hence are slow and expensive. Also, they can be fooled by local minima that do not represent the smallest possible error. One cure for this is to start from very different values and see if the final result is the same.



A second classical approach to nonlinear inverse problems is *linearization*: substituting a nearby linear problem as a proxy for a more difficult nonlinear one. After making that substitution, we solve the linear problem using the measured data.

Linearization tends to be very useful when the problem is known to be close to a well-understood one. If there is reason to think that the true configuration is a small perturbation of a known one, then linearization is the method of choice. Algorithms based on linearization are often faster and require less computer memory than optimization methods. Moreover, a study of a linearized problem gives insight into the corresponding nonlinear problem. Unfortunately, linearization is often used- for lack of a better method- even in the case of large perturbations. In that case, it is not clear how the answers are related to the true solutions of the problem.

A more general method of solution is called *successive approximation*. There are many ways to make successively better reconstructions of an unknown system; one of the most used methods involves the dependence of the problem on a parameter. The procedure consists of changing the parameter in small increments, and making corresponding small changes in the reconstruction:

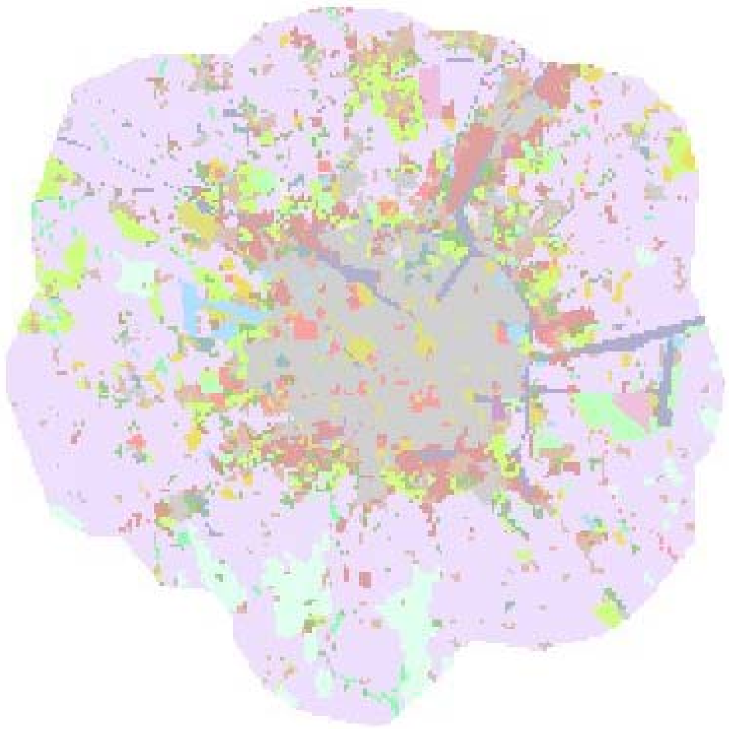


Successive approximation methods do address the full nonlinearity of the problem. However, they tend to be computationally intensive. Also, it is not clear under what circumstances the iterative process will converge to the correct answer.

Now we want to analyze the inverse problem's issues for an Urban Cellular Automaton. Here the challenge is trying to reproduce the behavior of the system "city"; available data are maps displaying cells land use.

Nowadays these kinds of data are easily collectable thanks to remote sensing instruments and software for working with images; the picture in the next page is an example of this kind of maps.

Of course we cannot assess that our data are noise-free, because as a matter of fact the procedure of dividing the city into cells through a regular grid introduces errors: the cell value is a mean of all the land uses that are actually in the cell. As a consequence, we are dealing with a numeric identification, according to the definitions introduced in the first chapter.



Example of map showing land use: Milan in 1955

Still, the fact that we are dealing with a system that cannot be reproduced in laboratory, make us focusing on data availability, instead of data pureness. Two important factors are the span covered by data and the frequency they are collected with; both these issues do affect greatly the precision we can obtain using a given dataset.

What assessments can we do about the structure of transition rules?

As we have already stated, the dynamic of a system consists in rules that determine the system's evolution now and forever. As far as a city is concerned, these rules do not keep unchanged while time elapses because of historical, economic and social factors; changing will be fast or slow depending on the particular context, but it inevitably takes place. So, if we are interested in models that work well on long run terms, we have to look for nonstationary rules.

Moreover, if only a few maps are available, it is worth structuring the system as a deterministic one, because this kind of model is easier to estimate than the stochastic one and requires less data to be fitted to the real system.

Choosing to model a city through a Cellular Automaton means that we want to use a kind of "if…then" rules. We are free to decide which kind of grid is the best, how many land uses consider and the span of a time step Δt.

Consider a regular grid of n cells and m land uses; we focus now on neighborhood and time issues.

If the observer lacks of experience, he may forms the starting hypothesis of a simple structure and modify it later, as the successive approximation method suggests for a parameter.

First, we consider a very lucky situation: the observer has 10 maps M_1, M_2, …. displaying the same area at one-year intervals. Obviously the observer takes $\Delta t = 1$ year.

In order to find transition rules, he might follow this procedure: after having decided a starting configuration for the neighborhood using the minimum radius, he compares the first couple of maps, recording the results in a table; every record contains neighborhood cells states and the corresponding transition. At the end of the comparison he scans all the records and looks if there are any neighborhoods that produce different transition; if yes, the set of rules collected is not deterministic. So he has to use a bigger radius and repeat the comparison.

On the contrary, if the observer forms the hypothesis of stochastic rules, instead of varying the radius he has to go on comparing successive couples of maps, collecting a series of tables. These are to be analyzed looking for neighborhoods that produce different transitions and drawing the probabilities of every transition.

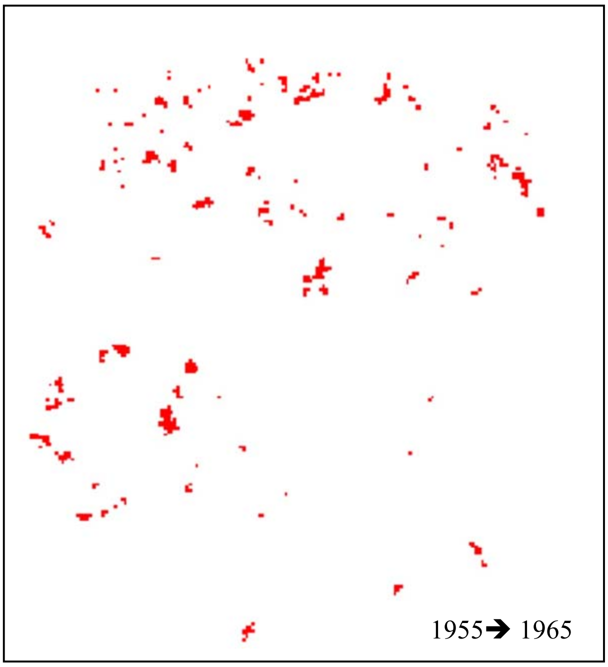
In both cases, in the end the observer obtains the set of rules that took place in those 10 years.

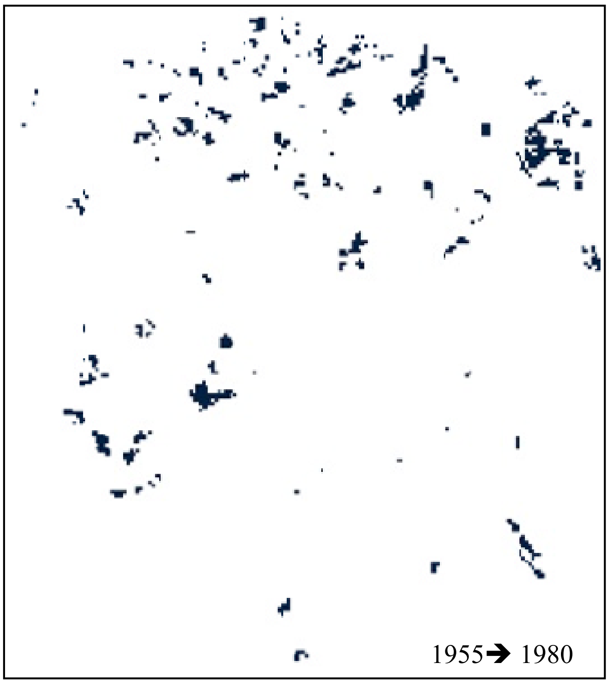
Unfortunately, most of the times we are not so lucky to have plenty of maps at equal time step. Frequency of availability affects greatly the choice of the simulation step and, in the end, the degree of precision we may obtain if we use the model to make predictions. In fact, rules derived from the comparison of maps N years far in time can be applied to make predictions on N years time intervals; you cannot hope to get more precise results with that set of rules.

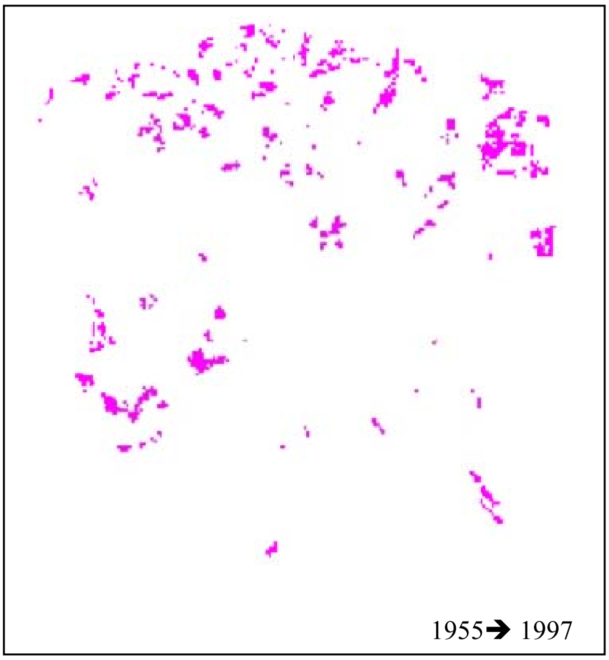
For instance, have a look to the following page: there are three maps showing the changes that took place between three couple of years: 1955/1965, 1955/1980, 1955/1997. Every map is concerned with cells that had an agricultural use in 1955 and became residential respectively in 1965, 1980 and 1997.

It is outstanding that the maps are different.

An observer, who would have had only one of them, would obviously drag conclusions different from the ones dragged by another observer equipped with all the maps.


1955➔ 1965


1955➔ 1980


1955➔ 1997

When availability of data is a problem, the best thing to do is simplify the structure of the model; minimal models, even if they could appear too naïve, are more reliable than more complex models because they require less data to be fitted to the real system.

5. From theory to applications

Now we introduce some observations about the application of previous concepts to the available data.

We have maps of three Italian cities and three European cities: Milan (1955, 1965, 1980, 1997), Palermo (1955, 1963, 1989, 1997), Padova-Mestre (1955, 1963, 1988, 1997), Dublin (1956, 1968, 1988, 1997), Wien (1958, 1971, 1986, 1997) and Porto (1958, 1968, 1983, 1997).

So, for every city it is possible to study three transitions; these maps were drawn using remote sensing instruments, and the represented theme is land use.

We have data and we want to understand how works the system that generated them: we are observers that have to solve an inverse problem, so we might attempt to use methods explained in chapter 4.

Optimization method prescribes to take the first map of the series and form and hypothesis on transition rules, hypothesis that is like a trial value for starting the iterative procedure. Applying these rules to the map, we get another map that we have to compare with the second map of the real series to calculate the distance between them. Our purpose is minimizing this distance, and this is done changing progressively the transition rules.

Once we have found the minimum, we go on analyzing the second transition: now the starting point is the second map of the series and the trial value are the transition rules gathered in the previous analysis. Following the same iterative procedure, this set is modified in order to minimize the distance between maps (the second generated by the model and the third of the real series). And so on, till all available maps are processed.

We have to pay attention when we initialize the procedure with trial transition rules: the more these rules are distant from the real ones, the more long the iterative procedure.

The *linearization method* applied to a Cellular Automaton implies to consider every cell as standing alone, as the superposition principle prescribes. As a consequence, we might structure an evolution path for every single cell without considering what is going on to the neighbors, at least as we have presented till now. In fact, the path may be made more complex considering the interaction with neighboring cells, but this kind of interaction must be linear, that is the transition depends on a weighted sum of the cells state considered.

You have to pay attention that this method gives good results only when the real dynamic is not too far from a linear structure; otherwise, the errors generated might be huge.

In the end the *successive approximation method*: starting with the first map of the series, the observer forms a simple hypothesis on transition rules set for one step and applies it to the map obtaining the evolution after one step. Then, he modifies the state of one or more cells in the starting map and analyzes how these changes affect the evolution, gaining information about how the system works. On the basis of these insights, he modifies a little the set of transition rules and goes on iterating the procedure.

This method is useful when there are only a few data available. On the contrary, if there is plenty of data and you prefer to use the optimization method, successive approximations may help you in finding a good trial value for initializing the iterative procedure of optimization.

References

Batty M., Xie Y. 1997. Possible Urban Automata. *Environment and Planning B-Planning and Design*, 24: 275-292.

Batty M., Torrens P. 2001. Modelling complexity-The limits to prediction, a paper presented at the *12th Europian Colloquium on Quantitative & Theoretical Geography*, September 7-11, 2001, St. Valery en Caux, France. (forth coming).

Casti J.L.1985. *Nonlinear system theory*, Academic Press Inc., Orlando, Florida.

Cheney M.1997. Inverse boundary-value problems. *American Scientist*, 85: 448-455.

Godfrey K.1983. *Compartmental models and their applications*, Academic Press Inc., London.

Groetsch C.W. 1993. *Inverse problems in the mathematical sciences*, Vieweg, Braunschweig, Wiesbaden.

Najim K., Poznyak A.S., Gomez-Ramirez E. 2000. *Self-learning control of finite Markov chains*, Marcel Dekker Inc., New York, NY.

O'Sullivan D. 2001. Exploring spatial process dynamics using irregular Cellular Automaton models. *Geographical Analysis*, 33 (1): 1-18.