Game-Theoretical and Evolutionary Simulation: A Toolbox for Complex Enterprise Problems

Yves Caseau

Bouygues Telecom, 82 rue Henri Farman, Issy-les-Moulineaux, France  
yves@caseau.com

**Abstract:** Complex systems resist analysis and require experimenting or simulation. Many enterprise settings, for instance with cases of competition in an open market or “co-opetition” with partners, are complex and difficult to analyze, especially to accurately figure the behaviors of other companies. This paper describes an approach towards modeling a system of actors which is well suited to enterprise strategic simulation. This approach is based upon game theory and machine learning, applied to the behavior of a set of competing actors. Our intent is not to use simulation as forecasting – which is out of reach precisely because of the complexity of these problems – but rather as a tool to develop skills through what is commonly referred as “serious games”, in the tradition of military war-games. Our approach, dubbed GTES, is built upon the combination of three techniques: Monte-Carlo sampling, searching for equilibriums from game theory, and local search meta-heuristics for machine learning. We illustrate this approach with “Systemic Simulation of Smart Grids”, as well as a few examples drawn for the mobile telecommunication industry.

**Keywords:** simulation, machine learning, game theory, evolutionary game theory, evolutionary algorithms, Monte-Carlo, dynamic games, serious games.

1. Introduction

Forecasting is an essential part of any enterprise’s activities. It is the essence of budgeting, marketing, sales management or strategic planning. Although forecasting is notoriously difficult and turns out to be wrong most of the time, it remains a necessary management tool. It is used to define a path and to align the company resources along a common objective, even if this objective needs to be constantly re-evaluated to take unforeseen events into account. Models which are used implicitly or explicitly to perform those forecasts are simplified to fit existing tools. Further-developed models would be necessary to capture interactions between competing companies. The most commonly-used tool is the spreadsheet. Spreadsheets make it difficult to capture two common traits of complex models: uncertainty and feedback loops. Hence the usual approach is to identify loops and uncertainty, make a few hypotheses to get rid of them, build a simpler deterministic model and rapidly forget about the simplifying assumptions.

Our proposed contribution is GTES, a tool to “play with” a model that is burdened with too much indetermination. This means learning a few insights from a repeated and organized set of simulations, mostly through exchanges and collaboration. GTES stands from *game-theoretical and evolutionary simulation* [1] and will be presented as a hybrid extension of evolutionary game theory [2]. This approach is based upon two key insights:

1. A number of classical techniques may be combined to get rid of a significant part of indetermination, such as sampling and machine learning.
2. Such “weak” models should be used to “play”, not to forecast. A global-scale model with too many unknown parameters is not useful to make a decision, but rather to help building a keener intuition about the situation at hand, through the repeated practice of scenarios.

This paper is organized as follows. Section 2 provides more detailed motivations for this work. We explain the limits of current forecasting methods and show why they need to be extended with a different approach. We present a few concrete examples which advocate for building a “model workbench”, that is, a tool to experiment with complex models of competition. For instance, we propose a “systemic model” for smart grids, where the “complex system” nature of electricity networks is made explicit, through multiple actors with conflicting goals.

Section 3 describes our proposed GTES approach. Formally, we need to solve a repeated non-cooperative game with uncertainty. The key idea is to separate the unknown parameters, from the model that is being studied, into three groups: the parameters that describe the goals of the actors – parameters that define the objective function, those parameters which are bound to an actor but dominated by the strategic choice expressed by the first group, and a third group of parameters which are independent from any actor – they describe the market in which the actors operate. GTES deals with each group of parameters differently: the first group is the control group, which defines a repeated game for which we look for Nash equilibriums [3]. The second group is managed with machine learning (the algorithm looks for parameter values that maximize the objective function) and the third group is managed with Monte-Carlo sampling [4].

Section 4 shows the application of GTES to our “Smart Grid System”, as well as a few other models taken from the author’s experience in the Telecom industry. Section 5 discusses different perspectives about GTES, from stability and performance point of views.

2. Motivations

2.1 Complex Models and Enterprise Simulation

As stated in the introduction, the ubiquitous use of spreadsheets has a profound impact on the way people see and think about their environment. Although many sophisticated tools have been designed for decision aid – from operations research to statistical and datamining tools –, spreadsheets still represent the vast majority of the modeling effort that takes place in an enterprise. It is not because that the vast majority of business activity is simple, it is because the simplicity of the underlying model is crucial to the appropriation that is required in most business situations. Our dual experience – ten years as an operation research scientist and ten years as an executive manager – has convinced us that a great decision tool that relies on a sophisticated model faces a lot of skepticism and distrust from business managers. Hence uncertainty is tamed with scenarios and ranges of values (proverbial *min* and *max* values), although scientific evidence shows that uncertainty about extreme values is even higher. The complex feedback loop that describe the fierce competition in an open market is reduced to making a few assumptions about “what our competitors will do” – most often, the same as they did before – which are treated as input parameters for the spreadsheet model. The result is a practice that is well suited to “complicated situations” (spreadsheets with many sheets and huge amount of data) but not to complex situations.

To take the complexity of a business situation into account, one would, on the contrary, expect to develop the following traits:

* Take into account the interaction loops between the company and its various competitors. This is one of the most obvious sources of complexity, but there usually are other feedback looks as we shall see later on.
* Keep within the model what is understood qualitatively, even if the quantitative formulas are unknown.
* Refrain from abusing linear formulas for elasticity or other forms of price sensitivity. Extrapolating trends that were seen in the past few years, which is quite easy with a spreadsheet, is another way of oversimplifying a business situation.

The intent here is nicely summarized by Carveth Read’s famous quote, often misattributed to Keynes: “*It is better to be vaguely right than exactly wrong*”. The work that is presented in this paper was born twelve years ago, while trying to evaluate the best strategies during the UMTS license bids. The dominant thinking at the time was that bidding for a new license was unavoidable because of the promises of 3G revenues. All market analysts had come up with similar spreadsheets models that described a truly significant increase of value creation. However, these models shared two short-comings: they did not factor the necessary price war that occurs when new actors enter a fixed-cost activity and need to gain market share, and the competition between the different motives for spending money in European households was not taken into serious consideration, although a wealth of accumulated knowledge existed. A first, simplified, game-theoretical simulation model was built to look into the dynamics of competition with a potential new entrant. With hindsight, these first simple simulations were quite relevant and provided a useful, though modest, contribution to the decision process.

2.2 Systemic Simulation of Smart Grids

Smart Grids are receiving a lot of attention from many scientific communities. The need for smart grids is emerging from the multiplication of intermittent, de-centralized energy sources, such as wind or solar plants. The possibility of smart grids comes from the combination of price de-regulation and the lower cost of IT, which makes reaching dynamic demand-response equilibriums possible through price signaling. The necessity of smart grids, as explained by its advocates, is the urgency of global warming and depletion of fossil energy sources, which requires a behavioral change as far as energy consumption is required. Smart grids (with their components, such as smart cities, smart neighborhoods and smart homes) are meant to provide consumers with dynamic and constant feedback. One of the promises of smart grids is to optimize the amount of “shaved” energy through the right price incentive - “shaving” is the reduction of peak consumption when the price gets too high (demand-response adjustment).

The idea to build a “serious game” model for smart grids came from the following list of questions. Without being an expert on energy production or distribution, there are a few puzzling reasons to be doubtful about “distributed smart grids”. Centralization dampens part of the variation through aggregation (the “independent” variation, according to the “law of large numbers”). It also leverages the “economy of scale”. Large-scale energy plants are more efficient (mostly because of the cost advantage of nuclear power plants), even when transportation costs are factored in. On the other hand, there are a few factors that are favorable to local operators (as opposed to national suppliers):

* Distribution means that production & pricing policies are better suited to each city.
* A distributed approach yields a “system of system” (network of smart grids), which may prove to be more resilient and better able to manage peak situations.
* CO2 taxes will encourage renewable energies which are easier to manage in a distributed way. A key question is the availability of storage at a reasonable price, since most renewable energies are intermittent.
* Feedbacks works better at a “community level”, which means that dynamic pricing and the “demand-response” behavior that produces shaving is likely to be more effective if managed locally. This is a key principle for smart grids in Japan and South Korea: focusing on communities and behavioral changes.

To get a better understanding of this complex system and its associated issues, we decided to build a simple model (S3G: Systemic Simulation of Smart Grids) with the four kinds of players:

* The “**regulator**” (political power) whose goal is to reduce CO2 emissions while preserving economic output and keeping a balanced budget (between taxes and incentives). Its tactical play includes setting up a CO2 tax, regulating the wholesale price for the suppliers and creating a discount incentive for renewable energies.
* The existing energy companies, here called “**suppliers**”, whose goal is to maintain their market-share against newcomers, maintain EBITDA (revenue) and reduce exposure to consumption peaks. Their tactical play is mostly through pricing (dynamic), but they also control investment into new production facilities on a yearly basis.
* The new local energy **operators**, who see “smart grids” as a differentiating technology to compete against incumbents. Their goal is to grow turnover, EBITDA and market-share. Their real-time tactical play is dynamic pricing, and they may invest into renewable and fossil energy production units, as well as storage units. Another tactical choice is how to best use storage capacity, both as a “buffer” (for their own production) and as a “reserve” (buy energy when it is cheap and resell it when it is more expensive).
* The consumers are grouped into **cities**, whose goal is to procure electricity at the lowest average price, while avoiding peak prices and preserving their comfort. The cities’ tactical play is mostly to switch its energy supplier (on a yearly basis) and to invest into “negaWatts”, which are energy saving investments (more energy-efficient homes, etc.).

The objective of S3G is to simulate the production and consumption of electricity throughout a long period of time (15 years). The following figure is an overview of the S3G model. The set of equations that describes each player’s behavior does not fit into this paper, but may be summarized with four parts:

* Energy demand: for each city, energy demand is generated from an hour-by-hour and day-by-day template, adding some random variation (the extent of which is a model parameter) together with a city-specific variation. This number is then reduced by the amount of “negaWatts”, computed from the total amount invested by the city. The model uses a ratio obtained from a concave-increasing function of the investment.
* Dynamic Pricing: both suppliers and operators use a simple affine pricing model, with a constant price when the demand is less than a “base power”, and a linear formula when the demand is higher.
* Production: suppliers use nuclear power according to planned schedule and adjust to resulting demand with fossil plants. Operators always use their green power (store it in the “buffer” or resell it when there is too much of it). They adjust to the city demand with their own fossil plant and wholesale electricity from suppliers, at the lowest marginal cost.
* Consumption: The actual electricity consumption for each city is the demand, minus “shaving”, which is obtained by applying an S-curve to the sale price.
* Market-share: for each city, the market balance between the national supplier and the local operator is determined yearly using another S-curve.



**Fig. 1** Systemic Simulation of Smart Grids

This S3G model is both simple and complex. It is simple because it is based on a handful of equations, resulting in a simulation code that is 500 lines long. On the other hand, it is a complex model for two reasons. On the one hand, there are multiple feedback and interaction loops that make it difficult to analyze how the system will react to perturbations. On the other hand, there are many unknown parameters in this model (such as market sensitivity, demand-response behavior, negaWatt capabilities, etc.).

2.3 Cellular Game Simulation

Our second example (CGS: *Cellular Game Simulation*) was co-developed with GTES over the last ten years. Its ancestor is the simple “competition game” that was mentioned in Section 2.1. The heart of the CGS model is a simplified vision of how an operator works, based on a few public figures that may be found in yearly financial reports:

* Turnover (sales), EBITDA (*Earning Before Interest, Taxes, Depreciation and Amortization),* ARPU *(Average Revenue Per User),*
* Number of customers (size of its “base”), yearly acquisitions (addition to the base), “churn” (removal from the base),
* Acquisition and renewal costs, interconnection costs (the other costs are labeled as “operational expenses”).

It is indeed truly simplified since all kind of lines of business (corporate vs. mass-market, MVNO, voice vs. data, prepaid vs. post-paid) are aggregated. This yields the concept of an “average telecom package”, sold for an “average price”. Operational expenses are supposed to be constant, or to vary according to a yearly trend. This simple model is interesting in itself because of its simplicity and the availability of indicators for competing operators.

However, the true interest of such a simple model is that it is possible to instantiate the three or four operators of the French market and see how they interact. The major tool that is used to implement this coupling is the S-curve function that describes market-share distribution according to prices. Each operator has three “control parameters” that describe its strategy on the open market: what is the price of the “average package”, how much is spent on renewal and on acquisition. A “tactical play” can be seen, as a first approximation, as the values of these parameters for the three coming years (what is commonly referred to as a 3YP – three year plan).

The concept of an “average telecommunication package” supports some form of elasticity, which has varied over the years and will continue to change as we enter a world of “access price with unlimited use”. Ten years ago the consumption unit was the minute of voice, and there was a fair level of elasticity. Today, the “average package” aggregates voices and data, from plans which are both unlimited (fixed-price) and usage-based. Hence the parameter that represents the price-to-volume elasticity in our model is both approximate and uncertain.

The overall CGS model is described by Figure 2. It may be decomposed into five steps, which are run consecutively for each year of the simulation (i.e., 3YP requires 15 steps). The **first step** is to compute the value of churn (the flow of customers who leave their current operator) as well as the flow of renewals (customers who obtain a new handset from their operator). In both case, the flow value is derived by applying the S-curve to the price variation. The **second step** distributes the combined flow of churn customers and new customers (market growth) into distribution channels. Based on the three-year plan (from which new prices are derived), the S-curve function produces the new market shares (for each channel). The **third step** is quite similar since it distributes the flow of customers assigned to each channel to all operators, using the same mechanism. The **fourth step** applies the previously mentioned price-to-volume elasticity to generate the turnover obtained by each operator. The **fifth step** computes the revenue by subtracting the costs.



**Fig. 2** CGS simulation architecture

There are a number of common traits to these types of models: S3G, CGS and the other telecom models which are mentioned in section 4.3. They are all “simulation model” that produce financial results (costs and revenue). They are “generic” simple models, but with a lot of “indetermination” (many of the key parameters in the parametric equations are unknown - or at least not known precisely). In each case we may separate all these unknown variables into three groups:

* Those which are independent from the players and simply reflect some uncertainty about the underlying economic model (for instance, the parameters for the S-curve function).
* Those which represent the strategy of the players, that is, what needs to be optimized.
* Last, the variables that are associated to each player, which set the behavior of the payers (such as the three “tactical” levers that we mentioned for the CGS model).

3. GTES: Game Theoretical and Evolutionary Simulation

3.1 Principles

GTES is a framework designed to study a model through simulation, in order to extract a few properties from this model (learning through examples), either explicitly or implicitly. The input material is a set of parametric equations, with a number of unknown parameters, which represent the behavior of a set of interacting actors/players. The result is a set of computational experiments, from which some information may be extracted.

GTES is based upon the combination of three techniques:

* Sampling: since the value of the parameters that occur in the economic equations is unknown, we draw them randomly from a confidence interval, using a Monte-Carlo approach. This will generate a large number of simulation runs, from which we look to extract common characteristics.
* Search for Nash Equilibrium in a repeated game [5]: We set the parameters that define the player’s objective functions and look for an equilibrium using an iterative fixed-point approach (in the tradition of the Cournot Adjustment [6]).
* Local Search as a machine learning technique to solve the sub-problem of tactical optimization. Once the parameters that define the objective function are set (what we call a “strategy” from a business perspective), the other parameters that define the behavior of each player may be computed to find each player’s “best response” to the current situation. This search is performed using classical local optimization techniques [7].

This approach sits at the intersection of « classical » lines of work such as economic modeling, game theory and evolutionary game theory [8] [9] [10] [11], and local search (genetic algorithms, simulated annealing, stochastic and Tabu search [12] [13] [14], etc.). We depart from the classical approach of evolutionary game theory since only one agent is introduced for each player, and genetic algorithms is only one of many optimization techniques that we use to compute each player’s response. On the other hand, GTES is clearly an offspring of evolutionary game theory [2], since we look for equilibriums that are the result of evolutionary processes. We care about the path to get to the equilibrium as much as the equilibrium itself.

This paper has been strongly influenced by works from R. Axelrod. In his book [15], R. Axelrod uses genetic algorithms to find the best strategy to play a repeated version of the “prisoner dilemma”. Very interestingly, he compares an experimental protocol and computer simulations (where the combination between many initial strategies is optimized with genetic algorithms). Both approaches reach a similar conclusion, that TIT-for-TAT is the most robust strategy for this game. More generally, this work also belongs to the field of economic modeling and simulation with multi-agent systems, using evolutionary algorithms [16] [17]. For instance, chapter 7 of R. Nelson and S. Winter’s book show an example of the kind of model that we try to study here, which the authors propose to simulate with Markov Chains. Using machine learning is a common thread in Evolutionary Games [6] [18] [19].

The different application examples that were developed using GTES are quite similar to simulations that have been made in the past following the “System Dynamics” framework [20] [21]. System Dynamics has a rich history throughout many decades, and has been applied successfully to corporate strategy and market analysis. The heart of this approach is to describe a model as a network of relationships between a few key variables from a system. The main feature is the “polarity” associated to each link: positive means that an increase of the originating parameter causes an increase in the destination parameter; negative means that an increase of the first causes a decrease of the second. Our experience is that this network is relatively easy to identify, whereas the actual equations that express the relationships between variables require more time to fine-tune. Because GTES uses local search and randomization, the underlying mathematical model is “shaken” during the simulation (cf. Section 5), and model tuning is much more demanding than it would be for producing a simpler simulation.

3.2 GTES Model

We shall now describe GTES with some details in the remainder of this section (a more formal description may be found in [1]). From now on, we will consider that the input model may be seen as a parametric function *f*p which computes the satisfaction of each actor. We suppose that the computation of *f*p entails the computation of all the local variables that describe each actor’s state. In the remainder of the paper *f*pi represents the i-th component of the result tuple, that is, the satisfaction of the i-th actor.

The parameter *p* represents the “strategy” of the model (not in the sense of game theory), that is how we evaluate the satisfaction of each player. Thus we will be able to compare the properties of the two parametric games *f*p and *f*p’, that is, with two different ways to evaluate success for each player.

The arguments of *f*p are the model’s parameters, which we represent by two variables *x* and *e*, according to the distinction made previously: *x* is a vector of numbers which represents the tactics of each player and *e* is a vector of unknown econometric parameters. If the tactic of an actor (what we called a “tactical play” in Section 2) may represented with *m* numbers and if there are *n* actors, *x* is a vector of size *n* x *m*, and *xi* will represent the tactic of the i-th player. We call *X* the set of all possible tactic vectors, and *X*i the set of all possible tactics for the i-th player (*X* = *X1* × .. ×*Xm*). To facilitate reading, we shall use the letter *x* to represent vectors of (*n*) tactics and the letter *t (*or *xi)* to represent one tactic – which is also a vector – for instance we may write *x* = (*t, t’, …*) = (*xi,x-i*). We borrow the convenient notation x-i from game theory [6].

Given the values of *x* and *e*, *f*p(*x,e*) represents the computation of the satisfaction of all players. This says that once a value is given for each unknown parameter, the set of equations that make the model may be resolved to find each actor’s behavior and, therefore, satisfaction. We call *E* the Cartesian product of each possible range for the parameters that are associated with *e*. Hence we can state the problem that we want to resolve as:

Since *f*p is a tuple-valued function, its maximization is a “multi-objective” maximization problem which fits precisely the framework of game theory.

3.3 Equilibriums and Neighborhoods

If we assume that all parameters are known but for xi, this turns into a classical optimization problem (single objective function *f*pi) and we may define the “best response” (BR in the tradition of game theory [6]) as a tactic (not necessarily unique) that maximizes the satisfaction of the i-th player:

BR(*i,x*) is a solution to this maximization problem, which may be solved using traditional techniques from operations research (depending on the complexity of the underlying model *f*p). In this section, *e* is assumed to be chosen and constant, so we will drop it from the formulas.

The most natural approach with our problem (1) is to search for is a Nash equilibrium [3], that is a tactic vector *x* such that:

There does not always exist a Nash equilibrium in a game with deterministic tactics. In a reciprocate way, there may exist more than one equilibrium. We will look for an equilibrium that is characterized as a fixed-point of a sequence of “best response” moves (where each player successively replaces its tactic with BR(i,t), what is called a *Cournot Adjustment*).

In the tradition of evolutionary game theory [2], we look not only for plausible equilibriums but also for plausible trajectories. If the set of tactics that represent the equilibrium requires a very complex mathematical computation to find it, one may wonder if it will be reached by real players. Hence we introduce a neighborhood structure on the set of tactics, similar to the neighborhood structure that is used with local optimization algorithms [7]. A neighborhood *V* associates to each tactic vector *x* a set of “near” tactics *V*(*x*). For instance, a k-neighborhood associates to *x* all the tactics that may be obtained by changing *k* parameters only.

We can restrict the concept of “best response” to the search of tactics from the existing neighborhood:

Local optimization is defined as the exploration of the transitive closure of the neighborhood structure: V\*(*x*) is the set tactics *t* that be found as the end of a path (*x,x1, …, t*) where each tactic of the path belong to the neighborhood of its predecessor, according to a given local search heuristic. For instance, the simplest search heuristic is Hill-Climbing, where we only look for increasing chains. We extend the previous definition to BRV\*(i,x) which is the best response in the extended neighborhood V\*(xi). A local optimization method is complete when BRV\* = BR.

This structure defines the concept of a pseudo-Nash equilibrium, when each player’s tactic is optimal modulo the set of alternate tactic from that neighborhood:

One may see this as the application of the “bounded rationality” concept: from a given position, each actor is not able to contemplate all possible tactics, but only a smaller subset which may be characterized through an operational process. Obviously, the choice of the neighborhood structure becomes a key feature of the actor model. The larger the neighborhoods are, the “smarter” the actors are supposed to be. The search for a pseudo-Nash equilibrium is an iterative process that looks for a fixed-point. We apply BRV\* to each actor successively.

More precisely, we may define the iterative process as follows:

* Select each player in turn with their current tactic t = xi,
* Apply a local search algorithm to compute t’ = BRV\*(i,x)
* Replace t by t’ in the tactic vector x,
* Measure the convergence (see later),
* Exit after a given time-out (or a given number of iteration *M*), or repeat.

Iterative search does not necessarily converge. When it does, we obtain a pseudo-Nash equilibrium by construction, and we qualify the GTES simulation as stable. If it does not, we distinguish between two cases: either there is a divergence pattern, where each actor become less satisfied after each iteration, which we qualify as a “war”, otherwise we cannot say anything and we quality the simulation as “chaos”.

The following figure shows two examples of GTES simulations, which we call trajectories. The first one exhibits a “chaos” behavior, while the second one shows a “stable trajectory” with the convergence towards a pseudo-Nash equilibrium.

We use three families of metrics to evaluate convergence:

* We perform a linear regression on the total satisfaction of all actors.
* We define a Euclidian distance and its associate norm on tactic vectors, to measure the distance between two tactics during the adjustment.
* We also define a Euclidian distance over a vector of internal state variables, to ensure that the states of actors converge when the tactics do.

We cut the trajectory into two equal pieces and evaluate its final status as follows:

* If the deviation of the linear regression is less than 5% and if the resulting slope is less than 1%, the trajectory is called “stable” [6].
* If the deviation is less than 10% and if the slope is less than -2%, the trajectory’s status is called “war”
* Otherwise, we set the status to “chaos”.

In the example from Figure 3, the metric that is used is the sum of the economic results (EBITDA) for all players. Hence the result of one iterative search is a triplet: *status, typical values, convergence measure*. The “typical values” come from a few selected state variables associated to the actors. In the CGS model, we use the EBIDTA, the market share, the total number of customers, the ARPU (average revenue per user). Obviously the most important of these values is the actors’s “satisfaction” function *f*pi(x\*). These typical values give a sense of what “the situation is like” when the equilibrium is reached. The convergence measures are drawn from the three previously mentioned metrics to qualify the degree of convergence.

**Fig. 3** Two trajectories for the search of an equilibrium (total satisfaction showed)

3.4 Monte-Carlo Sampling

We have described how a GTES trajectory is computed for a given parameter vector *e* (by convention we use Greek letters to designate such parameters). Providing *e* is performed through sampling [22]; that is, randomly picking a value for each “economic parameter” from a confidence interval. There are typically between 5 to 10 such parameters in most models we have applied GTES to. Since nothing is known about a probability distribution, each random picking follows a uniform distribution.

Here are some examples from the CGS model (Section 2.3):

*  is the price-to-volume sensitivity (the slope of an S-curve, cf. 4th step).
*  is slope of the S-curve that is used to derive renewal figures from price variations (slope of the *f* function = derivative at the 0 value).
*  is the slope of the S-curve that is used to derive sales figures from price variations.
*  is the number of month that the average customer uses when evaluating the TCO (*total cost of ownership*) of a phone package when comparing two operators (how to balance between the handset price and the monthly fee).

A complete GTES simulation (for a given value of the strategy parameter *p* which “defines the game”) is defined as the repetition of *N* identical steps: sampling to build the *e* vector, followed by the iterative search of an equilibrium as explained in the previous section. Each trajectory’s result is aggregated into an “overall” result triple:

* Status distribution: % of trajectories that are respectively stable, wars and chaos.
* Average/deviation for “typical values” (satisfaction, EBITDA, …)
* Average/deviation for the convergence metrics.

A key issue with Monte-Carlo sampling is the number (*N*) of drawings necessary to get statistically significant results. We measure the standard deviation of all the result components. It is easy to observe the increase stability of results and their interpretation when N grows. For most problems, we get interesting results with only a few hundred sampling trajectories, although a few thousand seems the ideal trade-off. Going to higher value of N only yields very small improvements, which are not relevant for such approximate models, nor for the way GTES is used (serious gaming).

3.5 Search for Equilibriums

We shall conclude this section with different techniques that may be applied to improve the search for equilibrium, namely: **interweaving** (of the two optimization loops presented in Section 3.3), **parallel** application of BR moves and “*minmax*” evaluation, yielding the concept of a “**Forward Nash Equilibrium**”.

The idea behind **interweaving** comes from the realization that GTES uses two embedded loops that search for a fixed point: the search for a local optimum for the tactic of a given player (computing BRV\*) and the search for a pseudo-Nash equilibrium, with at most *M* iterations. Hence we may decide to approximate the search for the optimal tactic with BRVq which is the best tactic found in the Vq neighborhood, which is the set of tactics that may be found with at most *q* moves from V. By construction (the limit exists because V*n* is an increasing sequence).

Interweaving means computing t’ = BRVq(i,x) in the sequence described in Section 3.3, and expecting the outside loop to progressively approximate best responses from V\*. There is a trade-off: if we select a small value of *q*, it will take many outside loops to reconstruct a longer chain and there is no guarantee that the interplay between the actors will not prevent from finding these longer optimization chains. However, the inner loop (exploring Vq for each actor) will run faster so we can run more occurrences of the outside loop (moving from one actor to another). Computational experiments could not be reproduced here for lack of space, but they show that a “moderate” form (*q* ranging from 5 to 10) of interweaving produces an improvement (faster convergence for the same run-time).

The choice of searching the best response of each actor in turn, sequentially, may seem surprising (this is the “alternate-move” versus “simultaneous-move” approach [6]). A more natural approach would be to apply the search of the best response in **parallel**: exploring BRVq(i,x) for all *i* and from the same x simultaneously. When then apply the transformation simultaneously to each actor. Hence one step of the algorithm may be described as changing the vector of tactics x as follows:

x → (BRVq(1,x), BRVq(2,x), …, BRVq(n,x))

instead of the sequential transformation:

x → (BRVq(1,x), x-1), x → (BRVq(2,x), x-2), …, x → (BRVq(n,x), x-n)

For lack of space we do not include the result of comparing the two approaches. The short summary is that they are very similar. A more detailed analysis shows that the parallel approach converges faster in the early cycles but slower in the last runs. For the practical application to the CGS problem, we found that there was no sufficient difference between the two options.

Many of the situations without Nash equilibriums are resolved in the real world precisely though a *maxmin* evaluation, that is, when each player considers the possible reactions of other players to her/his moves [23]. This consideration may be the fruit of experience in a repeated game or the result of a thought experiment (e.g. chess players). We can introduce this “*maxmin*” evaluation in our GTES model as follows. First we define a new valuation function for tactics:



This means that the evaluation of a given tactic *ti* for an actor *i* is obtained by considering sequentially all possible “best responses” from other actors, when “best response” means to explore the neighborhood of *tj* according to *V* . With this new valuation, we can extend the concept of a Nash equilibrium to what we will call a *Forward Nash Equilibrium (FNE)*:



To extend GTES to search for FNE, we need to extend the search of a local improvement move through the neighborhood structure V. This requires to replace the inner local optimization loop that computes BRV(i,x) by a double loop:

* The first loop explores all possible moves from *V(ti)* – as previously (Section 3.3) – that is, consider all tactics *x* that belong to *V(ti).*
* For each possible new tactic from this neighborhood, we look for the best response for all other players. This means that we recursively enumerate, for all *j* different from *i*, the tactics that are reachable in *V(tj).* We look for the tactic that maximizes the satisfaction of the actor *j* and record the satisfaction of the first actor (*i*).

The evaluation of a tactic vector *x* is the *min* of all recorded value (the worst possible response from one of the other player). The goal of the first loop is to maximize the minimum satisfaction (hence the “*maxmin*”).

We have applied the search for FNE in the CGS example (cf. Section 4.2) because the competitive nature of the business problem yields many “non-stable” situations, with no Nash equilibriums. We applied this FNE approach with only 50 iterations. When we ran the FNE algorithms over a large number of experiments, we obtained a clear improvement in stability (i.e., percentage of “stable” trajectories as defined in Section 3.3), and the behavior of the players became “less aggressive”, resulting in an overall higher satisfaction. There is, however, a significant problem with this approach, namely the computational time. It takes two hours on a fast PC to run the 50 iterations of the FN search loop, as opposed to a few minutes required for the 400 iterations of the BR loop. We will return to the performance issue in Section 5.

4. Applications

4.1 Computational Experiments with Smart Grids

GTES is a framework, which has been implemented as a library. The part that simulates each situation is specific to each problem, but the control algorithms that implement learning (local optimization) and randomization (Monte-Carlo) are generic. The S3G model is implemented with three files: a data model which describes the structure from Figure 1, a simulation file that implements the behavior of the different players as summarized in section 2.2, and a “control” file that contains the generic GTES methods. The following figure shows, on the right part, a rough summary of the simulation loop that is run for each time period (3 hours, hence 8 times per day). On the left part, it shows the three main GTES generic procedures that were introduced in Section 3 [1].

**

**Fig. 4** S3G GTES Architecture

A “serious game” session is made of interactive runs of “experiments”, which are GTES computational executions. More precisely, an experiment is defined through two things:

* The randomization boundaries, for those parameters that will be sampled using Monte-Carlo technique (Section 3.4).
* Some specific values for some parameters, since the goal of a “serious game” is to play “what-if scenarios”, by explicitly changing these parameters. For instance, we may play with the investment cost of storage, to see if storage is or will be critical to smart grids

A GTES simulation run returns the average and standard deviation of a few key business parameters, as well as some indication of the Nash convergence. In the S3G instance, finding a pseudo-Nash equilibrium is simple and all trajectories are stable. Giving averages and a few deviations is a poor restitution of the rich data gathered during the computational experiment, but the goal is simply to “get a feeling for what is happening”, as opposed to producing a forecast. The following table shows some results obtained with a list of experiments designed to understand the main issues that were exposed in Section 2.2. This is a simple experiment, with a fictional country somehow similar to France decomposed into 10 regions/cities. We only picked 7 resulting parameters, whereas a typical output is between 20 and 50 pairs (average/variation). This table shows 8 experiments that may be defined as follows:

* The “default” is a reference point, from which “what-if” sensitivity analysis is made. The economic parameters are set in such a way that alternate operators start with a 20% market-share and should be able to increase it if they demonstrate a better management of variability.
* The second experience raises the variability of energy consumption (globally), while the third experience raises the local variability (each city is more different from each other)
* The fourth experiment doubles the fossil energy price (gas and coal). In the default scenario, it is randomly drawn between 20€ and 40€/MWh.
* The fifth experiment imposes a 5% reduction of the nuclear assets for the supplier during the first 5 years.
* The sixth experiment sets a carbon tax at 100€/t, the proceeds of which is used by the “regulator” to subsidize green energy investment.
* The last experiment is a small variation of the first one, where wholesale prices are more rigidly constrained.

GTES is a “serious gaming” framework, whose value is *implicit learning* while running multiple experiments, and playing “what-if” scenario. Hence it would be illogical to see the previous table as a computational results from which conclusions may be drawn. Furthermore, this table is not really meaningful without the full list of hypothesis that is part of the randomization/systemic parameter settings. The expected use of such results is to show them to a domain expert who will instantly criticize some of the figures (there is not enough “negaWatt”, your fossil price is too low, etc.), propose an alternate value … and the “serious game” begins ! Most of the time, the benefit of a “serious game” session is to help oneself understand a few things that are “obvious” in hindsight, but not so much when you start.

**Table 1. A few experiments with Systemic Simulation of Smart Grids**



This being said, here are a few findings that may be drawn from the hundreds of runs made with the S3G model, and that are worth sharing:

* There is a systemic benefit of distribution and autonomy to cope with variation. This is shown in our second and third line. It is a “subtle” variation (small effects), which means that the economy of the local operator is dominated by its capacity to operate at much lower customer management costs than the supplier.
* CO2 tax increases play a very small role, and one that is difficult to anticipate since it both favors the local operator (support green subsidies) and the supplier (raises the difference between fossil and nuclear).
* “De-nuclearization” is a favorable scenario for smart grid operators, as are most regulations that are adverse to the supplier. The obvious limitation is the resulting price increase that reduces the total economy output (and the country’s competitiveness).
* The “community advantage” (that is, the ability for a local operator to better manage the demand-response loop because it is “closer” to its end customer) is marginal, and it is quite unclear if the payback from demand-response management is enough to sustain the operator’s business model.
* Investing in local storage is never an interesting option (at current prices). We needed to slash the price by over an order of magnitude to see a viable payback in less than 10 years.
* There is a clear competition between local operators and suppliers. The learning component of GTES makes for “agile” players who react closely to each other signals. The pricing structure plays an important role (we have only explored a simple variable pricing scheme). A logical consequence is the importance of regulation.
* The results are sensitive to the strategies of the player. A next step for S3G is to build a “strategy matrix” similar to the one shown in the following section. A strategy matrix is a tabular “what-if” sensitivity analysis where we see what happens if the goals of the players (cf. Section 2.2) are changed.

4.2 Various CGS Games

The following figure is drawn from a previous article where GTES was applied to CGS with three players [1]. The *x* axis shows different experiments (E1 to E6), where the parameters define the objectives for these players. Note that the concept of an “experiment” is related to the set of parameter *p* from the formal model in Section 3.2. E1, E3 and E5 represent similar situations but the goals are more and more aggressive. This means that the target figures for EBITDA gets higher from E3 to E1 and E5 to E3. As a matter of fact, for CGS we state the objective as the combination of three goals which are all expressed as a yearly growth rate: EBITDA, market share and sales (turnover). Each even strategy (E2, E4, E6) is a variation of the previous one with a focus on market share and sales (as opposed to E1-E5 which are more “financial” strategies). Figure 5 shows three types of outputs from the GTES simulation. First, we indicate the status distribution, that is, the percentage of trajectories that respectively yielded a stable, war and chaos status, as explained in Section 3.3. Second, we indicate the standard deviation of the overall satisfaction of all players. Last, we print the average satisfaction of each player.

These types of results tell a story about the different competitive situations. Obviously the E6 context is more difficult for all players than the initial situation E1. The percentage of chaotic trajectories yields a kind of “confidence index”. Success for a GTES simulation translates into a large majority of stable trajectories. When this is not the case, we usually analyze the trajectories and, most often, we either find a weakness in the model or a business contradiction in the way the objectives are stated. We also look for a relative ranking of satisfactions, more than the absolute values, when we compare one “situation” against another.

**Figure 5.** Sample of GTES results on CGS problem

In this example, the “strategies” are similar for all players. Obviously we can compare the influence of strategies actor against actor, in a game theory fashion (somehow, we apply game theory twice: once internally in GTES and once when we perform manually a parametric analysis). We illustrate this with another scenario with “four players”, where a new operator is introduced (with no customer base but an aggressive cost structure and a low price). The interest of the CGS model (few macro variables) is that making such an experiment is easy. The following table shows the matrix-analysis of three strategies for the three “original” players:

* S1 is a “conservative strategy” where the goal of the operator is to *maintain* its market share, its turnover and its EBITDA (actually, quite an aggressive goal).
* S2 is a quieter strategy, where the operator expects a small setback when the fourth player is introduced.
* S3 is a “financial strategy” that focuses on maintaining the EBITDA, at the possible expense of market share or turnover.

The matrix compares the strategy picked by the first player (lines) and the strategies chosen by the other two players (columns). We assume that the fourth player (the small one that get introduced) does not vary (mostly, to try to break even as soon as possible). The table shows the satisfaction of the first player against the satisfaction of the two others.

**Table 2.** A strategy matrix for CGS games

|  |  |  |  |
| --- | --- | --- | --- |
| Player 1: | Players 2 & 3: S1 | Players 2 & 3: S2 | Players 2 & 3: S3 |
| Strategy S1 | Sat : 51% vs 5/74% | 98% vs 5%/61% | 6% vs 5%/76% |
| Strategy S2 | 23% vs 47/97% | 80% vs 57%/98% | 0% vs 87%/96% |
| Strategy S3 | 45% vs 9%/61% | 97% vs 5%/52% | 5% vs 14%/52% |

This type of matrix tells that a more aggressive choice is a better strategy for Player 1, but it is also full of interesting insights. For instance, first column – second row shows that the satisfaction of the first player is worse although its goals are easier to reach. Even when all three players have a moderate strategy (second column, second row) the EBITDA is poor (remember that a fourth player is introduced). On the contrary, although the satisfaction of the first player in the first column-third row is only 45%, this translates into a better EBITDA than the “default” strategy. The real benefit from GTES is not what is obtained from such a result matrix but rather what is learned through a number of experiments, or through a “serious game” scenario. This is precisely the same argument made for the S3G (previous) example. Here is a list of more global insights that were derived by running many experiments for a couple of months:

* The best strategy for a small player is to be slightly more aggressive than the bigger ones, but not too much.
* Defining mostly financial goals for the players yield a stable game (i.e., most trajectories are characterized as stable since a pseudo-equilibrium is found). The search for market share growth is quite a different story that yield either chaotic or war trajectories (which ends with the failure of one of the players).
* If the strategies coincide towards the search for a global increase of profit, acquisition costs are lowered down while loyalty expenses (for renewals) are raised up.
* When competition increases (when all players pick more aggressive strategies), a price pressure occurs, which is then stabilized with respect to the EBITDA goal.
* This price war is favorable to large players, or more precisely, players with the best fixed/variable cost structure (who tend to be the large players, but not always).
* Competitions that generate stable trajectories display a form of “mimetic” behavior between players: price evolutions follow similar patterns. The optimal tactic found by machine learning is both prudent (price slashing occurs through small decrements) and coupled to each other’s behavior.

Many of these insights either relate to common sense (something that most managers would say they knew already) or to previous results of economic studies of competition in an open market. However, each insight is still valuable because it is illustrated with “trajectories” that are closer to the business situation than most analytical models.

4.3 Two Other Applications from the Telecom Industry

We have used GTES over a number of different real-life problems during the past 10 years. Here we briefly mention two problems for which GTES has shown to be useful. The first example models the two first years of a customer lifecycle. A customer enters a shop from one of the six possible distribution channels (such as an online shop). The customer picks a package (phone and service plan) and uses the product accordingly for a year. Then, depending on the current price levels and the evolution of mobile phone technology, she or he may decide that time has come to replace the phone. A double choice is made: either to simply buy a new one or to “renew” the phone – while keeping the service plan – by taking advantage of the operator’s renewal proposition. A second year of phone use is then simulated so that the customer “total two year value” may be shared between the actors.

This simple model is interesting because it describes the coupling between the phone operators through distribution channels. By optimizing their own revenue, distribution channels have a clear impact on the competition between operators. A key insight of one executive at Bouygues Telecom was to envision that “channels could think in term of full lifecycle too” and favor those operators who pay more “over a complete cycle of sales & renewal”. To experiment with this model, two ingredients were needed that were absent from the spreadsheet model: to factor in the price-based competition and to optimize the behavior of the distribution channels automatically as a response to the operator’s strategy. This required building a simple model on how customers would react, during each phase of the game, to price changes made by the channels. The main result achieved with GTES simulation was to confirm that it makes a lot of sense for the operator to put himself in the shoes of the distribution channels and think in terms of lifecycle. As such, it would not be worth a computational experiment, but the GTES serious game is able to translate this intuition into real numbers, where one can grasp the sensitivity of the equilibrium building (how fast do other actors react to one’s own moves).

The second example deals with the allocation of resources to distribution channels, including those controlled by the operators (branded shops and web sites) for which long-term investments can be made. A critical issue is the level of internal competition between the channels for one given operator. This is hard to measure, hard to evaluate and there is no consensus among the operational experts of the subject matter. This second model was built as a tool to illustrate various scenarios and make those experts react to these scenarios.

The model is also quite simple, but is built on top of two qualitative equations whose parameters are unknown. The first one describes how customers react to price changes from the channels. The second qualitative model is a simpler channel competition matrix, which represent how likely one would switch from one channel to another. This matrix focuses the wide spectrum of opinions: for some, there was no issue, the customer would go to where the price was lower; for others, people would shop where it was convenient and where they were used to go, especially for specific demographics. Hence we develop and used GTES, to capture and play with this uncertainty and wide range of conflicting opinions. The second example – optimizing distribution costs over different channels – showed the value of practical demonstration over theory. GTES illustrates the concept of “elasticity” and how different channels can cannibalize each other. The value of the experiment does not come from the elasticity parameters (which are a “wild guess” at first) but from the numerous interaction with the channel managers who react to the GTES outcome and build, through successive iteration, a commonly accepted picture of this customer elasticity to price variation.

5. Discussion and Future Directions

Most models are quite simple from a computational point of view – tens to hundreds of parameters and tens of equations - but are already more complex than typical spreadsheet applications, because of loops and temporal series. With the smart grids example, one computation (i.e., computing *fp(x,e)*) takes up to one second (on a fast personal computer). In the three telecom examples, computing *fp(x,e)* takes between 10 to 100 ms. However, the multiplying factor to run GTES is between 106 and 109. Indeed, we need to multiply three factors (three nested loops). Finding the best response tactic (*BRV(i,t)*) typically requires between 100 and 1000 optimization cycles. Searching for pseudo-Nash equilibriums usually requires a few hundred iterations (cf. Section 3). Interweaving yields a gain of a factor of 10. Last, Monte-Carlo sampling requires computing between a few hundred up to a few thousand trajectories. All this translates into total run time ranging from one day to a year (with a complex model or if Forward Nash is introduced). Since this is obviously not practical, we compromise for less precise/stable experiments: we both reduce to sampling size and reduce the time allocated to Nash convergence. We still use long-running sessions to perform the analysis and the fine-tuning (from an hour to a day of computing), and run simpler/faster experiment when playing the serious game (a few minutes). However, it is clear that GTES is ideally suited for parallel computation, since the search for the best response in V may be parallelized (using classical local search techniques), the search for the pseudo-Nash equilibrium can process the moves in parallel and Monte-Carlo sampling is obviously a candidate for parallel evaluation.

One major benefit of the GTES approach is that it is a true “torture test” for models. The combination of random sampling and machine learning (which will find and exploit all breeches in the model if there is a way to increase satisfaction through a poorly calibrated equation) “shake” the input model and quickly point out faults. A very similar observation was made ten years ago when we worked on automatic generation of optimization algorithm for vehicle routing [24]. Using machine learning coupled with random exploration technique over parameterized algorithms is an effective way to discover bugs and limitations. In particular, GTES requires a stable behavior of the model “at its boundaries”. Very often, the simpler equations of a “naïve model” are very rough when the values get close to their limits of validity. One of the benefit of S-curves (over linear equations) is precisely to combine the differential analysis of linear/derivative approaches with an overall “global system” approach (defining overall boundaries). Another complexity comes from the multi-criteria nature of “player’s satisfaction”. It usually takes a while to tune the formula that defines satisfaction (a weighted combination of terms that tell how far the player is from his strategic goals – cf. Section 2.2 for the S3G example). For GTES to deliver interesting “serious gaming”, the satisfaction objective function needs to be tuned with domain experts.

In addition to model definition, neighborhood structure and exploration strategies are key choices as far as performance and quality of the results are concerned. The performance part is rather obvious and was mentioned earlier: larger neighborhood and more sophisticated local search strategies increase the computing time. The second point is that if the local search strategy is too simple, experience shows that we get fewer stable strategies. There is a compromise to be found: in many cases we had to increase the neighborhood structure to obtain a satisfactory game, that is, to ensure that all “logical answers” (or defined as such by domain practitioners) were actually found by the local search phase of GTES. The criterion that we use is to make sure that local search always gets the same performance level irrespectively of the initial (randomly chosen) tactic. We have experimented with genetic algorithms in the past [1], and we plan to experiment with more complex neighborhood structures and Tabu search to improve the speed and accuracy of learning, which translates into better performance and stability for GTES.

6. Conclusions

Complex problems in an enterprise require *learning* more often than they require *solving*, because of the uncertainty, the feedback loops with the environment and the rapidly changing nature of business. The GTES approach is a practical toolbox to play with under-specified models and transform them into serious games. Through the combination of techniques from evolutionary game theory, operations research and stochastic simulation, GTES is a workbench where complex models may be simulated and interacted with. GTES is a generic approach, which could be used for a variety of problems. We have shown here a few applications, ranging from complex and speculative models, such as “Systemic Simulation of Smart Grids” to the very practical business applications of Section 4.3. GTES has been used, to give another example, to evaluate the best strategies for the 2011 800 MHz LTE bidding in France.

The value of the GTES approach is demonstrated through *experiments*, when the strategies of one or a few players are changed and we may observe how the other actors would react. This is most of all a learning experiment, which is why the model needs to be simple (i.e. each business variable needs to make “business sense” for the human players). When the outcome seems surprising or counter-intuitive, it is often necessary to look at a few trajectories, which is why “white box” approaches are preferred. If the goal is to run simulation as a forecasting tool, a “black box” may work, once the technique has acquired some credibility, but if the goal is to learn from practical experiments, it is crucial that the participants understand how the model works.

References

1. Y. Caseau (2009). *GTES: une méthode de simulation par jeux et apprentissage pour l’analyse des systèmes d’acteurs*. RAIRO operations research, vol. 43, n° 4.

2. J. Weibull (1995). *Evolutionary Game Theory.* The MIT Press.

3. R. Gibbons (1992). *Game Theory for Applied Economists*. Princeton University Press.

4. Korn G. A.. *Advanced Dynamic-system Simulation: Model-replication Techniques and Monte Carlo Simulation*. Wiley Interscience, **2007**

5. B. Slantchev (2004). Game Theory: Repeated Games. University of California – San Diego. <http://polisci.ucsd.edu/~bslantch/courses/gt/07-repeated-games.pdf>

6. D. Fudenberg, D. Levine (1998). *The Theory of Learning in Games.* The MIT Press.

7. E. Aarts and J. K. Lenstra(1993). *Local Search in Combinatorial Optimisation*. Wiley.

8. [S. Jørgensen](http://www.amazon.com/exec/obidos/search-handle-url/002-2561564-3320824?%5Fencoding=UTF8&search-type=ss&index=books&field-author=Steffen%20J%C3%B8rgensen), [M. Quincampoix](http://www.amazon.com/exec/obidos/search-handle-url/002-2561564-3320824?%5Fencoding=UTF8&search-type=ss&index=books&field-author=Marc%20Quincampoix), T. Vincent (Eds) (2007). *Advances in Dynamic Game Theory: Numerical Methods, Algorithms, and Applications to Ecology and Economics (Annals of the International Society of Dynamic Games)*. Birkhauser Boston.

9. N. Nissan, T. Roughgarden et al. (2007). *Algorithmic Game Theory*. Cambridge University Press.

10. R. Duncan Luce, H. Raiffa (1957). *Games and Decisions – Introduction and Critical Survey*. Dover Publications, New York.

11. F. Alkemade, H. La Poutré, H. Amman (2006). *Robust Evolutionary Algorithm Design for Socio-economic Simulation.* Computational Economics n° 28, p. 355-380.

12. P. Siarry, J. Dréo *et al*. (2003). *Métaheuristiques pour l’optimisation difficile*. Eyrolles, Paris.

13. R. Horst, H. Tuy (2010). *Global Optimization : Deterministic Approaches*. Springer, Berlin.

14. M. Milano (2004). *Constraint and Integer Programming*. Kluwer Academic Publishers.

15. R. Axelrod (1997). The Complexity of Cooperation- Agent-Based Models of Competitions and Cooperation. Princeton University Press.

16. R. Nelson, S. Winter (1982). An Evolutionary Theory of Economic Change. Belknap Harvard.

17. J. Ferber (1999). Multi-agent systems: An introduction do distributed artificial intelligence. Addison-Wesley.

18. M. Kandori, G. Mailath, R. Rob (1993). *Learning, Mutation and Long Run Equilibria in Games.* Econometrica, Vol. 61, N° 1 (January, 1993), pp 29-56.

19. A. Blum, M. Blum, M. Kearns, T. Sandholm, M.T. Hajiaghayi (2006). Machine Learning, Game Theory and Mechanism Design for a Networked World. NSF proposal, <http://www.cs.cmu.edu/~mblum/search/AGTML35.pdf>.

20. J. Forrester (1971). *Principles of Systems*. System Dynamics Series, Pegasus Communications, Waltham.

21. J. Sterman (2001). *Business Dynamics – System Thinking and Modeling for a Complex World*. McGraw Hill.

22. S. M. Lucas, G. Kendall (2006). *Evolutionary Computation and Games.* IEEE Computational Intelligence Magazine, February 2006

23. M. Bowling, M. Veloso (2000). *An Analysis of Stochastic Game Theory for Multiagent Reinforcement Learning.* Carnegie Mellon University, CMU-CS-00-165.

24. Y. Caseau, [G. Silverstein](http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/s/Silverstein:Glenn.html), [F. Laburthe](http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/l/Laburthe:Fran=ccedil=ois.html) (2001). *Learning Hybrid Algorithms for Vehicle Routing Problems*. [TPLP 1](http://www.informatik.uni-trier.de/~ley/db/journals/tplp/tplp1.html#CaseauSL01)(6): 779-806.