

Apprentissage dans les modèles multi-agents

en économie

Christophe Deissenberg

Université de la Méditerranée et GREQAM

deissenb@univ-aix.fr

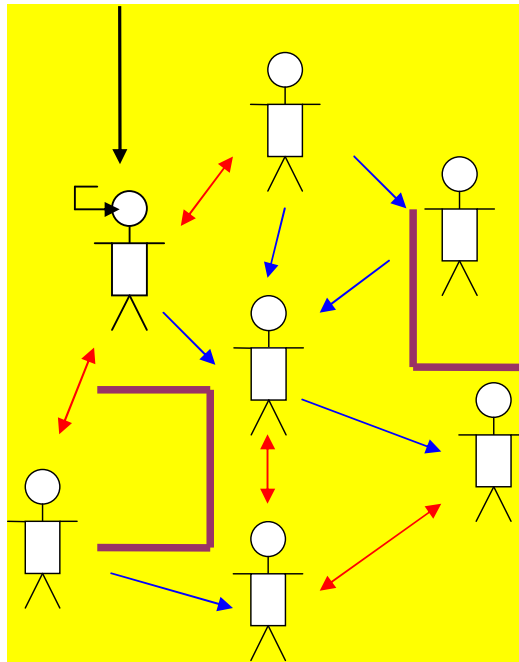
1. Introduction : Equilibre général, agents représentatifs, et modèles multi-agents *en économie*
2. Apprentissage dans les modèles standard et apprentissage dans les modèles multi-agents
3. Typologie des apprentissages : individuel, social, évolution. Apprentissage anticipatif et backward looking. Renforcement. Etc.
4. Systèmes de classifieurs et algorithmes génétiques et leur application à modélisation de l'apprentissage.
5. Illustrations :
 - Apprentissage et évolution
 - Apprentissage individuel et apprentissage social
6. Lectures additionnelles et logiciels

MARKET ECONOMIES

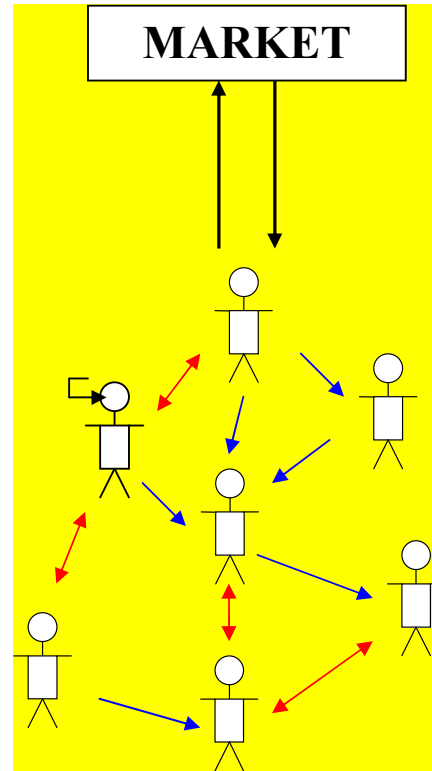
cf. Leigh Tesfatsion

- Large number of self-interested *economic agents* involved in *distributed local interactions*
- in a world crucially shaped by *Institutions*
- *Two-way feedback* between micro-structure and macro-regularities
- *Strategic behavior*
- *Behavioral uncertainty*
- Possible existence of *multiple equilibria* — or even *absence of any meaningful equilibrium concept*

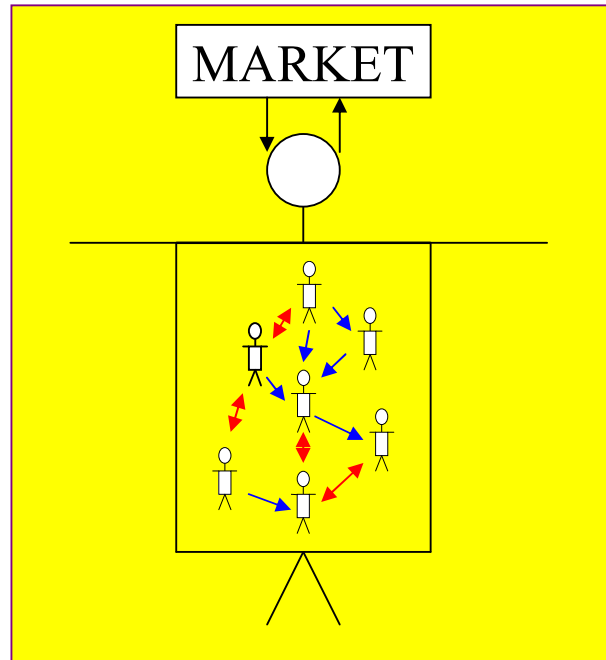
How is this usually captured in economics?



A real economy: locally rational agents



General equilibrium approach 1: efficient markets



General equilibrium approach 2: representative agents

But: A straightforward link

Many locally interacting agents → one representative agent

exists only under most restrictive conditions that go far behind the homogeneity of
the agents.

What does “homogenous agents” mean anyway in a world characterized by local interactions and slow transmission of information?

Learning in ACE Models

· **Neo-classical (general equilibrium) position:**

- There is a *true* model of the economy.
- Learning occurs through an “incrementally efficient” processing of additional information (e.g. Bayesian learning).
- Learning ultimately *has to* converge to a situation where the agents fully know the true model and thus can perfectly anticipate the future, modulo possible random chocks.

- Thus, the “orthodox” learning literature in economics (e.g. Evans and Honkapohja, etc.) focuses almost exclusively on stability and convergence to Rational Expectation equilibria.
- **Rational expectations** = anticipations based on the “true model “ of the economy = Benchmark in standard economics.

But in an ACE model and (presumably) in reality as well, only the modeler himself (or God) may ever know the true model, assuming it exists! An agent acting *within* the system will typically never be able to learn the true model within any reasonable time span.

This is due to the fact that (a) the underlying model co-evolves with the actions of the agent and in particular with the learning process itself; and that (b) the transmission of information (the impact of one's actions on its future environment) is slow.

- **Learning** is then basically a way to locally cope with a changing world. However, **the cause of the change in the world is partly the learning process itself.** There is a change in the perception of the underlying circumstances (learning), but this causes a change in these underlying circumstances as well.
- Thus, agents can *believe* they learn. But this learning will largely be *spurious*. In any case, it is undistinguishable from some sort of **reinforcement**, including rule selection through GA, neural net, etc. **Cf. Robert Axtell.**

NOTE: For simplicity's sake I may use in the following “probabilistic” terms – this does not imply that I refer to situations where there are well defined, known probabilistic distributions.

SOME DEFINITIONS

Learning and evolution

- **Individual learning:** Modification of an agent's behavior based only on its own experiences.
- **Social learning:** Modification of an agent's behavior into account the experiences of (all or some) other agents -- the probability of choosing another's strategy being normally a monotonically increasing function of the payoffs obtained by this strategy.

- **Evolution:** Modification of the composition of the population by elimination of the (**locally**) unsuccessful ones.

Backward looking and anticipatory learning

Backward looking: stimulus and response. Based on a comparison of the payoffs obtained in the past under different strategies. No deliberate attempts by the agents to “improve” their environment.

Anticipatory learning: Builds conditional expected probabilities of future gains under diverse strategies to ask the question: “If I take this action now, what outcome might occur in the future?”

Potential problems with anticipatory learning:

- infinite regress – back to equilibrium again?
- trade off between gains from experimentation and short term gains: “dual control”
- in any event, computational complexity of the underlying structure

NOTE THAT both backward-looking and anticipatory learning admit input-output (stimulus-action) representations!

NOTE ALSO that ACE researchers currently use many types of learning representations.

Supervised and unsupervised learning

Supervised learning: The output produced by an agent is compared to a given desired output. The deviation determinates the *fitness* of the objects that produced the output. Ex.: Multilayer perceptron.

Unsupervised learning: The system does not get any information about the desired output. It must find it's own classification of inputs. Ex.: Kohonen maps, K-nearest neighbour classification.

Reinforcement learning: Compromise between supervised and unsupervised learning: The desired output is not known, but there is a **fitness** (evaluation) **function** providing information about the quality of the solution. The tendency to implement an action is increased (reinforced) if it produces favorable results, weakened if it produces unfavorable results.

Reinforcement learning can be:

- **non-sequential**: no real dynamics, in each period the agent maximizes its immediate rewards
- **sequential**: there are true dynamics, the actions taken in one period may influence (in an unknown way) the future rewards →

CREDIT ASSIGNMENT PROBLEM

Example:

1. I am in state C, I take action 1, I get 0
2. I am in state F, I take action 4, I get 0
3. I am in state H, I take action 1, I get 0
4. I am in state Y, I take action 22, I get 0
5. I am in state Z, I take action 7, I get 0
6. I am in state Q, I take action 1, I get 2400

Nice, I got a big reward. But which of my actions along the way helped me to go there?

The problem is compounded in a multi-agents setting ...

Some commonly used reinforcement algorithms:

- Derivative-Follower (gradient-type)
- Gibbs/Boltzmann: Use a probability distribution to map rewards into future actions
- Er'ev and Roth RL
- Q-Learning (dynamic programming-type)

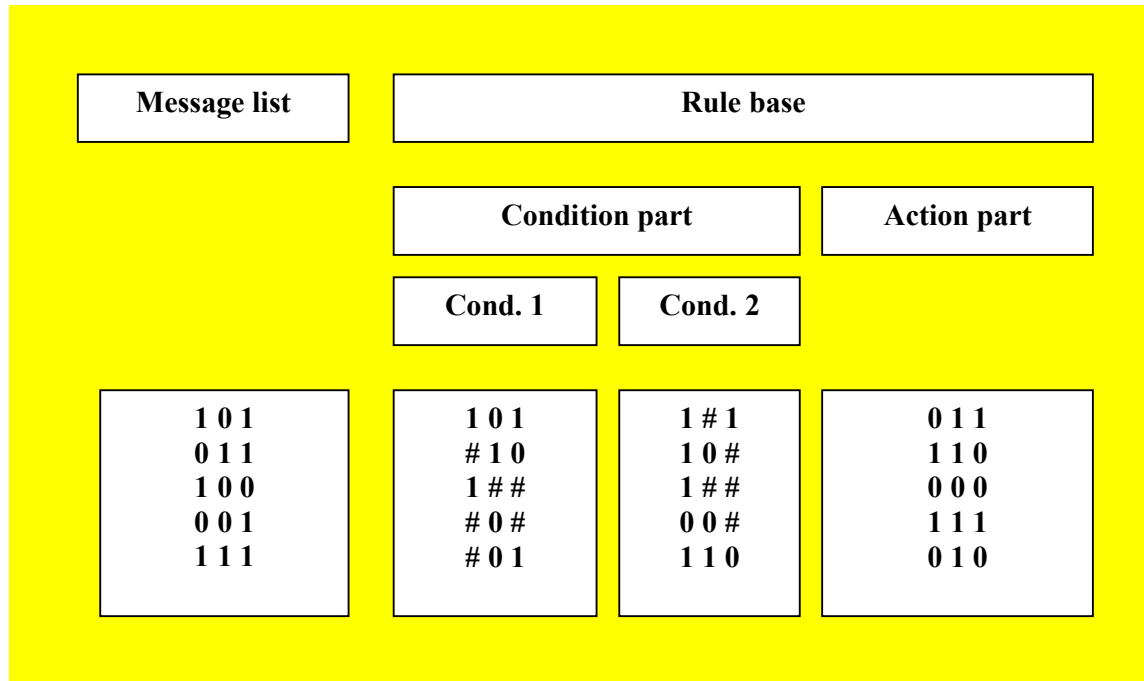
Classifier Systems CS and Genetic Algorithms GA

CS are rule-based decision mechanisms
GAs can be used to update CS

Classifier systems

Cf. Brenner 1999

- Holland (1976) – pattern recognition
- Rule-based decision mechanisms.
- Connection between input and output signals (typically) consisting of vectors of binary or integer entries → proper coding/decoding often necessary.

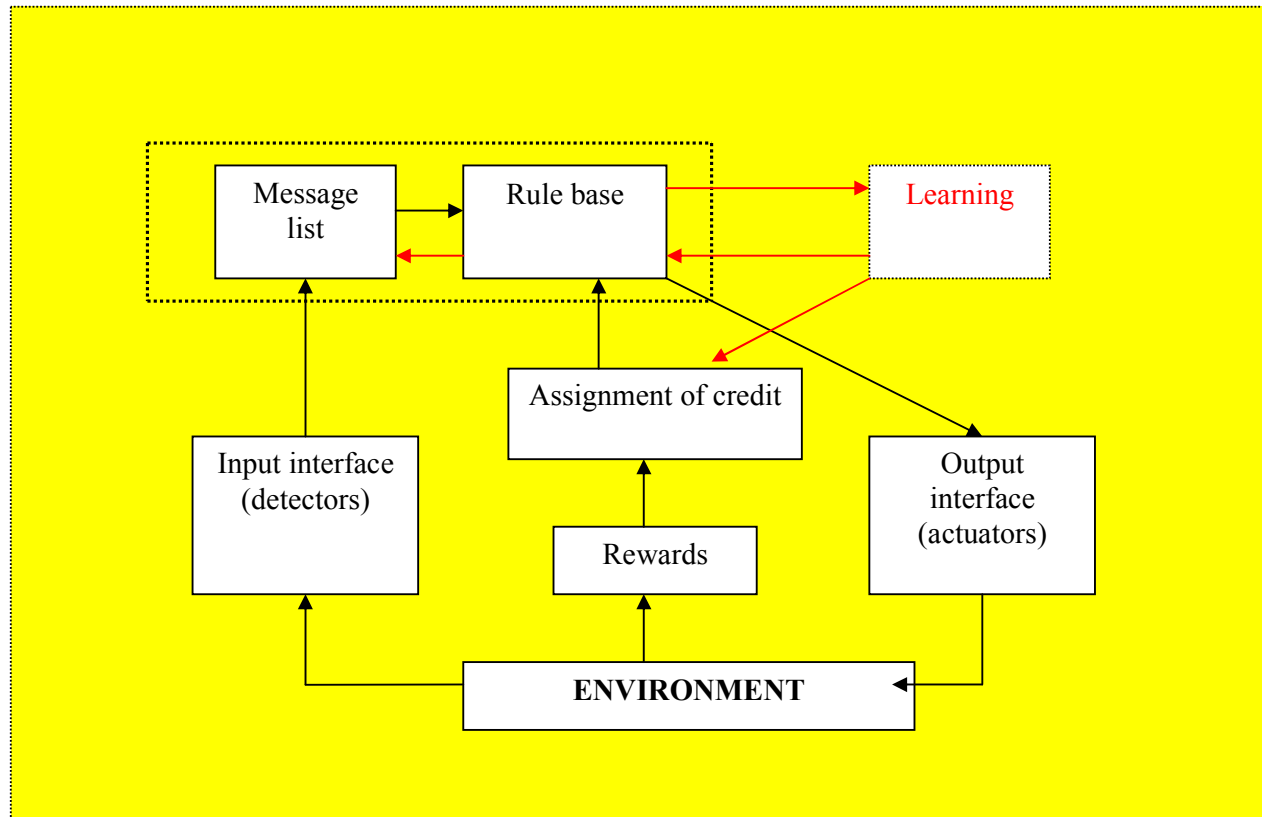


Message list and rule base of a CS

= “do not care”

Message list = information received by the agent (properly coded)

- 1) **Compare information with conditions, linewise**
- 2) **Message = condition modulo # → condition fulfilled**
- 3) **Here above: 1st, 3rd, 4th conditions fulfilled**
- 4) **The fulfilled rules are candidates for an action**
- 5) **If more than one candidate, select one randomly using as weight *strength* and *specificity* of the rule**
- 6) ***Strength* ~ relative success of the rule in the past**
- 7) ***Specificity* ~ frequency of # in the rule**
- 8) **The chosen action feeds back to the *environment* and/or to the message list.**



A classifier system

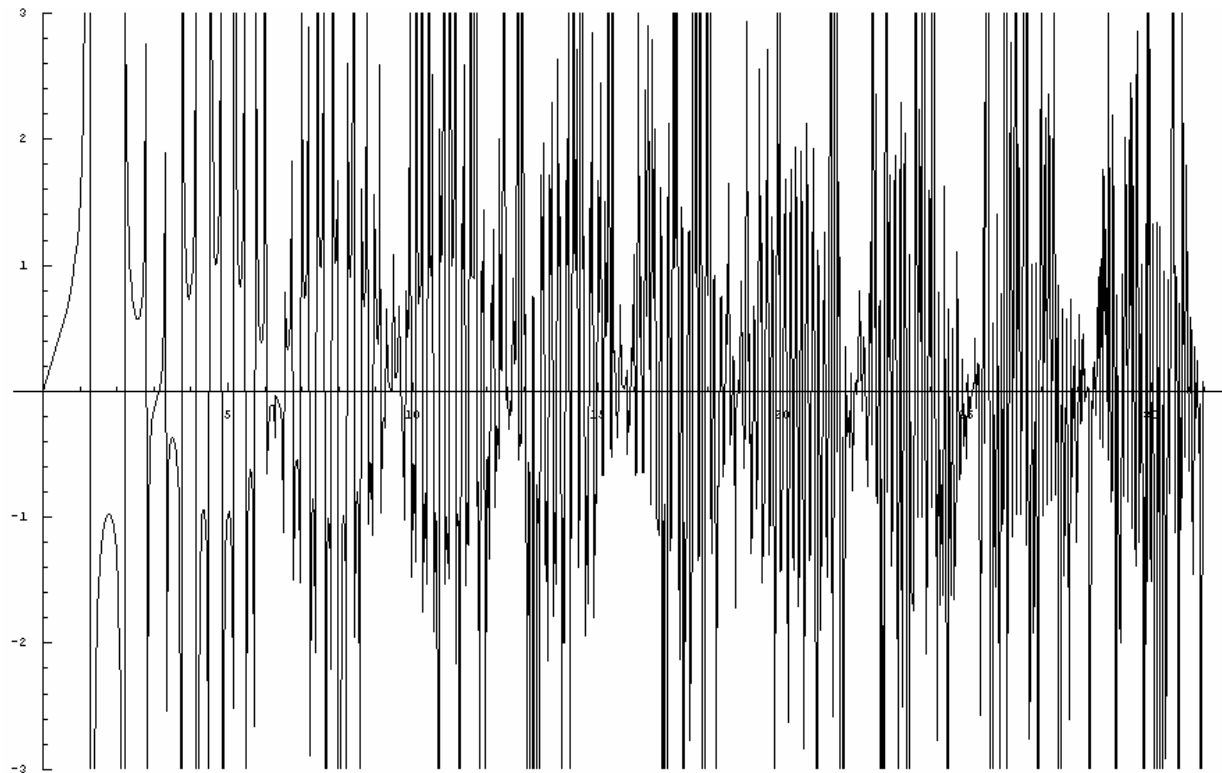
Straightforward extensions can be used to capture more complicated situations, such as time lags between receiving a message and reacting to it (understanding it).

LEARNING BETTER RULES

Basic idea: Create new rules using a GA (or any other suitable learning algorithm) on the present population of rules.

GENETIC ALGORITHMS

Genetically inspired, massively parallel search algorithms over large and complicated landscapes.



ELEMENTS:

- 1) Population of (e.g. integer) strings of fixed lengths
~ chromosomes

$$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5$$

Chromosomes ~ e.g. individuals, solutions, or in the CS case: rules

Set of all potential chromosomes = *search space*, very large

- 2) Fitness function: measures e.g. the
 - * ability of a string to solve some problem, to perform some task
 - * ability to survive and prosper in an artificial world
- 3) Genetic operators which, starting with the current population, increase the proportion of the fitter strings and generate new, even fitter strings

SEQUENCE:

- Generate a large, random initial population of genomes
- Evaluate the fitness of each genome
- Improve on the average fitness iteratively through:
 - 1) Selection: e.g. roulette wheel (global network), tournament (local network),

If we applied only selection, only *exploitation* would occur. We add *exploration* through:

2) Use (all or some) genetic operators:

- a) replication
- b) recombination
- c) mutation
- d) election

on the existing strings.

Structure as pseudo-code

Begin

$t = 0$

Initialize $P(t)$

Evaluate $P(t)$

If termination criterion not fulfilled

Begin

$t = t + 1$

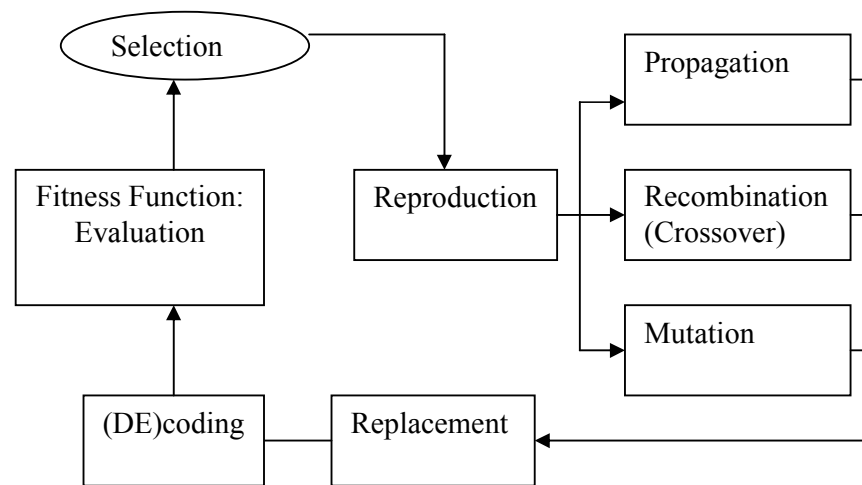
select $P(t)$ from $P(t-1)$

reproduce pairs in $P(t)$

evaluate $P(t)$

end

end



Basic structure of a GA: operations on the strings

NOTE: LEARNING VS. EVOLUTION

Key difference: Elimination of individuals or not!

Learning: changes the individuals

Evolution: changes the distribution of different given types of individuals

Gas can be used to model both:

- strings as *individuals* → evolution
- strings as *rules* → learning

SELECTION

The selection randomly chooses the strings that are allowed to have offspring in the next generation:

Roulette wheel selection, scaling techniques,
tournament, elitist models, and ranking models

Roulette wheel e.g.:

$$P_i = \frac{F_i}{\sum_{j=1}^J F_j}$$

Many variations!

Crossover

Simple crossover, e.g.:

Cuts the strings at a randomly chosen position and exchange the string tails:

crossover site

a_1	a_2	a_3		a_4	a_5
b_1	b_2	b_3		b_4	b_5



crossover site

a_1	a_2	a_3		b_4	b_5
b_1	b_2	b_3		a_4	a_5

Mutation

A randomly chosen a_i is replaced by a (randomly chosen) \tilde{a}_i

a_1 a_2 a_3 a_4 a_5



a_1 a_2 a_3 \tilde{a}_4 a_5

START AGAIN ...

Termination criteria

- Number of generations; or
- Convergence:
 - genotype: all bits positions in all strings are identical
 - phenotype: identical behavior with dispersant structures

WHY DOES IT WORK?

Consider e.g. binary strings of length 6:

The *similarity template* 00##11 matches all strings of the set {000011, 001111, 000111, 001011}.

GAs implicitly search for templates with higher fitness

There are many more templates than individuals:

$$2^6 = 64, \quad 3^6 = 729 \text{ e.g.}$$

Every time a GA processes a single individual, it in fact processes many templates.

When the GA processes a single individual, it in fact processes many templates and quickly eliminates bad templates.

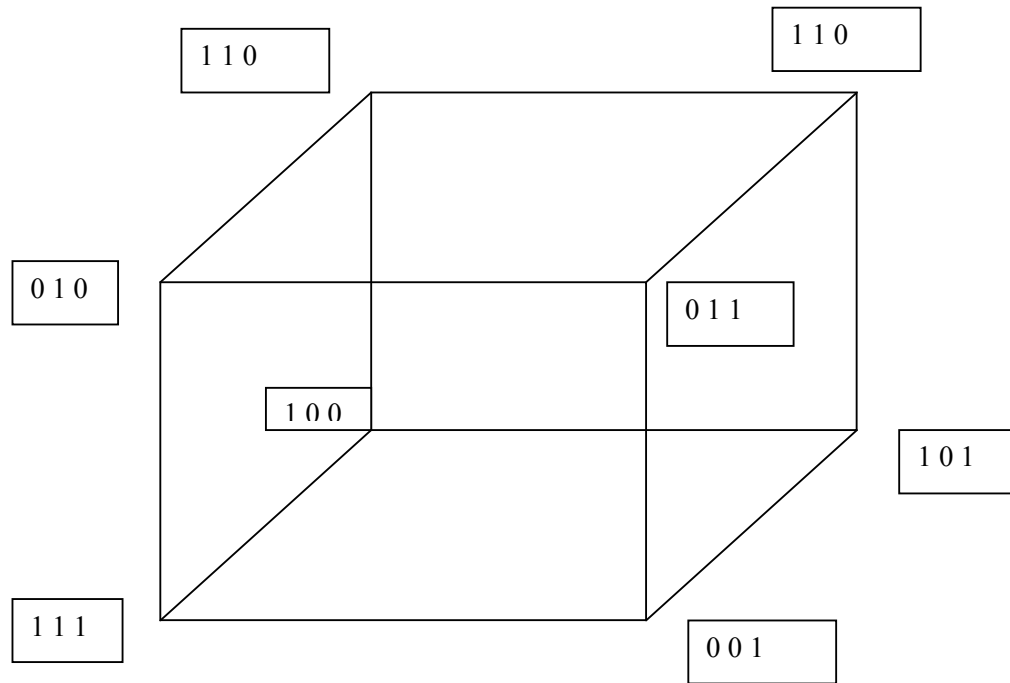
Crossover and mutation destroy existing templates, with a probability proportional to the:

* *length* of the template = distance between the first 0 or 1 and the last 0 or 1 in the template: #1###0## has length 4

* *order* of the template = number of 0 or 1s in the template: #1###0## has order 2.

Highly fit, short, low order templates are called *building blocks*. They tend to stay intact (short length) and process more individuals (low order).

Geometric visualization (D. Goldberg):



Space = # # #

Planes = templates of order 1, e.g. 1 # #

Lines = templates of order 2, e.g. 1 # 1

Points = templates of order 3, e.g. 1 1 1

SOME PROBLEMS WITH USING GAs in ACE

- How much information to give to the GA? Example: Discovery of the law of mentions underlying a chaotic time series, Chen and Yeh JEDC 1997
- Nonsensical rules → semantic restrictions, e.g *strongly typed genetic programming*: consistency of in- and output data
- Improper reliance on insights from Computer Science. Computer scientists usually try to efficiently get an answer to a *well specified* problem. In ACE, **we are not searching for anything in particular**. The choice of parameters of the GA depends from *empirical considerations* rather than from technical ones.

- Lack of systematic investigation of the difference between *individual* and *social* learning.

Individual and social learning

Jasmina Arifovic, “Genetic Algorithms Learning and the Cobweb Model”, *JEDC* 18, 1994, 3-28.

Nicolas Vriend. "An illustration of the essential difference between individual and social learning, and its consequences for computational analyses", *JEDC* 24 (2000), 1-19.

The economic framework

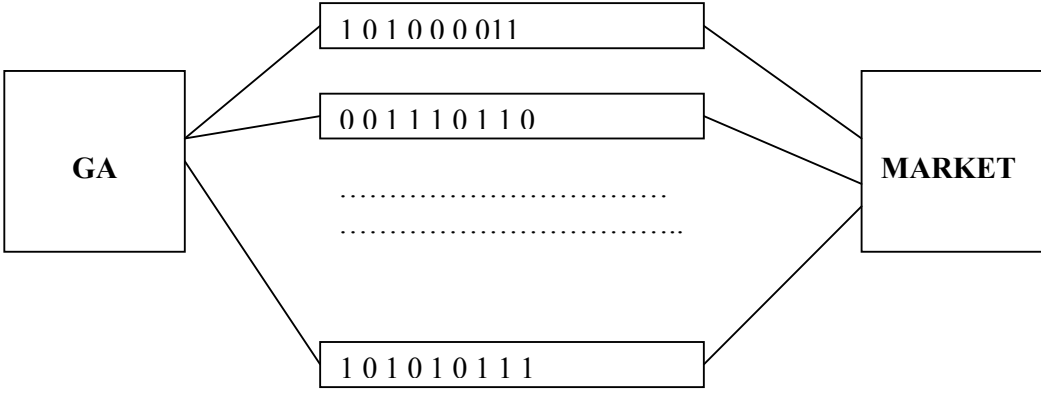
- Standard Cournot oligopoly game.
- n firms producing the same homogenous good
- Decision variable of firm i : q_i
- $P = a - bQ$ downwards sloping with $Q = \sum_i q_i$
- Fixed costs 0, marginal costs $k \rightarrow$ each agent is willing to sell any quantity at any price $P \geq k$, but would prefer to maximize his profit.

Social learning

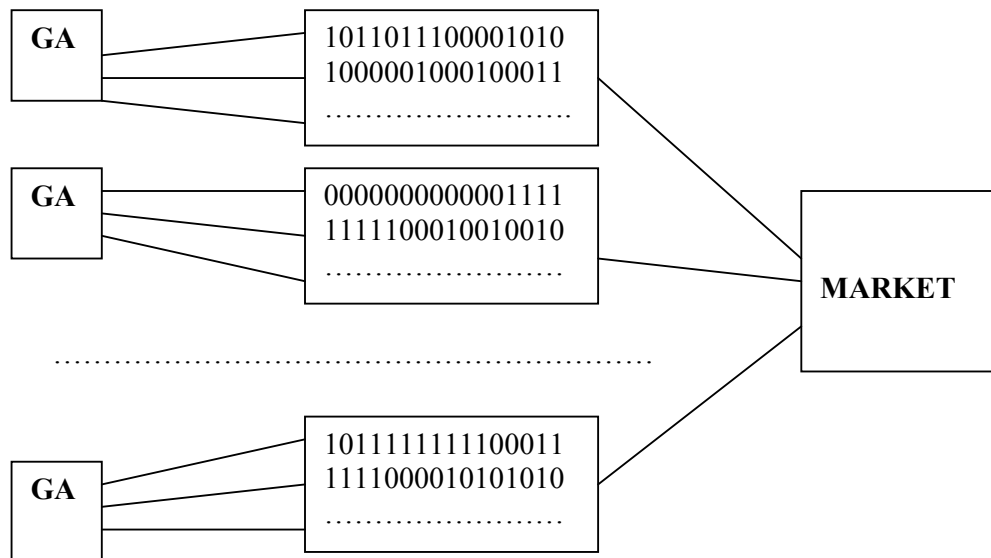
- Each firm is characterized by *one* output rule = binary string characterizing the production level
- Each day, each firm produces according to this rule
- Fitness of the rule = profits it generated
- Every 100 days, reproduction, mutation, crossover among the individual rules - with higher probability of using rules that were most profitable: GA

Individual learning

- Each firm starts with a *set* of rules
- In each period, each firm uses *one* rule
- Fitness of the rule = profits it generated when activated
- Every 100 days, reproduction, mutation, crossover exactly as in the social learning case -- but now comparing only the rules used by *the* firm



Social learning



.....

Individual learning

Results

40 firms, 25 runs

BENCHMARKS: “Standard” symmetric equilibria under perfect information:

- **Walras** (the firms are price-takers): $P = MC$, $Q = 80242$ i.e. $q = 2006$, zero profits.

- **Cournot-Nash** (the firms recognize that they can influence the price through their output decisions but assume that the quantities chosen by the other firms are given): $Q = 39928$, $q = 998$, positive profits.

Individual learning leads to Cournot-Nash, social learning to Walras!

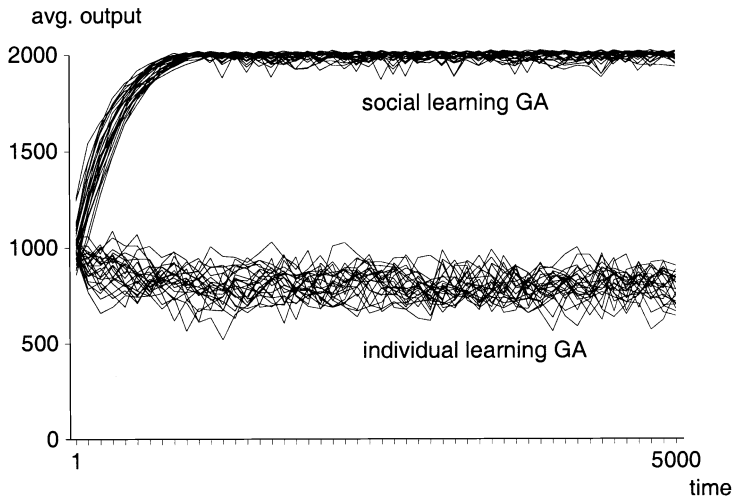


Fig. 5. Average output levels individual learning GA and social learning GA.

According to Vriend, the difference between social and individual learning is due to a "*spite effect*": Choosing an action may hurt oneself, but the others even more.

To recognize it, simple 2 firms example:

- $P = a - bQ$, $k = 0$
- Walras $\rightarrow Q = -a/b$, $q = Q/2$, profit = 0
- If one firm produces more, both firms make losses. But the firm that deviates makes the higher losses.
- If one firm produces less, both firms make profits. But the firm that deviates makes the smallest profit.

- Thus, the firm that produces at the Walrasian equilibrium fares always best. This even if the other chooses e.g. its Cournot-Nash share.
- Now, in social learning, each firm is characterized by its own production rule -- and all these rules compete one against the other. When $Q > Q^W$ the firms that produce LESS are advantaged, and vice-versa if $Q < Q^W$ → convergence to Walras.
- In individual learning, however, the rules that compete against each other do not interact on the same market! The learning process is unaffected by the spite effect.

REMARKS

- In the model, there is no co evolution of learning and environment
- The use of a GA as learning mechanism is irrelevant

BUT:

J. Arifovic, M. Maschek, WP 2005, Simon Fraser U.: Revisiting Individual Evolutionary Learning in the Cobweb Model – An Illustration of the *Virtual Spite Effect*

SOME READINGS

Brenner, T. (Ed), Computational Techniques for Modeling Learning in Economics, Kluwer 1999

Chipperfield. A. et al., Genetic Algorithms Toolbox for MatLab – Users' Manual, University of Sheffield 1998

Dawid, H. Adaptive learning by genetic algorithms, Springer 1999.

Fiebrink, R., A partial listing of open source software, 2005

Fudenberg, D. and D. Levine, Theory of Learning in Games, MIT Press 1998

Helbing, D. Quantitative Sociodynamics. Stochastic Methods and Models of Social Interaction Processes, Kluwer 1996

Holland, J. , Adaptation in Natural and Artificial Systems, 2nd Ed., The MIT Press 1992

Mitchell, M., “Genetic Algorithms: An Overview”, *Complexity*, 1, 1, 1995, 31-39

Tesfatsion, L. “Notes on Learning”, 2004. <http://www.econ.iastate.edu/testafi>

THE REFERENCE WEBSITE FOR ACE MODELING:

<http://www.econ.iastate.edu/tesfatsi/ace.htm>

SOME SOFTWARE

- The MatLab GA Toolbox
- The MatLab GA Toolbox of A. Chipperfield et. al. from U of Sheffield (freeware), comes with an excellent documentation.
- The Genetic Algorithms Archive: <http://www.aic.nrl.navy.mil/galist/>
- A large archive with many GA applications sorted by language.
- Genetic algorithms for the solution of optimization problems:
http://www.economics.ltsn.ac.uk/cheer/chap13_1/chap13_1p16.htm
- Large listing of free, commercial, and demo evolutionary computation software, including Gas: http://www.it.uom.gr/pdp/DigitalLib/EC/ec_soft.htm
- Another large listing of software: <http://www.GeneticProgramming.org>