Numerical solution of Nash and Stackelberg equilibria: an evolutionary approach

João Pedro Pedroso

Centre for Operations Research and Econometrics
34 Voie du Roman Pays,
B-1348 Louvain-la-Neuve, Belgium
e-mail: `jpp@core.ucl.ac.be`

Abstract. In this paper we describe evolutionary heuristics for numerically solving systems of several, interdependent optimisation problems. They can be used for the solution of some games with simultaneous moves of the players (Nash equilibria), asynchronous moves (Stackelberg equilibria), or a mix of these situations. The application is possible in cases where the presence of non-convexities, integral variables, or other factors restrain the use of traditional methods, based on derivatives.
The solution of instances of well known economic equilibrium problems with these algorithms is supplied. The results obtained for these simple cases show potential applications of the strategies, and provide limited convergence evidence.

1 Overview

Applications of game theory to the simulation of natural processes has been treated in [6], through an approach there called *evolutionary game theory.* In that text, the assumptions usually made in game theory that players will behave rationally, and according to some criterion of self-interest, are replaced; rationality is substituted by the criterion of population dynamics and stability, and self interest is replaced by Darwinian fitness. The "solution" to a game is given by what is called an *evolutionarily stable strategy* — which is essentially the same as an unbeatable strategy, in game theory. The general approach of that book is to use game theory to model and explain evolutionary processes.

The approach that we follow in this text is the inverse one: we aim at using the computer simulation of evolutionary processes to model and numerically solve some game theoretic problems.

Evolutionary approaches have successfully been applied to solve a large variety of problems; in particular, successful applications to the numerical solution of global optimisation problems have been repeatedly reported. In this text we propose an evolutionary approach to numerically compute equilibria: systems of interdependent optimisation problems. The algorithms for equilibria computation presented here are deeply based on a method for the solution of global optimisation problems. In our implementation we have relied on an evolutionary algorithm described in [2], although equivalent methods could be used instead.

We depart from the solution of a single global optimisation problem to focus on heuristics for the solution of Nash equilibria (i.e., equilibria with simultaneous moves of the intervening players) and Stackelberg equilibria (asynchronous moves: one agent plays before the other, taking into account its reaction)[1]. Extension to a mix of the two types of games is presented afterwards. We then state some assumptions on the type of games that we aim to solve, and to a superficial analysis of the convergence properties.

The essential characteristic of our approach is that each of the optimisation problems, corresponding to the behaviour of a particular agent, is seen as a function of the strategies of the other players.

We conclude by presenting some examples of application to well-known economic equilibrium problems, for whose solution we have implemented an evolutionary computer program. We supply and analyse the numerical results obtained for these test problems.

2 The algorithms

2.1 Nash equilibria

The motivation for the solution method that we propose to systems of simultaneous optimisations comes from the observation that, in life, independent evolution of different species leads to an "optimisation" of the adaptation, or fitness, for each of the species, to the natural environment; this arises even when the behaviour of one of the species has a direct influence on the behaviour of the others. This fact leads to the assertion that an algorithm that is based on the simulation of life (as is a genetic algorithm), when extended to support the simultaneous existence of several species, should provide a way to solve equilibria.

The principle used in the algorithm that we propose for the computation of Nash equilibria is the following. In a Nash game, each of the players optimises its utility, or objective, subject to external parameters, which are the strategic variables of the other players, set up by them. It can, henceforth, be seen as a system of simultaneous optimisations, which we can solve using a fixed point iterative method.

We start the algorithm by setting up one evolutionary optimisation problem for each of the players in the game, taking the variables of all the other players as fixed. Each of these problems keeps a separate population, that will evolve through selection and reproduction. The series of evolutionary optimisation problems are assembled in a meta-problem, the Nash equilibrium problem.

The meta-evolution process starts by randomly initialising the population of each of the optimisation problems (i.e., each of the inner evolution processes). An element of the population of each of these problems is (randomly) chosen as the initial move of each player, to which we attribute an evaluation of $-\infty$ (assuming that we are treating of maximisation problems).

[1] A Stackelberg game can also be seen as a Nash game with incomplete information; see [1]

An iterative process starts now. A new generation of the population is created for the optimisation problem of the first player, through selection and the search operators generally used for evolutionary algorithms: crossover and mutation. Each of the individuals in the offspring generation is evaluated, by computing the objective function of this player subject to the moves announced by all the other players. From this population, the individual with the highest fitness is selected as the new move of the first player. A new generation is created this way for all the other players, and their move updated as above. At the end of the first meta-generation we have the first complete move of the Nash game. The iterations proceed until all the iterations of the meta-program are accomplished, or until some convergence criteria for the strategic variables is achieved.

The solution of the Nash equilibrium is the last strategy obtained in the meta-program. As argued in section 3.1, if certain assumptions hold, the limit of this process when the number of generations tends to infinity is a Nash equilibrium.

See figure 1 for pseudo code describing the evolution approach for Nash equilibria computation.

```
t := 0                                              Start with an initial time.
CreateStructure(strategies(t))     Create a structure where to hold the moves of all the players.
CreateEvolOptim(nash(t), strategies(t))   Create an evolutionary optimisation problem for each of
        the players. Each of these problems is a function of the strategies structure: its objective depends on
                                                   the moves announced for all the other players.
RandomInit(nash(t))                 Randomly initialise the population of each of these problems.
strategies(t) := RandomSelect(nash(t))          Choose randomly the initial move for each of the
            players; this corresponds to a random individual in the population of each evolutionary program.
iterate                                             Start the meta-evolution process.
    t := t+1                                        update the time counter
    nash(t) := NewGeneration(nash(t-1),strategies(t-1))    Evolve each of the optimisation
                                                    problems a generation further.
    strategies(t) := SelectBest(nash(t))       Update the moves: for every player, select the best
                                                    solution found in the current generation.
until Terminated()        Termination criteria: time, slack between strategies(t) and strategies(t-1), etc.
display solution                         Solution is the latest move, stored in strategies(t).
end
```

Fig. 1. Evolutionary algorithm for the computation of Nash equilibria. The problem of each of the players is a function of the moves of the other players; the system of these simultaneous functions is solved through a fixed-point iteration.

Elitism and Nash equilibria As described in [3], for the evolutionary algorithm that performs the optimisation for each of the Nash players to converge, it must rely on an elitist mode: the best element in the population should be inserted unchanged in the following generation. Elitism here must be redefined. The solution of each optimisation problem is parameterised on the solutions announce by the other players. Hence, the best solution found in the population of a particular problem will, in general, change when the other problems evolve. This means that the elite found in an iteration will, in general, no longer be the elite in the next iteration.

What we propose to tackle this problem is to evaluate all the individuals in the population at the end of each generation, and select the move of the player as the fittest of these. At the beginning of following iteration, when the moves of the other players are updated, we reevaluate the whole population, and choose the elite as the best of its individuals. This individual will be inserted unchanged in the following generation. Notice that the evaluation of this individual may be worse than best evaluation of the preceding generation.

Using this process ensures that, at the solution of the fixed point problem, the optimisation problems do not diverge – although the fixed point method itself may diverge. It henceforth replaces the standard elitist mode of evolutionary algorithms, with the only drawback of having to perform a supplementary evaluation, per iteration, for the whole population.

2.2 Stackelberg equilibria

The approach used to solve Stackelberg equilibria is even simpler than that used for Nash equilibria. As we provide a way of setting up a global optimisation problem as a function of some exogenous parameters, what we do is to set up the maximisation problem of the follower, given an arbitrary move of the leader. Then use this "functor" inside the objective function of the leader. Each time the objective function of the leader is called, it starts by computing the reaction of the follower to the current leader's strategy; afterwards, the evaluation of the objective of the leader is computed, taking into account this reaction. Hence, at each computation of the objective of the leader, an inner optimisation problem (that of the follower) is solved.

See figure 2 for pseudo code describing the evolution approach for Stackelberg equilibria computation.

As can easily be imagined, the computational burden inherent to this method is rather high. Several ways can be implemented to tackle this; the simplest is the following: limit to the number of generations spent solving the follower's problem to the number of the current generation in the leader's problem. This method ensures that at the beginning of the process, when the solution of the leader is still rather imprecise, the answer computed to the follower is also imprecise. As we go further in the number of generations for the leader, the number of generations spent in each follower problem is also increased, increasing its precision. At the end of the evolution process, when we get closer to the equilibrium move of the leader, the reaction move of the follower is also computed with a large number of generations, and hence with an increased precision[2].

[2] Another way consists on the implementation of an interpolation method for the determination of the followers reaction. This can be achieved in two ways. The first one is to build a database with follower's reactions for values covering all the search

```
t := 0                                                    Start with an initial time.
CreateStructure(ldr_strategy, flr_strategy)     Create the data structures that hold the leader's and
                                                                the follower's strategies.
leader(t) := CreateEvolOptim(ldr_strategy, flr_strategy)          Create an evolutionary
                                                         optimisation problem for leader...
follower(t) := CreateEvolOptim(ldr_strategy, flr_strategy)        ...and for the follower.
RandomInit(leader(t),follower(t))      Randomly initialise the population of each of the problems
iterate                                                 Start the meta-evolution process.
    t := t+1                                                      update the time counter
    leader(t) := NewGeneration(leader(t-1),follower(t-1))       Create the new generation of
        the leader's optimisation problem, using the standard selection and reproduction operators, and the
                                                             evaluation function coded below.
    ldr_strategy := SelectBest(leader(t))        Set the leader's strategy: select the best solution
                                                                   found in the last iteration
    flr_strategy := Solution(follower(ldr_strategy(t)))          Set the follower's strategy: the
                                                              reaction to the leader's solution
until Terminated()     Termination criteria: time, slack between strategies(t) and strategies(t-1) , etc.
display solution                                     Solution: the strategies computed above.
end


Evaluation(ldr_move) :                          At each evaluation of the leader's objective, do:
    ldr_strategy := ldr_move                    Set the leader strategy as the move currently tried.
    flr_strategy := Solution(follower(ldr_strategy))   Evaluate the follower's reaction: solve its
                                        optimisation problem based on the current move of the leader.
return LeaderObjective(ldr_strategy,flr_strategy)      Evaluate the leader's objective, considering
                                                                        the follower's reaction.
```

Fig. 2. Evolutionary algorithm for the computation of Stackelberg equilibria. At each solution point, the objective of the leader evaluates the reaction of the follower, by solving its optimisation problem parameterised by the current leader's solution.

The elitist procedure for the leader's problem must be as described for the Nash equilibria; the follower's problem can be solved using standard elitism.

3 Convergence properties

Convergence of the evolutionary algorithm that we use for global optimisation problems has been proved in [3]. In this context, convergence means that, with

space of the leader. Then, at the evaluation of the leader's objective, one could interpolate values in this database in order to obtain the reaction values, or train a neural network to compute it. The other possibility is to dynamically store the values of the reaction, at the time we are computing the leader's objective. The follower's move would be computed by an optimisation if there are no data available in the "neighbourhood" of the leader's move, and interpolated if those data exist. The neighbourhood criteria could be adjusted dynamically, being more exigent as the program approaches the end.

probability one, we obtain a monotone sequence of objective function evaluations which converges to the supremum (for maximisations) of the objective function, in the search domain, as the number of iterations tends to infinity.

Convergence for Nash and Stackelberg equilibrium problems depends on the convergence of each of the inner global optimisation problems, and hence, at the best of the scenarios, is bound to a sequence of function evaluations that converges to the corresponding evaluations at the equilibrium.

3.1 Convergence for Nash equilibrium problems

Conditions for ensuring the convergence of a fixed point method, where each of the equations of the system is a parameterised global optimisation problem, are rather difficult to state in a general way.

Examples of cases where convergence for a fixed point does *not* occur are easy to find. In order to be able to theoretically ensure convergence we would have to enforce rather restrictive conditions on the type of global optimisation problems, such as concavity and continuity. As these conditions are not verified, in general, for global optimisation problems, there is no reason to enter into deep detail in the theoretical analysis of the convergence.

Although we do not prove it, we claim that this algorithm converges to the solution of a Nash equilibrium if the following conditions hold:

1. The solution exists;
2. The solution is stable, i.e., the fixed point iteration leads to an equilibrium;
3. At the solution point, the maximisation problem for each of the players is a regular optimisation problem, if we keep the variables for all the remaining agents fixed.

By a regular optimisation problem we mean that there exists a subset of the feasible region contiguous to the optimum with a positive volume (see also [5, 4]).

We must stress, nevertheless, that in some situations fixed point iteration does not lead to a Nash equilibrium, and hence the heuristic that we propose cannot assure convergence for general problems. Therefore, the user must carefully verify the solution obtained.

3.2 Convergence for Stackelberg equilibrium problems

Stackelberg equilibrium problems are trivially assured to converge as long as the optimisation problems for each of the players are regular. Two aspects should be asserted for guaranteeing convergence:

1. the follower's problem is regular all over the leader's strategy domain;
2. the leader's problem is regular all over the follower's strategy domain;

These conditions are sufficient to ensure that the meta-problem is regular.

4 Examples

The examples presented here are academic classical cases in economics: Nash-Cournot and Stackelberg competition between firms that produce the same, homogeneous product. The examples are simple, in order to be possible to analytically verify the solutions obtained.

For the sake of simplicity of the analytical treatment, the demand function used is linear, with the following form:

$$P(x) = a - b \cdot x$$

where x is the quantity and $P(x)$ is the inverse demand function. For simplicity too, costs are supposed to be null:

$$C(x) = 0$$

Notice that although the examples presented are continuous and differentiable, the algorithms described can be used in much more general situations. For a more extensive set of test problems, the reader is referred to [4].

4.1 A Nash-Cournot game

Profits in a Nash-Cournot oligopoly with two identical firms, i, are given by:

$$\begin{aligned} \Pi_i(x_1, x_2) &= P(x_1 + x_2) \cdot x_i - C_i(x_i) \\ &= x_i[a - b(x_1 + x_2)] \end{aligned}$$

for firms $i = 1, 2$. The solution of the (linear) system of equations of the first order conditions leads to $x_1^* = x_2^* = \frac{a}{3\,b}$

We have used the algorithm for the computation of Nash equilibria described in this paper to obtain a numerical solution of this problem. Using the parameters $a = 10$ and $b = 1$, with 100 iterations, we achieved the correct solution, $x_1^* = x_2^* = 3.333$ and $\Pi_1^* = \Pi_2^* = 11.111$. A log of the evolution of the fixed point iteration is presented in figure 3.

4.2 A Stackelberg game

In a Stackelberg oligopoly with two firms with null costs, the profit of the follower firm is given by:

$$\begin{aligned} \Pi_f(x_l, x_f) &= P(x_l + x_f) \cdot x_f - C_f(x_f) \\ &= x_f[a - b(x_l + x_f)] \end{aligned}$$

where the index f stands for the follower, and l stands for the leader. The solution of the first order condition associated with the follower's profit leads to its reaction function:

$$\psi_f(x_l) = \frac{a - b\,x_l}{2\,b}$$

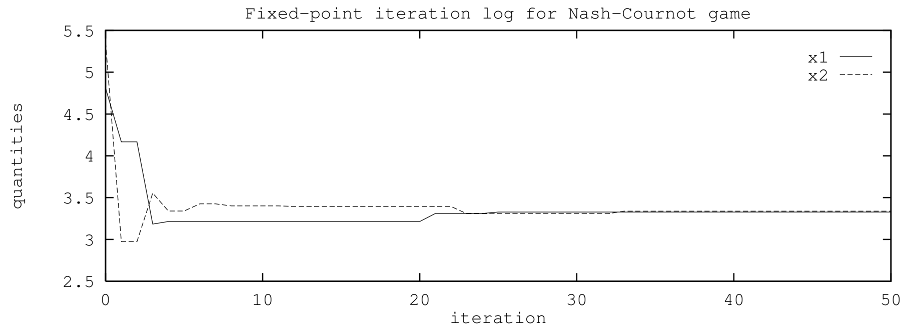

Fig. 3. Fixed-point iteration log for a Nash Cournot game with two players: quantities determined by the algorithm as a function of the iteration number.

The profit of the leader is then given by:

$$\begin{aligned}\Pi_l(x_l) &= P(x_l + \psi(x_l)) \cdot x_l - C_l(x_l) \\ &= \frac{x_l(a - b\, x_l)}{2}\end{aligned}$$

The first order condition for the optimisation of the leader's profit drives to the determination of its optimal quantity, $x_l^* = \frac{a}{2\,b}$. This, in turn, leads to $x_f^* = \frac{a}{4\,b}$.

This problem has been numerically solved with the algorithm for Stackelberg equilibria computation presented in this paper. For the parameters $a = 10$ and $b = 1$, with 100 iterations, we obtained the correct solution, $x_l^* = 5.000$ and $x_f^* = 2.500$, with profits $\Pi_l^* = 12.5$ and $\Pi_f^* = 6.25$. A log of the evolution of the iterative process is presented in figure 4.

4.3 Stackelberg firm followed by a Nash-Cournot game

We consider now the situation where a leader firm is followed by two other firms, which play a Nash-Cournot game between them, taking into account the leader's quantity. In this oligopoly, if firms have null costs, their profit is given by:

$$\begin{aligned}\Pi_j(x_l, x_{f_1}, x_{f_2}) &= P(x_l + x_{f_1} + x_{f_2}) \cdot x_j - C_j(x_j) \\ &= x_j[a - b\,(x_l + x_{f_1} + x_{f_2})]\end{aligned}$$

where the indexes f_1, f_2 stand for the followers, l stands for the leader, and j for any firm. The solution of the system of first order conditions for the two followers leads to their reaction functions:

$$\psi_{f_i}(x_l) = \frac{a - b\, x_l}{3\, b}$$

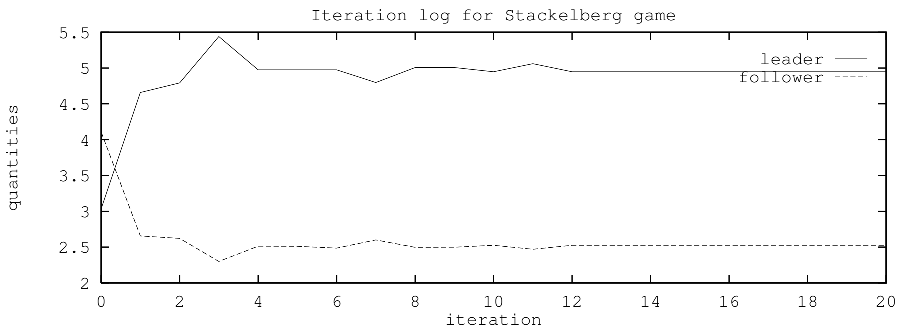

Fig. 4. Iteration log for a standard Stackelberg game. Quantities for the leader and the follower determined by the algorithm, as a function of the iteration number.

for $i = 1, 2$. The profit of the leader can now be determined by:

$$\begin{aligned} \Pi_l(x_l) &= P(x_l + \psi_{f_1}(x_l) + \psi_{f_2}(x_l)) \cdot x_l - C_l(x_l) \\ &= \frac{x_l \cdot (a - b\, x_l)}{3} \end{aligned}$$

The first order condition for the optimisation of the leader's problem leads to the its optimal quantity, $x_l^* = \frac{a}{2\,b}$. This, in turn, leads to $x_{f_2}^* = x_{f_2}^* = \frac{a}{6\,b}$.

We have implemented a hybrid of the two algorithms proposed for the numerical solution of this two stage problem. The output of the second stage, the inner Nash game, was the reaction functions of the two followers, determined through a fixed point iteration. This equilibrium was determined at each evaluation of the leader's objective, considering the input leader's strategy as fixed.

For $a = 10$ and $b = 1$, with 100 iterations, we obtained the correct solution, $x_l^* = 5.000$, $x_{f_1}^* = x_{f_2}^* = 1.667$, with profits $\Pi_l^* = 8.333$, $\Pi_{f_1}^* = \Pi_{f_1}^* = 2.781$. A log of the evolution of the iterative process is presented in figure 5.

5 Conclusion

In this paper we describe evolutionary heuristics for numerically assessing systems of several, interdependent optimisation problems. The approaches can be used for the solution of games with simultaneous moves of the players (Nash equilibria), asynchronous moves (Stackelberg equilibria), or a mix of these.

The application of the algorithms is possible in cases where the presence of non-convexities, integral variables, or other factors restrain the use of traditional methods, based on derivatives.

In many cases, simple fixed point iteration does not lead to an equilibrium, in Nash games. Hence, the method proposed for its solution does guarantee convergence in general. Nevertheless, this strategy converged for the practical examples

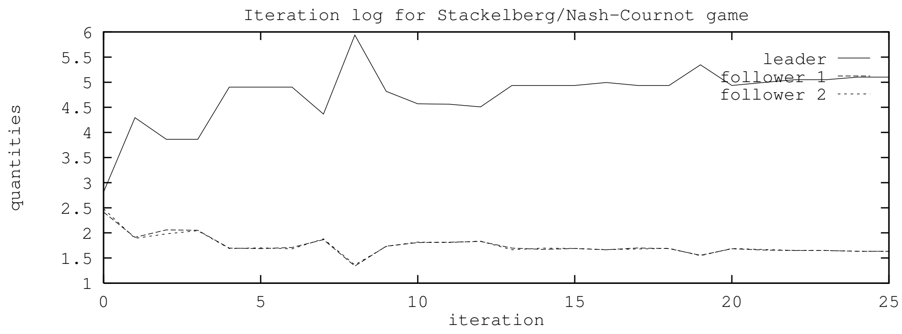

Fig. 5. Iteration log for a Stackelberg game where the leader's firm action is followed by a Nash game between two followers. Quantities determined by the algorithm as a function of the iteration number.

reported in this paper and in [4]. Formal analysis of convergence properties, and improvement of the fixed point method remain as future research directions.

The computational burden inherent to the strategy used for the determination of Stackelberg equilibria restrains the use of the method for medium to large scale problems. However, the acceleration techniques suggested and the continuous progress in the power of computers, bring the method promising for future use.

The examples presented in this paper allowed us to experimentally assert the convergence of the heuristics that we propose to the equilibrium solution for some simple, well known games in economics.

References

1. R. Amir and I. Grilo. Stackelberg vs. Cournot/Bertrand equilibrium. Discussion Paper 9424, Centre for Operations Research and Econometrics, Belgium, 1994.
2. J. P. Pedroso. Niche search: an evolutionary algorithm for global optimisation. In *Proceedings of the Fourth Parallel Problem Solving from Nature Conference*, Berlin, Germany, 1996. Springer. In press.
3. J. P. Pedroso. Niche search and evolutionary optimisation. Discussion paper, Centre for Operations Research and Econometrics, Belgium, 1996. Submitted to publication.
4. J. P. Pedroso. *Universal Service: Issues on Modelling and Computation.* PhD thesis, Université Catholique de Louvain, 1996.
5. H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, 1981.
6. J. M. Smith. *Evolution and the Theory of Games.* Cambridge University Press, 1982.